Réalisation d'une plate-forme de supervision d'instruments analytiques dans les laboratoires à distance grâce aux nouveaux outils et technologies

Dorine Mazeyrat

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE REGIONAL RHÔNE-ALPES

CENTRE D'ENSEIGNEMENT DE GRENOBLE

MEMOIRE
présenté par **Dorine Mazeyrat**

en vue d'obtenir
LE DIPLÔME D'INGENIEUR C.N.A.M.
en INFORMATIQUE

Réalisation d'une plate-forme de supervision d'instruments analytiques dans les laboratoires à distance grâce aux nouveaux outils et technologies

Soutenu le 27 mai 2013

JURY

Président : M. Eric Gressier-Soudan

Membres : M. Jean-Pierre Giraudin
M. Claude Genier
M. Laurent Tardif
M. Arnaud Pierrel

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE REGIONAL RHÔNE-ALPES

CENTRE D'ENSEIGNEMENT DE GRENOBLE

MEMOIRE
présenté par **Dorine Mazeyrat**

en vue d'obtenir
LE DIPLÔME D'INGENIEUR C.N.A.M.
en INFORMATIQUE

Réalisation d'une plate-forme de supervision d'instruments analytiques dans les laboratoires à distance grâce aux nouveaux outils et technologies

Soutenu le 27 mai 2013

Les travaux relatifs à ce mémoire ont été effectués à Persistent Systems France S.A.S. sous la direction de M. Laurent Tardif.

Remerciements

Je tiens tout d'abord à remercier toutes les personnes qui me font l'honneur de participer au jury de ce mémoire :

Monsieur Eric Gressier-Soudan, professeur au CNAM Paris et président du jury,
Monsieur Claude Genier, responsable régional EICNAM Lyon,
Monsieur Jean-Pierre Giraudin, professeur à l'Université Pierre Mendès France de Grenoble,
Monsieur Laurent Tardif, ingénieur logiciel chez Persistent Systems France,
Monsieur Arnaud Pierrel, responsable du site Persistent Systems France.

Je souhaite particulièrement remercier mon tuteur, Laurent Tardif qui a accepté de me guider par ses conseils et dans un même temps m'a fait profiter de ses connaissances et de son expérience professionnelle ainsi que pour la confiance qu'il m'a accordée.

Mes remerciements s'adressent également à toutes les personnes de l'équipe GreenLAB, Frédéric, Isabelle, Jean, Sophie, Sylviane, pour leur sympathie, leurs encouragements ainsi que leur coopération professionnelle tout au long de ces neuf mois de stage.

Je n'oublie pas Bruno Orsier pour ses recommandations et son soutien dans la réalisation de ce mémoire.

Je remercie aussi Matthieu Ayme, Sébastien Duffournet, Jean-Charles Durand et Nicolas Vignard pour leur soutien et leurs conseils ainsi que parents, amis pour leurs relectures.

Merci également à Arnaud Pierrel de l'entreprise Persistent Systems France S.A.S., qui s'est montré compréhensif et m'a encouragée durant mes cinq années de formation au CNAM de Grenoble.

Sommaire

Liste des figures

Liste des tableaux

Liste des abréviations

AJAX : « Asynchronous JavaScript And XML » est une architecture informatique qui permet de construire des pages HTML dynamiques.

API : « Application Programming Interface » littéralement « interface de programmation » est une interface fournie par une application pour autoriser les interactions avec d'autres programmes.

BPM : « Business Process Management » littéralement « gestion des processus métier » est un concept visant à modéliser l'ensemble des processus métier d'une entreprise pour offrir une meilleure visibilité aux responsables.

CDS : « Chromatography Data System » désigne un logiciel de chromatographie qui collecte et analyse les résultats retournés par un intrument analytique.

CSS : « Cascading Style Sheets » littéralement « feuilles de style en cascade » est un langage de programmation qui décrit la présentation des pages HTML.

DAAS : « Data As A Service » littéralement « données en tant que service » est un service capable de fournir un accès à un dépôt de données dans l'informatique dématérialisée.

EC2 : « Elastic Compute Cloud » est un service proposé par Amazon qui permet de louer des serveurs dans l'informatique dématérialisée.

ELN : « Electronic Laboratory Notebook » littéralement « cahier de laboratoire électronique » est un outil électronique pour gérer au quotidien le suivi des expériences chimiques.

FDA : « Food and Drug Administration » est l'administration américaine des médicaments et des produits alimentaires.

GC : « Gas Chromatography » littéralement « chromatographie en phase gazeuse » est une technique d'analyse chimique en phase gazeuse.

HTML : « HyperText Markup Language » est le format de données qui représente les pages web.

HTTP : « HyperText Transfer Protocol » littéralement « protocole de transfert hypertexte » est un protocole de communication client-serveur.

IAAS : « Infrastructure As A Service » littéralement « Infrastructure en tant que service » est un service qui met à disposition des infrastructures informatiques matérielles dans l'informatique dématérialisée.

IHM : Interface Homme-Machine désigne les moyens mis à la disposition d'un humain pour communiquer avec une machine ou une application.

ITIL : « Information Technology Infrastructure Library » littéralement « bibliothèque pour l'infrastructure des technologies de l'information » est un ensemble de bonnes pratiques pour la gestion d'un système d'information.

LC : « Liquid Chromatography » littéralement « chromatographie en phase liquide » est une technique d'analyse chimique en phase liquide.

LIMS : « Laboratory Information Management System » littéralement « système de gestion de l'information du laboratoire » est un outil qui fournit la traçabilité des échantillons chimiques analysés au sein du laboratoire.

MIB : « Management Information Base » littéralement « base d'information pour la gestion du réseau » est un ensemble d'information stocké sur une entité réseau.

MS : « Mass Spectrometry » littéralement « spectrométrie de masse » est un procédé physique d'analyse en phase gazeuse qui permet d'identifier des molécules en mesurant leur masse

MVC : Modèle Vue Contrôleur est un patron de conception logicielle.

PAAS : « Plate-forme As A Service » littéralement « plate-forme en tant que service » désigne un environnement d'exécution mis à dispositon dans l'informatique dématérialisée.

PHP : « Hypertext Preprocessor » est un langage de script pour générer des pages web dynamiques via un serveur web.

POC : « Proof Of Concept » littéralement « preuve de concept » est la réalisation d'une méthode ou d'une idée pour démontrer sa faisabilité.

REST : « Representational State Transfer » est un style d'architecture destiné au web.

RIA : « Rich Internet Application » littéralement « application internet riche » est une application web accessible via le navigateur.

RDA : « Rich Desktop Application » littéralement « application de bureau riche » est une application lancée par le navigateur mais exécutée par le poste de travail.

SGBD : Système de Gestion de Bases de Données désigne le logiciel chargé du stockage et du partage des informations dans une base de données.

SQL : « Structured Query Language » littéralement « langage de requête structurée » est un langage pour effectuer des opérations sur des bases de données relationnelles.

SAAS : « Software As A Service » littéralement « logiciel en tant que service » est un service qui fournit un accès à une application hébergée dans l'informatique dématérialisée.

SDMS : « Scientific Data Management System » littéralement « système de gestion de données scientifiques » est un type de logiciel qui stocke et partage les données scientifiques issues du laboratoire d'analyses chimiques.

SNMP : « Simple Network Management Protocol » littéralement « protocole simple de gestion de réseau » est un protocole de communication pour gérer des équipements informatiques.

SOAP : « Simple Object Access Protocol » est un protocole de service web basé sur le langage XML.

SSH : « Secure Shell » est un protocole de communication sécurisé.

URI : « Uniform Resource Identifier » littéralement « identifiant uniforme de ressource » est une chaîne de caractère qui identifie une ressource sur un réseau.

XML : « eXtensible Markup Language » littéralement « langage de balisage extensible » est un langage informatique de balisage.

Chapitre 1 Présentation

1.1 Introduction

Le domaine de l'analyse chimique est un environnement complexe qui réclame l'utilisation de nombreux logiciels et instruments analytiques. De manière à garantir son activité, un laboratoire d'analyses chimiques a besoin de visibilité sur l'ensemble de ses processus métier afin d'identifier les problèmes techniques, d'automatiser et d'ordonner les tâches afin de maximiser sa production pour un meilleur rendement.

Les outils informatiques facilitent grandement les activités de nombreuses entreprises. Ils sont devenus incontournables et indispensables. Parmi ceux-ci, on peut distinguer les solutions de supervisions qui assument la surveillance des processus métier et des ressources matérielles et fournissent des rapports spécifiques et détaillés afin d'optimiser la gestion des processus. Ces logiciels augmentent ainsi la disponibilité des ressources et garantissent les activités de l'entreprise.

Leur utilisation dans le domaine de l'analyse chimique et plus particulièrement pour la supervision des instruments analytiques dans les laboratoires de contrôle qualité pharmaceutique et pétrochimique n'est pas encore pleinement exploitée. Ils ne sont pas adaptés en l'état aux différents acteurs du domaine métier.

Ce stage suggéré par mon employeur Persistent Systems France a donc pour finalité d'appliquer une solution de supervision adaptée au domaine de l'analyse chimique. Cette solution se doit plus particulièrement de proposer une interface accessible depuis un terminal mobile.

1.2 Persistent Systems

1.2.1 Historique

Le site de Fontaine en Isère, où je suis technicienne-développeuse depuis bientôt six ans, développe depuis de nombreuses années des logiciels scientifiques.

La société française JMBS Développement a été créée en 1991. Elle était spécialisée dans le pilotage d'instruments d'analyses chimiques ainsi que dans l'acquisition et le traitement de données chromatographiques.

Rachetée successivement par les fabricants américains d'instruments Varian fin 2001, puis Agilent Technologies en 2010, elle a été finalement vendue en 2011 à Persistent Systems, une Société de Service en Ingénierie Informatique (SSII) indienne en pleine expansion (cf. figure 1.1).

FIGURE 1.1 : ACQUEREURS SUCCESSIFS DU SITE DE GRENOBLE

La principale force du site français est la composition mixte de ses équipes. Elles sont à la fois composées de développeurs informatiques, mais également de chimistes aux expériences professionnelles riches dans le domaine analytique. Cette mixité fait depuis toujours sa réputation. Persistent Systems a donc choisi de fonder un centre d'excellence en rachetant le site français de l'entreprise américaine Agilent Technologies.

1.2.2 Secteur d'activité

Persistent Systems, entreprise mondiale spécialisée dans le développement logiciel et l'innovation technologique, a été fondée en 1990 à Pune, Inde. La société compte à ce jour 7000 employés et plus de 300 clients sur les continents européen, américain et asiatique (cf. figure 1.2).

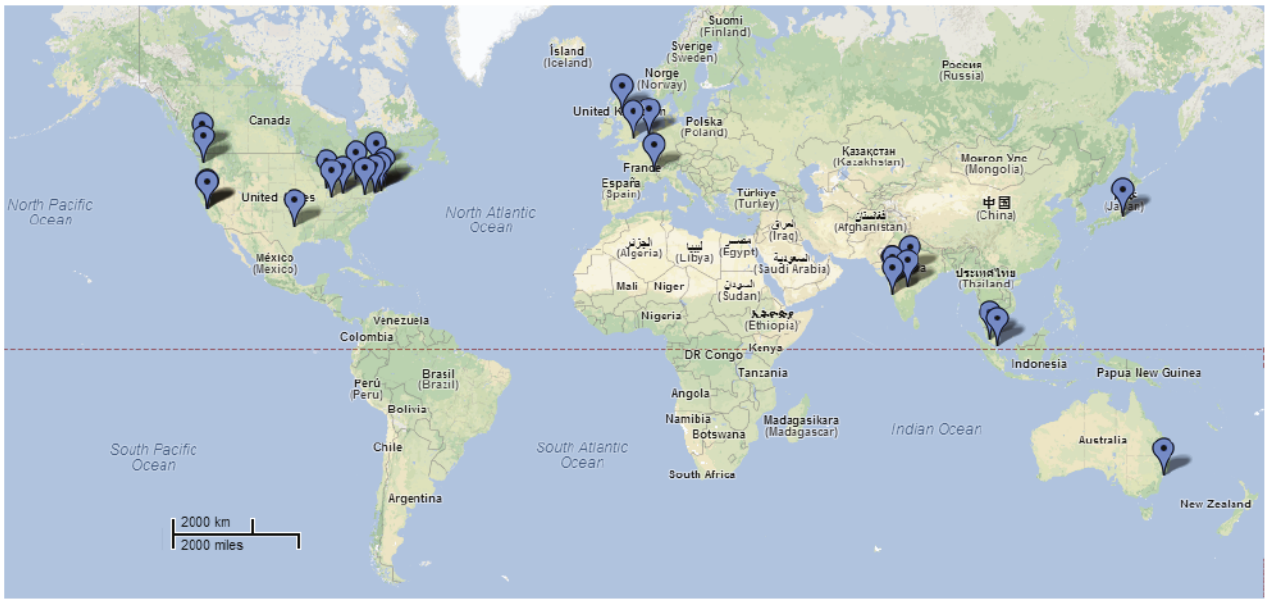


FIGURE 1.2 : IMPLANTATION DE PERSISTENT SYSTEMS DANS LE MONDE

Persistent Systems est depuis plus d'une vingtaine d'années le partenaire d'importants acteurs technologiques mondiaux connus (cf. figure 1.3).

FIGURE 1.3 : QUELQUES PARTENAIRES TECHNOLOGIQUES DE PERSISTENT SYSTEMS

Persistent Systems appuie principalement son développement sur quatre axes technologiques.

Informatique dématérialisée ou « Cloud computing »

Persistent Systems est un investisseur précurseur dans les technologies de l'informatique dématérialisée. La société a développé sa première application utilisant cette technique en 2007. Elle a acquis des connaissances reconnues dans les technologies sous-jacentes à l'informatique dématérialisée. L'entreprise a ainsi établi des centres de compétence dédiés aux plates-formes d'informatique dématérialisée les plus connues : Amazon Web Services, Google Apps, Microsoft Azure et Salesforce [per13a].

Informatique décisionnelle et outils d'analyses ou « Business intelligence & analytics »

L'informatique décisionnelle permet d'exploiter un ensemble de données métier afin d'obtenir une aide à la décision concernant l'activité traitée. Ce type d'application implique l'utilisation de données massives ou « big data » qui désignent des ensembles de données trop volumineux pour être administrés par des outils de gestion de bases de données classiques [per13b].

Plates-formes de collaboration

Les plates-formes de collaboration sont des espaces de travail virtuel qui centralisent tous les outils liés à la conduite d'un projet. Elles sont de plus en plus employées dans les entreprises. Persistent Systems participe au développement de plusieurs solutions créées par des éditeurs de logiciels indépendants [per13c].

Mobilité

Persistent Systems détient également de solides connaissances en développement mobile. En effet, la société initialise et participe à de nombreux développements d'application mobile sur la plupart des plates-formes existantes : Android de Google, iOS d'Apple, BlackBerry OS de BlackBerry, Windows Phone de Microsoft [per13d]…

Pôle sciences de la vie

Le pôle sciences de la vie ou « Life Sciences » de Persistent Systems regroupe 600 ingénieurs et experts métier dans le monde entier qui interviennent sur les marchés suivants :

- Instrumentation en chimie analytique,
- Bio-informatique,
- Appareils médicaux,
- Bases de données pour essais cliniques,
- Dossiers médicaux informatisés,
- Gestion des parcours de soin.

Dans le cadre de son expansion européenne, Persistent Systems a donc fondé Persistent Systems France, un centre d'excellence pour le pôle « Life Sciences », basé à Fontaine en région Rhône-Alpes pôle de recherche français reconnu au niveau européen, grâce à l'acquisition d'une équipe de 30 personnes, spécialisée dans le développement agile de logiciels scientifiques depuis de nombreuses années.

Le domaine de la chimie analytique est vaste et diversifié, l'équipe française travaille plus particulièrement dans le domaine de la chromatographie qui est une méthode d'analyse de séparation. C'est l'une des techniques d'analyse les plus usitées qui peut être employée aussi bien dans le monde pétrochimique que pharmaceutique. Galaxie « Chromatography Data System » (CDS) était le logiciel phare du site de Fontaine. J'ai été intégrée à l'équipe responsable du développement de ce logiciel lors de mon embauche.

1.2.3 Besoins

Le site de Fontaine a toujours travaillé dans le domaine de la chromatographie. Lors du rachat de l'entreprise par Persistent Systems, les droits sur le logiciel Galaxie CDS ont été conservés par Agilent Technologies. Le site français ne peut donc plus l'exploiter directement et doit s'orienter vers de nouveaux projets. Le développement d'un nouveau logiciel de chromatographie ne présente pas beaucoup d'intérêt au regard de Persistent Systems car cela créerait une concurrence avec un de ses plus gros clients du pôle sciences de la vie, à savoir Agilent Technologies. De plus, le développement de ce type de logiciel est long, donc coûteux et de nombreuses solutions existent déjà sur le marché.

Le logiciel de chromatographie utilisé en laboratoire est au cœur de processus métier complexes. De nombreux acteurs, ressources et applications interviennent pour préparer les échantillons qui doivent être analysés. Les résultats obtenus sont ensuite étudiés, validés par la hiérarchie et ensuite enregistrés pour une exploitation future. Les processus métier propres à chaque entreprise peuvent être modélisés par des outils de gestions de processus métier ou « Business Processus

Management » (BPM). Ce type de logiciel favorise la visualisation globale de l'ensemble de ces processus et permet d'automatiser, voire optimiser les différentes étapes de production.

Persistent Systems a donc choisi de développer une nouvelle ligne de produits intitulée GreenLAB articulée autour d'un outil BPM destiné spécifiquement au domaine de l'analyse chimique. L'objectif principal de cet ensemble d'applications est donc de simplifier, d'automatiser et d'optimiser la gestion d'un laboratoire. Parmi ces applications, on distingue une solution de supervision d'instruments analytiques.

1.3 Objectifs du stage

L'objectif du stage est de concevoir et d'implanter l'architecture d'une solution de supervision d'instruments de laboratoire d'analyses chimiques. La mise en production de l'application n'est pas la finalité du stage. Celui-ci vise plutôt à fournir une preuve de concept ou « proof of concept » (POC) qui démontre la faisabilité du logiciel. Cette application fait partie intégrante de la suite logicielle GreenLAB et se nomme LabMonitor. Mon travail s'est déroulé en trois étapes :

- J'ai élaboré la conception de l'architecture logicielle. Dans un premier temps, j'ai réalisé un état de l'art de la supervision des systèmes d'information. J'ai ensuite effectué une comparaison d'un échantillon d'outils de supervision afin de dégager une architecture globale de ce type d'outils. Enfin, j'ai étudié les différentes architectures logicielles en couche pour concevoir celle qui sera employée par l'application LabMonitor,

- J'ai défini les technologies utilisées. Grâce à l'architecture déterminée auparavant et aux spécifications du projet LabMonitor, j'ai pu identifier les différents besoins de chaque couche logicielle. Ainsi, j'ai comparé et sélectionné celles qui sont utilisées lors du développement,

- J'ai implanté et déployé l'architecture choisie de l'outil de supervision. Cette tâche a entraîné la création de services web, le développement d'une interface web ainsi que la réalisation d'une application mobile. Ces nouvelles technologies m'étaient jusqu'alors inconnues.

1.4 Plan du mémoire

Ce premier chapitre a donc été consacré à la présentation de l'entreprise Persistent Systems, de son orientation et de ses aspirations.

Le second chapitre décrit en détail le projet GreenLAB et plus particulièrement l'application LabMonitor. Les différents objectifs et contraintes environnementales ou logicielles sont mis en évidence.

Un troisième chapitre présente un état de l'art de la supervision. Il traite des différents desseins de la supervision ainsi que des diverses applications déjà existantes dans plusieurs secteurs d'activités autres que la chimie analytique.

Le quatrième chapitre expose les choix de conception en fonction des contraintes précédentes quant à l'architecture logicielle de l'application. Il est constitué d'une étude préliminaire des différentes architectures et techniques de déploiements possibles et de la proposition finale.

Pour continuer, un cinquième chapitre argumente les différents choix techniques que j'ai adoptés pour le développement de l'application GreenLAB LabMonitor.

Le sixième chapitre retrace les implantations logicielles que j'ai effectuées tout au long du stage. Il décrit la mise en place de la nouvelle architecture de l'application ainsi que la réalisation des deux interfaces disponibles sur poste de travail classique et terminal mobile.

Enfin, un dernier chapitre conclut ce mémoire en présentant une synthèse technique des réalisations puis évoque l'avenir du projet GreenLAB. J'ai ensuite dressé un bilan personnel relatif à cette expérience de stage.

Chapitre 2 La suite logicielle GreenLAB

Dans ce chapitre, on aborde la suite logicielle GreenLAB et plus particulièrement le projet LabMonitor. Les spécifications de ce dernier que nous avons recensées y sont décrites ainsi que les différentes contraintes liées à l'environnement de la chimie analytique.

2.1 Contexte

Les laboratoires d'analyses chimiques sont des environnements complexes. Ils nécessitent l'utilisation de nombreux instruments analytiques et logiciels métier différents. Parmi ces derniers on peut citer :

- Système de gestion de l'information du laboratoire ou « Laboratory Information Management System » (LIMS). Il fournit la traçabilité des échantillons chimiques analysés en laboratoire,

- Logiciel de chromatographie ou « Chromatography Data System » (CDS). C'est l'application centrale du laboratoire en charge d'acquérir, de traiter et d'interpréter les résultats générés par les instruments d'analyses,

- Cahier de laboratoire électronique ou « Electronic Laboratory Notebook » (ELN). Cet outil est utilisé au quotidien pour favoriser le suivi d'expériences chimiques,

- Système de gestion de données scientifiques ou « Scientific Data Management System » (SDMS). Il permet de stocker toutes les données scientifiques issues du laboratoire et propose des fonctionnalités de recherche avancées.

Ces logiciels issus d'éditeurs différents sont utilisés quotidiennement par les nombreux acteurs du laboratoire.

Dans le but de développer une nouvelle ligne de produits dans le monde des laboratoires d'analyses chimiques et donc de s'ouvrir vers de nouveaux marchés, Persistent Systems a décidé d'appliquer l'utilisation d'un outil BPM au domaine de la chimie analytique. Cette approche consiste à modéliser informatiquement les différentes étapes des processus métier du laboratoire. Elle prend également en compte les différentes ressources matérielles, applicatives ou humaines afin de les piloter. La suite logicielle GreenLAB se propose donc de modéliser les processus métier du laboratoire en fournissant une solution universelle de gestion de laboratoire. Elle cible plusieurs objectifs :

- Visualisation globale des étapes des processus métier du laboratoire ainsi que les différentes ressources associées,

- Automatisation du laboratoire : une gestion automatisée du laboratoire permet de diminuer les interventions humaines et donc de limiter les erreurs qui en résulteraient. On peut

mentionner par exemple la génération et l'archivage automatique d'un rapport concernant une analyse dans un SDMS,

- Optimisation du laboratoire : c'est la finalité de tous les environnements de production. Cette optimisation améliore l'utilisation des ressources et donc réduit les coûts de production.

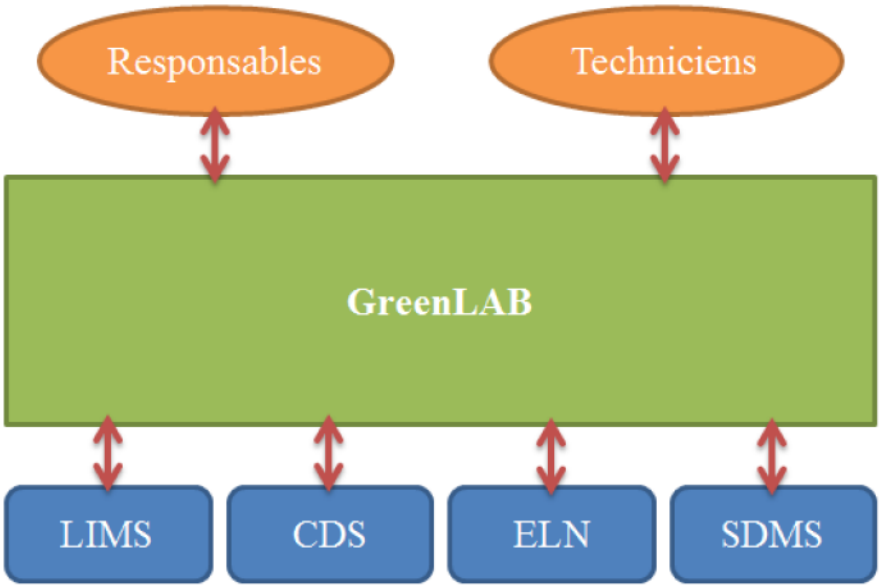


FIGURE 2.1 : ROLE DU PROJET GREENLAB

GreenLAB propose donc un accès unique aux différents utilisateurs du laboratoire pour gérer leur travail quotidien (cf. figure 2.1).

2.2 Présentation

La suite logicielle GreenLAB s'organise en trois parties distinctes :

- Workflow : Cœur de la solution GreenLAB, il modélise les processus métier et automatise certaines étapes de ces derniers,

- Search : de nos jours, la capacité de production de données d'analyses chimiques croît de façon exponentielle. Dans ce contexte, cet outil doit dans un premier temps permettre à l'utilisateur de retrouver des données dans un grand volume de stockage et de façon très rapide, puis deuxièmement offrir la possibilité d'extraire de la connaissance, de la valeur de ces données produites,

- LabMonitor : c'est l'application de supervision qui fournit l'état de santé en temps réel des instruments analytiques du laboratoire.

2.2.1 GreenLAB Workflow

GreenLAB Workflow est une application qui vise à améliorer la productivité d'un laboratoire. Elle automatise entièrement l'analyse chimique d'échantillon. Lorsqu'un échantillon est reçu par un laboratoire, il est enregistré dans le LIMS. Ce dernier génère un ordre d'analyse et Workflow entre en jeu. Il se charge d'exécuter les analyses sur les différents instruments du laboratoire en fonction de la disponibilité de chacun. Dès qu'un instrument est disponible, l'analyse est réalisée. L'application Workflow gère également la collaboration entre les différents acteurs du laboratoire pour valider les rapports générés à la fin de l'analyse. A l'issu de ce processus de validation, il les stocke dans le SDMS. Ce projet est basé sur une solution BPM.

2.2.2 GreenLAB Search

GreenLAB Search est une application qui dans un premier temps indexe un nombre important de données scientifiques. Puis dans un second temps, elle autorise l'utilisateur à effectuer des recherches sur ces données via une interface complète et intuitive.

La particularité de ce projet est de proposer un système de recherche identique aux moteurs de recherches Internet (exemple : Google). Il suggère deux types de recherches : la recherche simple grâce à un unique champ de saisie et la recherche avancée avec la création de requêtes spécifiques. La réponse graphique des résultats permet à l'utilisateur de converger très facilement vers les données cibles, d'extraire également de l'information et d'agréger de grandes quantités de données pour produire de la connaissance.

2.2.3 GreenLAB LabMonitor

GreenLAB LabMonitor est un programme qui informe l'utilisateur du laboratoire en temps réel de l'état de fonctionnement des instruments. Accessible depuis un poste de travail classique ou un terminal mobile, cette application permet également aux responsables du laboratoire d'optimiser la productivité de son laboratoire. Elle affiche en effet sous forme graphique des données statistiques sur l'utilisation des instruments. Ils peuvent ainsi identifier :

- Les instruments les plus souvent en panne,
- Les instruments les moins utilisés,
- Les instruments sur lesquels il y a trop d'attente.

Le recoupement de ces informations favorise les décisions nécessaires à l'optimisation du laboratoire.

Bien que les trois applications précitées soient complémentaires dans la gestion du laboratoire d'analyses, GreenLAB LabMonitor est étudié pour pouvoir fonctionner indépendamment.

2.3 Spécifications du projet LabMonitor

L'étape préliminaire indispensable avant de démarrer le développement d'une application est une étude précise des besoins des utilisateurs finaux du système ainsi que des contraintes de l'environnement ciblé. Dans le cas présent, ce dernier est le laboratoire d'analyses chimiques.

2.3.1 Eléments supervisés

Instruments analytiques

Un laboratoire est composé de plusieurs types d'instruments d'analyses chimiques (voir figure 2.2). On peut nommer par exemple :

- Les chromatographes en phase gazeuse (GC),
- Les chromatographes en phase liquide (LC),
- Les spectromètres de masse (MS).

Chromatographe GC *Spectromètre GC-MS* *Chromatographe LC*

FIGURE 2.2 : INSTRUMENTS SUPERVISES

Tous ces instruments sont issus de constructeurs différents : Agilent Technologies, Thermo Fisher Scientific, Waters, … Ils utilisent des protocoles de communication spécifiques qui n'autorisent pas plusieurs communications en parallèle. En effet lorsque le logiciel métier communique avec l'instrument, il rend impossible la récupération d'information comme son statut par une autre application. L'objectif principal de ce projet étant de superviser les différents instruments du laboratoire, il faut donc trouver une alternative.

Comme évoqués précédemment, ces instruments sont pilotés par des logiciels métier. La supervision pourra donc s'effectuer en interagissant directement avec ces logiciels qui possèdent

des interfaces de programmation ou « Application Interface Programming » (API) comme le montre la figure 2.3.

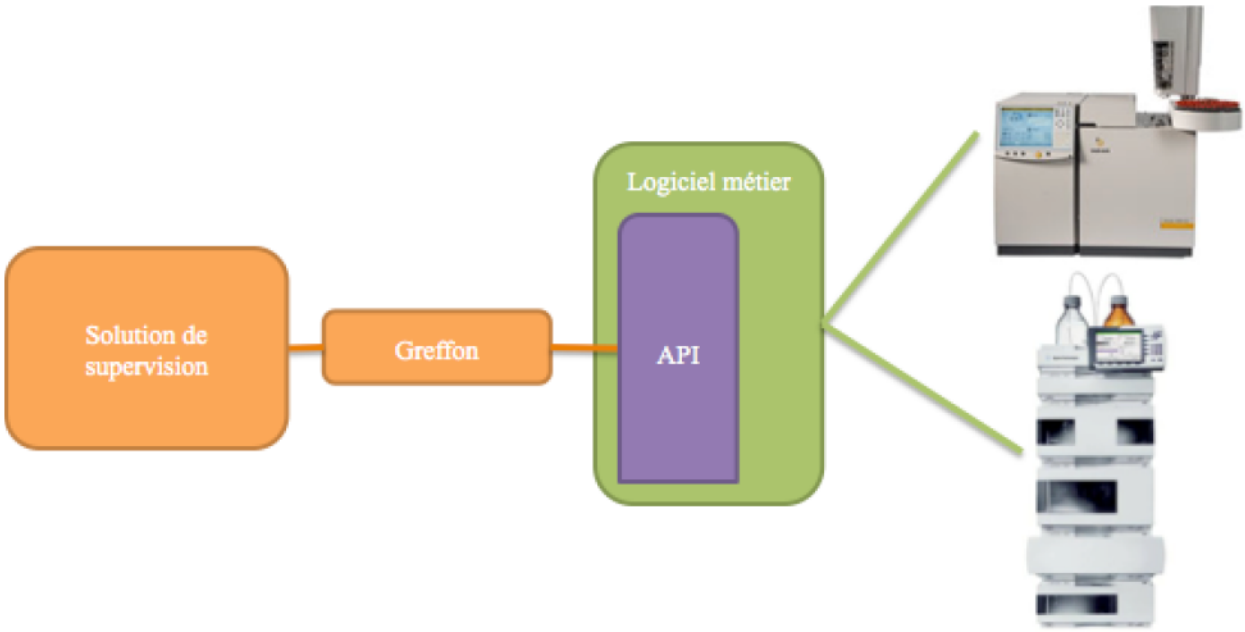


FIGURE 2.3 : SUPERVISION DES INSTRUMENTS ANALYTIQUES VIA LE LOGICIEL METIER

On peut citer entre autres les logiciels d'analyses chimiques suivants :

- Logiciels de chromatographie :
  - « OpenLAB CDS EZChrom Edition » par Agilent Technologies,
  - « Galaxie CDS » par Agilent Technologies,
  - « Chromeleon » par Thermo Fisher Scientific,
  - « Empower » par Waters.

- Logiciels de spectrométrie :
  - « MassHunter Workstation » par Agilent Technologies,
  - « Xcalibur » par Thermo Fisher Scientific,
  - « MassLynx » par Waters.

Chaque logiciel nécessitera donc un développement spécifique pour autoriser la supervision des instruments d'analyses chimiques.

2.3.2 Utilisateurs ciblés

Les deux principaux types d'utilisateurs ciblés par ce projet sont les responsables et les techniciens de laboratoire. Tous deux travaillent en collaboration.

Le responsable de laboratoire est le gestionnaire du laboratoire. Il est chargé de la gestion des coûts, des investissements ainsi que des plannings d'analyse. Il doit également identifier les améliorations possibles afin d'optimiser les performances du laboratoire et de réduire ses frais de fonctionnement.

Le technicien de laboratoire est, quant à lui, en charge des différents travaux planifiés par le responsable. Il doit analyser les échantillons, vérifier et contrôler chaque étape du processus d'analyse, valider les différentes analyses auprès de sa hiérarchie et surtout, il doit assurer la maintenance des instruments.

2.3.3 Interface web et interface mobile.

Comme les besoins des utilisateurs ciblés sont différents, il faut adapter les interfaces de l'application à chacun. En effet, le responsable du laboratoire doit pouvoir visualiser les différentes statistiques calculées par l'application afin de pouvoir prendre des décisions. L'interface qui s'adapte le mieux dans ce cas est l'interface web. Elle est consultable depuis un poste de travail classique connecté au réseau. L'écran de ce terminal possède un écran assez large pour afficher plusieurs graphiques et permettre une bonne visibilité de l'ensemble des informations.

Le technicien de laboratoire qui a en outre la charge de la maintenance des instruments doit pouvoir quant à lui être prévenu lorsqu'une panne se produit. Son travail journalier implique des déplacements constants au sein du laboratoire. Il n'est donc pas constamment face à un seul et même poste de travail. Une interface mobile est particulièrement adaptée à son rôle et plus fonctionnelle. Elle lui permet d'être alerté en temps réel et donc d'intervenir plus rapidement. Le gain de temps peut être considérable en optimisant ainsi le temps de travail.

La mobilité est un point important de ce projet. En effet, ce type de technologie présente un grand potentiel, mais n'est pas pleinement exploité dans le domaine de l'analyse chimique.

2.3.4 Contraintes liées à l'environnement

Hétérogénéité des laboratoires

De même que les utilisateurs, les laboratoires sont hétérogènes. En effet, ils sont tous différents les uns des autres, de par leur taille dans un premier temps, puis de par leur organisation. En effet, une entreprise peut posséder plusieurs laboratoires localisés dans différentes villes voire même différents pays. L'application LabMonitor a pour finalité d'être utilisée dans tous ces environnements. Il doit donc pouvoir être déployé et utilisé localement comme dans le web.

Sécurité du système d'information

Dans la majorité des entreprises, le système d'information est réglementé. Les communications entre superviseur et supervisés peuvent être bloquées par les pare-feu, les antivirus, les serveurs mandataires, l'existence de sous-réseau ou tout simplement par la politique de sécurité mise en place au sein de l'entreprise. Il est donc nécessaire que le ou les protocoles de communication utilisés impactent au minimum le système d'information supervisé. On peut citer par exemple la

réglementation 21 CFR Part 11 émise par la « Food and Drug Administration » (FDA) qui impose des contraintes de sécurité aux laboratoires pharmaceutiques [fda13].

Sécurité des données

Dans le secteur d'activité de l'industrie chimique, la sécurité et la confidentialité sont de rigueur. L'accès aux données doit être particulièrement sécurisé. En effet, même si chaque donnée collectée individuellement n'a pas réellement un caractère confidentiel, l'ensemble de ces données et les statistiques qui en découlent (indicateurs de la performance du laboratoire) possèdent, quant à elles, un caractère tout à fait confidentiel.

2.3.5 Contraintes générales

Au-delà des spécifications et des astreintes associées à l'environnement du laboratoire de chimie analytique, des contraintes générales applicables au logiciel doivent être respectées pour assurer sa fonction. L'application LabMonitor doit être résistante aux pannes pour garantir une qualité de service aux utilisateurs. Elle doit être performante pour fournir des temps de réponse rapides et être extensible pour résister à la charge en cas d'utilisation intensive.

De même, la solution de supervision doit résister et s'adapter rapidement aux changements de l'environnement. Pour rester compétitif, le logiciel doit nécessiter peu de développement pour s'adapter aux nouvelles exigences du laboratoire d'analyses, que ce soit un nouvel instrument ou un nouveau terminal mobile.

2.3.6 Bilan

Le projet GreenLAB LabMonitor est spécifiquement destiné à l'univers de la chimie analytique. Cet environnement est particulier et implique donc de nombreuses contraintes comme l'hétérogénéité des éléments supervisés et de ses utilisateurs. De plus, des exigences de qualité logicielle s'appliquent également à ce projet. En effet, dans le but de satisfaire les besoins des futurs utilisateurs de l'application LabMonitor, ce programme se doit d'être fiable, performant, sécurisé et évolutif.

La finalité du projet GreenLAB LabMonitor est de proposer un outil de supervision parfaitement adapté au domaine de la chimie analytique. De nombreuses solutions dans le domaine de la supervision informatique existent déjà sur le marché et offrent actuellement de nombreuses possibilités. Grâce à l'épreuve TEST réalisée en 2012, il a été démontré qu'un outil existant pouvait tout à fait s'adapter à ce type d'environnement en élaborant quelques modifications. Pour son fonctionnement, la supervision des instruments analytiques via le CDS nécessite la création de greffons. Le développement de nouvelles interfaces est également indispensable afin de s'adapter aux exigences des utilisateurs du laboratoire d'analyses chimiques.

Chapitre 3 La supervision

Ce chapitre, scindé en quatre parties, dresse un état de l'art de la supervision. La première partie définit le terme supervision en le différenciant du terme surveillance qui lui est souvent associé et présente ses objectifs. La deuxième s'intéresse aux différents types de supervision existants. La troisième compare ces différents outils de supervision afin de dégager une architecture globale qui sera énoncée dans la quatrième partie.

3.1 Présentation

3.1.1 Définition

Dans le dictionnaire français, le mot supervision signifie : surveillance et mesure d'une activité. Mais dans le monde de l'informatique, on fait la distinction entre le mot surveillance (plus souvent appelé « monitoring ») et le mot supervision.

La surveillance recueille en permanence tous les signaux issus du système à contrôler, elle reconstitue l'état réel du système et fait toutes les corrélations locales nécessaires pour produire les données qui seront utilisées plus tard par la supervision. Elle est donc limitée aux activités de collecte, d'historisation d'informations et de corrélation sans agir réellement. On peut donc définir son rôle comme passif par rapport au système à superviser [laa13].

La supervision, contrairement à la surveillance, possède un rôle actif. Elle est capable de prendre des décisions vis-à-vis des données collectées par les différentes surveillances mises en place. En temps normal, son rôle se limite à ordonnancer et optimiser les différentes commandes de surveillance. Mais en cas d'erreur, c'est elle qui va prendre toutes les décisions nécessaires pour un retour à la normale de la situation [laa13].

Le tableau 3.1, ci-dessous, récapitule les différences entre la surveillance et la supervision.

Surveillance (« Monitoring »)	Supervision
Local, proximité, courte portée	Regroupement, global, longue portée
Précision	Concentration, concaténation, consolidation
Temps réel	Temps différé
Orienté diagnostic	Orienté présentation, pilotage

TABLEAU 3.1 : SYNTHESE DE DIFFERENCES ENTRE SUPERVISION ET SURVEILLANCE

Source : [Maz12]

La supervision centralise donc la surveillance locale des systèmes. Elle est capable d'effectuer une surveillance ciblée à distance. Elle collecte des informations sur l'état d'un système, les analyse et les organise de manière à rendre visibles aux personnes compétentes l'infrastructure et les entités qui y sont reliées.

3.1.2 Objectifs

Si les systèmes d'information sont tous différents par leur nature, leur taille ou leur criticité, ils ont pourtant un point commun, celui d'être victime d'incidents à un moment ou à un autre. Ils peuvent être parfaitement configurés, mis à jour régulièrement et gérés par les meilleurs administrateurs, la loi de Murphy est immuable : « Si quelque chose peut mal tourner alors cette chose finira par mal tourner » [mur13]. La supervision est là pour détecter ces incidents, alerter les personnes concernées, leur permettre d'identifier le problème et de le réparer pour rendre à nouveau disponible le système d'information. C'est ce que nous allons préciser ci-après.

Prévision des problèmes

Certaines pannes sont précédées de signaux avant-coureurs. Repérer ces derniers est un atout majeur des outils de supervision. Cela peut permettre de régler le problème avant même qu'il ne se produise. Être proactif face aux pannes, voire même automatiser certains comportements préventifs est une demande grandissante pour tous les systèmes d'information. Les outils de supervision permettent ainsi de gagner un temps précieux afin de mieux pouvoir se concentrer sur des actions à véritable valeur ajoutée [Gab09].

Alertes en temps réel

Un système d'information est au service des utilisateurs. Lorsque ces derniers rencontrent des difficultés, ils ne manquent pas de le signaler aux administrateurs. Mais quand cela survient, les utilisateurs sont souvent incapables de retranscrire précisément l'anomalie et peuvent faire perdre du temps aux administrateurs lors de la résolution de celle-ci [Gab09].

La solution de supervision intervient justement pour éviter ce genre de situation. Elle prévient en temps réel l'administrateur d'un état anormal du système d'information. Il dispose alors d'une source d'information complète et pertinente et peut donc s'atteler directement à la résolution du problème. En effet, plus les erreurs sont détectées tôt, plus les interventions sont courtes. La crédibilité du service informatique ou de l'entreprise n'en est donc que meilleure [Gab09].

Identification de la source du problème

Les entités d'un système d'information sont de plus en plus liées entre elles, une simple erreur sur l'une d'elles peut déclencher une multitude d'autres problèmes et paralyser tout un pan de l'architecture réseau de l'entreprise. C'est dans ce genre de situation que la rapidité de détection et de correction d'erreurs s'avère indispensable [Gab09].

L'historisation des alertes détectées par la solution de supervision donne le moyen à l'administrateur de faire le lien entre les différents évènements survenus et donc de remonter facilement à l'origine de la panne [Gab09].

Amélioration de la disponibilité du système

La disponibilité d'un système est l'enjeu le plus important de la supervision. Elle se calcule de la façon libellée dans la figure 3.1.

Disponibilité = (MTBF / (MTBF + MTTR)) X 100

FIGURE 3.1 : FORMULE DE CALCUL DE LA DISPONIBILITE D'UN SYSTEME D'INFORMATION

Pour estimer cette disponibilité, il faut mesurer les deux éléments suivants :

- Moyenne des Temps de Bon Fonctionnement (MTBF) qui est la durée moyenne d'exécution de l'application avant une défaillance [msd13a],
- Moyenne des Temps pour la Tâche de Réparation (MTTR) qui est la durée moyenne nécessaire au service de réparation et de restauration pour résoudre une défaillance [msd13a].

Le tableau 3.2 détaille les différents taux de disponibilité ainsi que leur temps d'indisponibilité respectif. Des taux de 99 % sont facilement atteignables dans la plupart des systèmes d'information, mais pour ensuite réduire l'indisponibilité, certains choix technologiques sont cruciaux [msd13a].

Disponibilité en %	Indisponibilité par année
90 %	36,5 jours
95 %	18,25 jours
99 %	3,65 jours
99,9 %	8,76 heures
99,99 %	52,56 minutes
99,999 %	5,26 minutes
99,9999 %	31,5 secondes

TABLEAU 3.2 : TAUX DE DISPONIBILITE

Source : [mon13]

La supervision identifie les différents niveaux de criticité et met en avant ce qui est essentiel au rétablissement du bon fonctionnement du système d'information. Lors d'une panne, toutes les parties de son infrastructure ne sont pas logées à la même enseigne. Les administrateurs se doivent

de traiter en priorité les parties critiques, car le maintien en production de l'entreprise est leur objectif principal [Gab09].

Mais la supervision ne se limite pas à de simples détections de pannes, elle favorise l'amélioration continue du système. Toutes les mesures collectées mettent en évidence des parties de l'infrastructure plus sensibles que d'autres qui devront être optimisées pour augmenter la disponibilité du système d'information. Il existe différents types de supervision pour permettre de surveiller toutes les entités de l'infrastructure de l'entreprise.

3.2 Types de supervision existants

Les entités à superviser dans un système d'information sont hétérogènes. Il peut s'agir aussi bien d'un équipement réseau, d'un terminal, d'un instrument industriel ou encore même d'une application. A chaque type d'entité est associé un type de supervision dont voici les particularités.

3.2.1 Supervision réseau et matérielle

L'équipement réseau est la base de tout système d'information. Les outils de supervision sont capables de superviser un bon nombre de ces entités :

- Les commutateurs réseau (couramment appelé « switch » en anglais),
- Les routeurs,
- Les serveurs,
- Etc.

Tous ces matériels sont équipés par leur constructeur respectif d'une interface autorisant leur supervision. Ils possèdent par défaut une base d'information pour la gestion du réseau ou « Management Information Base » (MIB) qui stocke des informations matérielles, des paramètres de configuration ou encore des statistiques de performance qui sont directement liés au comportement en cours de l'équipement en question [tec13].

Ces informations peuvent être collectées au travers d'agents SNMP (« Simple Network Management Protocol », protocole simple de gestion de réseau en français) par la plate-forme de supervision [igm13a].

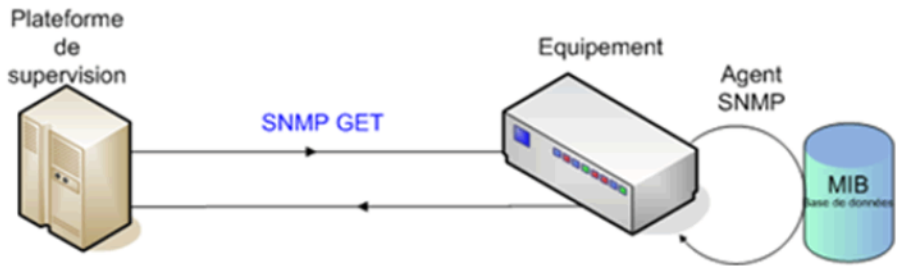


FIGURE 3.2 : SUPERVISION SNMP

Source : [igm13a]

Comme illustré dans la figure 3.2, le superviseur peut collecter les informations de l'équipement réseau en l'interrogeant via une requête SNMP. Elle est alors interprétée par l'agent SNMP qui se charge d'envoyer les données nécessaires stockées dans la base d'information MIB [igm13a]. On peut collecter le type d'information suivant :

- La bande passante,
- La latence,
- Le nombre de pertes de paquets,
- Etc.

Ce sont des indicateurs de performance du réseau. Leurs variations peuvent indiquer des problèmes ou des pannes exclusivement matérielles [igm13b]. Ce type de supervision permet donc de superviser les infrastructures réseau du laboratoire, mais n'autorise pas la supervision directe des instruments analytiques.

Néanmoins le réseau n'est pas le seul niveau de l'infrastructure du système d'information qui peut être supervisé, on peut également interroger les serveurs, les postes de travail utilisateurs ou encore les instruments industriels.

3.2.2 Supervision système

Cette technique de supervision vise à surveiller principalement les terminaux ou les serveurs via le système d'exploitation qu'ils possèdent et plus particulièrement leurs ressources :

- Les disques durs,
- Les processeurs,
- La mémoire vive,
- Etc.

L'accès aux informations des différentes ressources de ces matériels informatiques est possible grâce au standard CIM (« Common Information Model », modèle de données unifié en français) [dmt13]. Les données qui peuvent être collectées sont les suivantes :

- L'espace disponible sur un disque dur,
- La charge du ou des processeurs,
- La mémoire vive utilisée,
- L'état d'un ou plusieurs services,
- Etc.

Ce type de supervision signale bien sûr les pannes, mais identifie également les goulets d'étranglement d'une ou d'un ensemble de machines. On peut ainsi cibler plus judicieusement l'attribution de ressources supplémentaires afin d'augmenter efficacement les performances du système d'information [igm13a].

La supervision système collecte également des données sur des machines industrielles. Mais une interface spécifique est alors nécessaire si ces dernières sont incompatibles avec le standard CIM pour communiquer avec les outils de supervision.

En dehors de l'infrastructure et des systèmes, il existe un autre type d'élément à superviser, il s'agit des applications métier qui tournent sur les serveurs ou postes de travail.

3.2.3 Supervision applicative

La surveillance des applications et des processus métier de l'activité de l'entreprise vérifie le bon fonctionnement d'une application lancée sur un terminal en rassemblant des informations concernant :

- L'application en elle-même,
- Les données issues de l'application.

Grâce à cette surveillance, on peut récolter des données telles que :

- La disponibilité de ou des applications,
- La cohérence des données produites,
- Les performances de l'ensemble de l'activité,
- Etc.

Son but est de superviser les processus métier afin d'identifier les problèmes qui pourraient ralentir l'activité de l'entreprise [igm13a].

3.2.4 Bilan

Les différents types de supervision ne sont pas forcément tous utilisés dans une seule et même entreprise, mais ont le mérite d'être complémentaires comme on peut le voir dans le tableau 3.3.

	Supervision réseau et matérielle	Supervision système	Supervision applicative
Cibles supervisées	Matériel réseau	Serveurs, postes de travail et instruments industriels	Applications métier
Mesures effectuées	Vitesse et qualité du réseau	Etat courant des éléments supervisés	Etat courant de l'activité métier
Problèmes détectés	Problèmes de réseau	Goulets d'étranglement	Problèmes liés à l'activité métier

TABLEAU 3.3 : RECAPITULATIF DES DIFFERENTS TYPES DE SUPERVISION

Source : [igm13a] [igm13b]

Ces trois techniques, si elles sont mises en place dans la solution de supervision utilisée, assurent la surveillance totale de l'architecture d'un système d'information qu'il appartienne à une société dont l'activité est purement orientée informatique ou orienté analyses chimiques [Maz12].

L'objectif principal de l'application LabMonitor est de superviser les instruments analytiques. Comme mentionnée précédemment (cf. 2.3.1), une communication ne peut être directement établie avec l'instrument. Il est donc nécessaire d'interagir à un niveau supérieur : le logiciel métier. Le

type de supervision applicative est donc utilisé pour la supervision des instruments analytiques avec l'application LabMonitor.

3.3 Comparaison des outils de supervision existants

Le nombre de logiciels de supervision sur le marché est très important, ce qui est au premier abord un peu déconcertant lorsque l'on cherche un outil de supervision. Mais cette grande diversité a pourtant son avantage. En effet, chacun de ces outils possède ses particularités qui aident à choisir une solution qui correspond pleinement aux besoins de chaque entreprise. Avant toute décision, il est nécessaire d'identifier les critères qui vont nous permettre de faire la sélection.

3.3.1 Critères de comparaison

Un regroupement de bonnes pratiques, nommé ITIL (« Information Technology Infrastructure Library », bibliothèque pour l'infrastructure des technologies de l'information en français) spécifie les grandes lignes afin de choisir son outil de supervision. Il faut en premier lieu recenser les exigences concernant l'outil que l'on veut utiliser, puis leur accorder une priorité [KBS09].

Un outil de supervision est défini par sa capacité à collecter et traiter des données sans perturber ou modifier le comportement du système d'information observé. Le choix de ce logiciel dépend essentiellement de l'environnement que l'on veut superviser. Voici une liste des principaux critères requis qui vont nous permettre de choisir l'outil adapté au domaine de l'analyse chimique :

- Le type de supervision géré : pour répondre à la contrainte d'hétérogénéité du système d'information, l'outil doit être capable de superviser les entités réseau, les terminaux, les données de production et enfin les applications. Les protocoles utilisés pour l'échange d'information entre superviseur et supervisés sont standards comme SNMP ou CIM et sont facilement applicables dans des environnements sécurisés,

- Le développement de greffon (couramment appelé « plug-in ») possible : du fait de la présence d'instruments d'analyses chimiques dans les laboratoires que les logiciels de supervision sont nativement incapables de surveiller, il est nécessaire de pouvoir personnaliser l'outil grâce à de nouveaux modules de collecte de données. Ceux-ci sont capables de contrôler les instruments d'analyses de l'entreprise,

Ces critères nous ont permis de construire un tableau de comparaison des outils de supervision existants.

3.3.2 Tableau de comparaison

Lors de l'épreuve TEST, nous avions sélectionné un panel de solutions de supervision issues de différents horizons. Désormais, nous avons affiné notre recherche pour nous concentrer sur quatre outils potentiellement utilisables en tant que moteur de supervision pour l'application LabMonitor :

- Nagios : le plus populaire et le plus utilisé dans le monde, sa richesse vient de son importante communauté de développeurs qui lui ajoutent sans cesse de nouvelles fonctionnalités via des modules d'extension,

- Zenoss : un outil très complet qui possède également une communauté motivée,

- Zabbix : un logiciel de supervision open source similaire à Nagios,

- OpenNMS : une solution de supervision complète sous licence open source. Elle est développée en Java. Elle peut être redistribuée gratuitement.

Bien que le type de supervision adapté soit orienté applicatif à cause des contraintes de communication avec les instruments analytiques, il est préférable de se diriger vers des outils qui proposent le plus de fonctionnalités. L'application LabMonitor se doit d'être évolutive. En effet, la solution LabMonitor peut éventuellement évoluer vers une supervision globale de l'infrastructure informatique du laboratoire. Le tableau 3.4 sur la page suivante décrit les quatre outils sélectionnés.

			OpenNMS	Nagios	Zabbix	Zenoss
Type de supervision	Réseau et matérielle		✓	✓	✓	✓
	Système	Linux	✓	✓	✓	✓
		Mac OSX	✓	✗	✓	✗
		Windows	✓	✓	✓	✓
	Application		✓	✓	✓	✓
Greffon			✓	✓	✓	✓
Système de stockage utilisé			PostgreSQL	Fichiers MySQL	MySQL PostgreSQL SQLite	MySQL
Langages utilisés lors du développement de l'outil			Java	PHP	PHP / Java	Python / Java
Plate-forme de déploiement			Linux MacOSX Windows	Linux	Linux	Linux

TABLEAU 3.4 : TABLEAU COMPARATIF DES OUTILS DE SUPERVISION EXISTANTS

Source : [ope13] [nag13] [zab13] [zen13]

3.4 Architecture globale dégagée

Au cours de l'épreuve TEST, nous avons été amenés à étudier un échantillon de logiciels de supervision déjà présents sur le marché. Nous avons ainsi dégagé une architecture globale présentée par la figure 3.3.

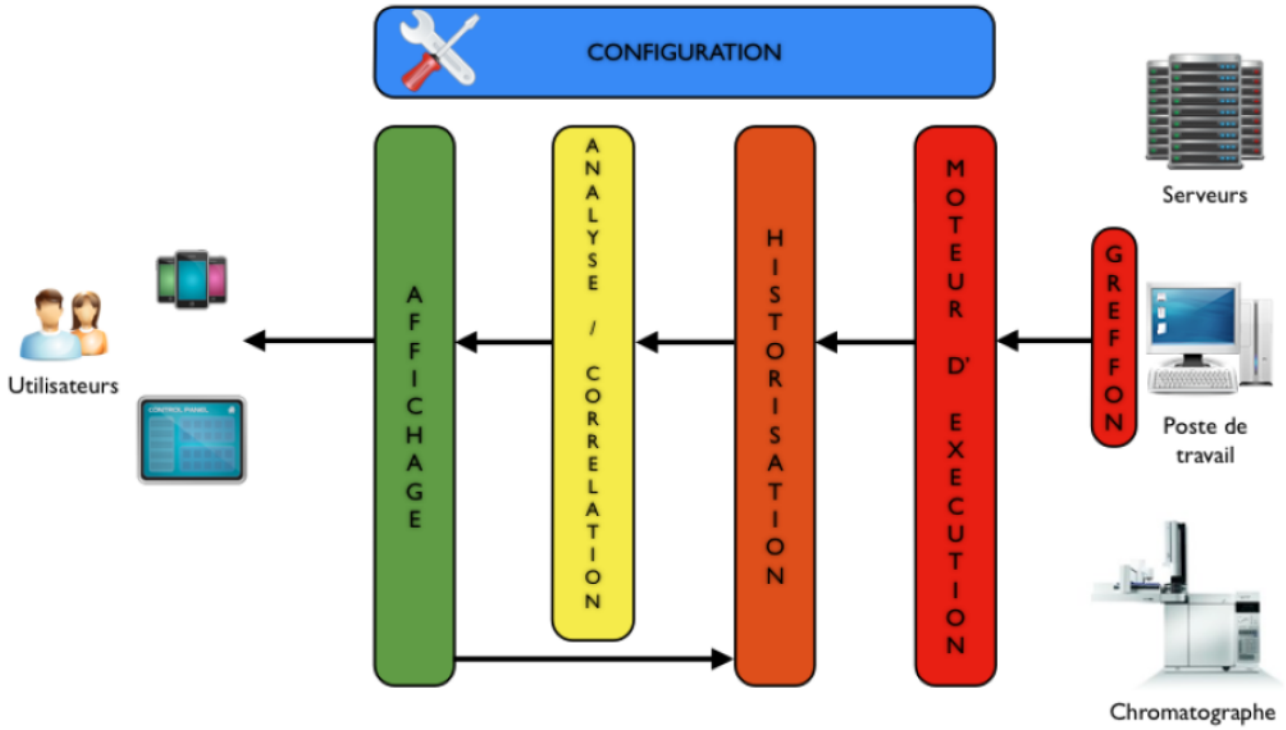


FIGURE 3.3 : ARCHITECTURE GLOBALE DES OUTILS DE SUPERVISION

Source : [Maz12]

En général, l'architecture des outils de supervision se décompose en plusieurs couches logicielles qui vont permettre la collecte, l'exploitation puis l'affichage des données du système d'information :

- Les greffons : ce sont des modules spécialisés dans la collecte de données. Ils envoient ensuite les mesures relevées au moteur d'exécution,
- Le moteur d'exécution : c'est lui qui ordonnance toutes les requêtes de collecte de données ainsi que le stockage des informations,
- La configuration : elle permet d'appliquer des paramètres spécifiques à tous les autres niveaux de l'architecture de l'outil de supervision,
- L'historisation : c'est le système de stockage de l'outil de supervision qui stocke et organise les données collectées en fonction du temps,

- L'analyse et la corrélation : elles permettent d'interpréter les données collectées par les greffons afin de déclencher et d'afficher des alertes aux administrateurs concernés,

- L'affichage : l'utilisateur peut la consulter pour vérifier l'état courant ou les performances de son système d'information.

L'architecture globale ainsi dégagée nous permet de concevoir celle qui sera implantée pour l'application LabMonitor.

Le chapitre suivant traite en détail différents types d'architectures logicielles en couches existantes et dévoile celle choisie pour le projet. Les notions de déploiement matériel sont également abordées.

Chapitre 4 Le choix de l'architecture logicielle

Ce chapitre présente les choix de conception relatifs à l'architecture et au déploiement de l'application LabMonitor. Une première partie définit les différentes architectures client-serveur en couches existantes. La seconde propose une architecture logicielle pour ce projet. Une troisième partie présente la notion de déploiement ainsi que ses objectifs. Et enfin, la quatrième envisage un déploiement possible de l'outil de supervision LabMonitor.

4.1 Architecture logicielle client-serveur

4.1.1 Définition

Dans le domaine de l'analyse chimique, la gestion centralisée des données et des utilisateurs est primordiale. Elle accroît la sécurité des données. Ce type de contrainte impose une architecture client-serveur. Cette dernière est un modèle de fonctionnement logiciel compatible avec la plupart des architectures matérielles. Elle est donc parfaitement adaptée au logiciel GreenLAB LabMonitor.

L'architecture client-serveur, comme son nom l'indique, est basée sur l'utilisation de deux types de logiciels : l'application serveur et l'application cliente qui peuvent s'exécuter sur deux machines différentes et qui communiquent entre elles à travers le réseau. Après la réception d'une requête envoyée par l'application cliente, l'application serveur se contente d'effectuer le traitement demandé et de retourner la réponse au programme client. L'application cliente à l'inverse, provoque le dialogue. Elle envoie une requête au programme serveur et attend la réponse pour ensuite afficher le résultat à l'utilisateur [lar13]. Une application utilisant l'architecture client-serveur se découpe traditionnellement en trois couches logicielles.

Couches logicielles

Une application informatique modélisant l'architecture client-serveur se compose en général de trois niveaux distincts :

- La couche de présentation, également appelée Interface Homme Machine (IHM) qui permet l'interaction de l'utilisateur avec le logiciel. Elle est en charge de la gestion des interactions de l'utilisateur via le clavier, la souris, mais également de la logique de navigation et de la présentation à l'écran des informations [car13] [msd13b].

- La couche métier, également appelée logique applicative, décrit les actions à effectuer par l'application. Elle peut être découpée en sous-parties : les traitements locaux qui regroupent les contrôles au niveau des échanges avec la couche présentation et les traitements globaux qui constituent l'application elle-même [car13] [msd13b].

- La couche de données est la partie logicielle en charge de l'accès et du stockage des informations stockées par l'application (fichiers, bases de données…) [car13] [msd13b].

Cette logique de couche a pour fonction :

- L'évolutivité : le découpage favorise le faible couplage. Ce qui garantit une indépendance et donc une meilleure évolutivité de chaque couche. Lors de la mise à jour d'une couche, les autres sont alors peu ou pas impactées.

- La sécurité : les données sont manipulées uniquement par la couche métier ce qui les rend inaccessibles aux utilisateurs.

- L'interopérabilité : la standardisation des interfaces entre les couches et l'utilisation de protocoles d'échange standard permettent d'utiliser des technologies (langages de programmation) et des supports (systèmes d'exploitation, matériels…) différents au sein d'une même application logicielle.

- La réutilisabilité : l'évolution constante des besoins du client impose une rétro compatibilité avec les applications antérieures. Ce type d'architecture, grâce aux séparations logiques et aux protocoles standards favorise la réutilisation des composants.

- La maîtrise de la montée en charge : les couches logicielles désassocient totalement le logiciel du matériel. On peut donc aisément augmenter la puissance de traitement matériellement sans modifier l'application.

Il existe divers modèles d'architecture logicielle client-serveur. Les principales différences sont liées à l'assignation des couches citées précédemment à l'application cliente ou serveur.

Il est important de bien définir le modèle d'architecture logicielle en fonction de l'environnement et du type d'application que l'on souhaite développer. Nous allons étudier maintenant les différents modèles d'architectures logicielles client-serveur ainsi que leurs avantages et inconvénients respectifs puis conclure sur l'architecture retenue pour l'application GreenLAB LabMonitor.

4.1.2 Architecture logicielle 2 tiers

Définition

L'architecture logicielle 2 tiers, également appelée architecture logicielle client-serveur de première génération, est constituée de la façon suivante : une application cliente est installée sur chaque poste client. Elle possède une interface très riche et se contente de déléguer uniquement la gestion des données à l'application serveur. Les couches de présentation et métier sont donc localisées au sein de l'application cliente et celle de données sur l'application serveur, comme représentées sur la figure 4.1 [car13].

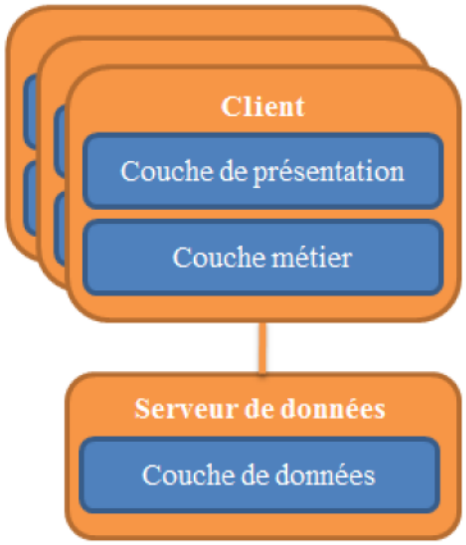


FIGURE 4.1 : ARCHITECTURE LOGICIELLE 2 TIERS

Cette application cliente qui fait la particularité de l'architecture logicielle 2 tiers est appelée client lourd.

Client lourd

Le type de client utilisé dans les architectures logicielles 2 tiers est dit lourd. Une installation est nécessaire sur le poste de travail de l'utilisateur. Cette application cliente est exécutée sur le système d'exploitation du poste de travail. Comme indiqué précédemment, elle est capable d'effectuer de lourds traitements autonomes et communique directement avec la couche de données [gro13] [zdn13].

Avantages

La présence des couches de présentation et métier au sein de l'application cliente dans cette architecture demande une interface utilisateur très riche en fonctionnalités. Les traitements étant effectués sur le client lourd, l'application cliente peut fonctionner de manière autonome en cas de coupure réseau. Elle génère également une importante appropriation des applications par l'utilisateur.

Inconvénients

A l'opposé, par la présence des couches de présentation et métier au sein de l'application client, cette architecture entraîne un inconvénient majeur. L'ensemble des traitements étant effectué sur le poste hébergeant l'application cliente, il se retrouve fortement sollicité et on ne peut malheureusement pas soulager sa charge de travail. Lorsque le logiciel client-serveur évolue, l'application cliente devient de plus en plus complexe et tous les postes clients doivent être mis à jour. Les conversations échangées entre les deux applications sont assez conséquentes, il faut alors une large bande passante pour supporter l'ensemble du trafic des données.

Bilan

Les limites de l'architecture 2 tiers viennent en grande majorité de la nature de l'application cliente utilisée. Elle est très complexe et n'utilise pas toujours des protocoles standards. Une relation étroite existe donc entre le client lourd et le programme serveur, ce qui complexifie les évolutions et les changements d'architecture matérielle. Ce type d'architecture est donc souvent cantonné à un réseau local au sein d'une entreprise [car13].

Ayant déjà eu l'occasion de travailler sur ce type d'architecture logicielle lors d'expériences professionnelles précédentes, nous avons pu nous rendre compte par nous-mêmes des nombreux avantages, mais également des inconvénients non négligeables lors de son utilisation en environnement de production. La volonté de changer de type d'architecture logicielle pour les futurs développements est très forte. Il faut donc s'orienter vers un autre modèle. Cette architecture logicielle ne sera donc pas retenue.

La solution aux problèmes rencontrés avec les architectures 2 tiers réside donc dans l'application du principe suivant : la séparation physique de la couche présentation et de la couche métier. Ce sont les architectures distribuées. L'architecture 3 tiers formulée ci-dessous applique ce principe.

4.1.3 Architecture logicielle 3 tiers

Définition

L'architecture logicielle 3 tiers, également appelée client-serveur de deuxième génération ou client-serveur distribué, sépare les différentes couches par le procédé suivant. Désormais le client ne supporte plus l'ensemble des traitements, mais un sous-ensemble prenant en charge par exemple les contrôles de saisie utilisateurs et l'affichage de données. Une application serveur indépendante gère la majorité des traitements requis par l'application cliente dite légère. La couche de données est toujours régie par une application serveur distincte comme instruit sur la figure 4.2 [car13] [smb13].

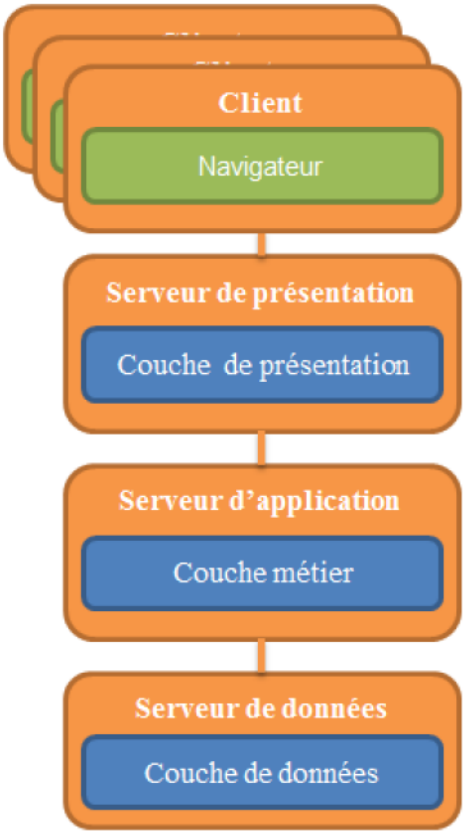


FIGURE 4.2 : ARCHITECTURE LOGICIELLE 3 TIERS

Client léger

A l'opposé de l'architecture logicielle 2 tiers, le type de client de ce modèle est qualifié de léger. Le client léger désigne ici le navigateur Internet dans le cadre d'une application accessible via une interface Web [ccm13] [gro13] [zdn13].

Avantages

Contrairement au modèle précédent, l'architecture logicielle 3 tiers compartimente la couche métier en deux parties, ce qui soulage la charge de traitement côté poste client. L'application cliente se trouve simplifiée et donc moins coûteuse en développement. Le traitement global se déroulant désormais sur un serveur applicatif indépendant, il est plus facile de faire face à des montées en charge en renforçant uniquement ce dernier et non chaque poste client. Un autre avantage de ce type d'architecture est la simplification des mises à jour. En effet, il n'est plus nécessaire de mettre à jour les clients lors d'un changement de version puisqu'aucun module relatif à l'application n'est installé directement sur le poste de travail de l'utilisateur [car13] [smb13].

Inconvénients

Bien que le traitement global soit désormais déporté du côté de l'application serveur, la difficulté demeure, celui-ci sollicite fortement la machine qui l'héberge. Le principal problème de l'architecture 2 tiers côté client a juste été déplacé côté serveur pour l'architecture 3 tiers. Il est donc très difficile de trouver un équilibre de charge entre le poste client et le serveur et on se retrouve confronté aux problématiques de dimensionnement de l'architecture matérielle [car13] [smb13].

Bilan

Les contraintes sont inversées par rapport à celles de l'architecture logicielle 2 tiers. Le client est soulagé, mais l'application serveur est beaucoup plus sollicitée. Du fait de l'environnement très hétérogène que le projet GreenLAB LabMonitor souhaite superviser, il est indispensable de trouver le bon équilibre entre la charge de l'application cliente et l'application serveur.

L'architecture logicielle 3 tiers dont nous venons de définir les applications n'est donc pas non plus retenue pour amorcer le développement. Une troisième architecture logicielle, n tiers, approfondit le découpage de l'application et rétablit l'équilibre entre la charge du client et du serveur.

4.1.4 Architecture logicielle n tiers

Définition

Comme l'architecture logicielle 3 tiers, l'architecture n tiers combine les trois couches : présentation, métier et données. Mais une interface est insérée entre la couche métier et la couche présentation. Cette dernière ne connaît donc plus la couche métier directement, mais uniquement cette couche intermédiaire appelée « couche applicative ». De façon identique, une nouvelle interface intégrant les mécanismes d'accès aux données sépare la couche métier et la couche donnée. Alors que le nombre de couches se multiplie, le modèle de découpage de base reste toujours identique, à savoir trois parties, présentation, métier et données comme le montre la figure 4.3. L'architecture logicielle n tiers met en avant la réutilisabilité de composants entre de multiples services en maximisant le découpage de l'application pour diminuer fortement le couplage [car13] [smb13].

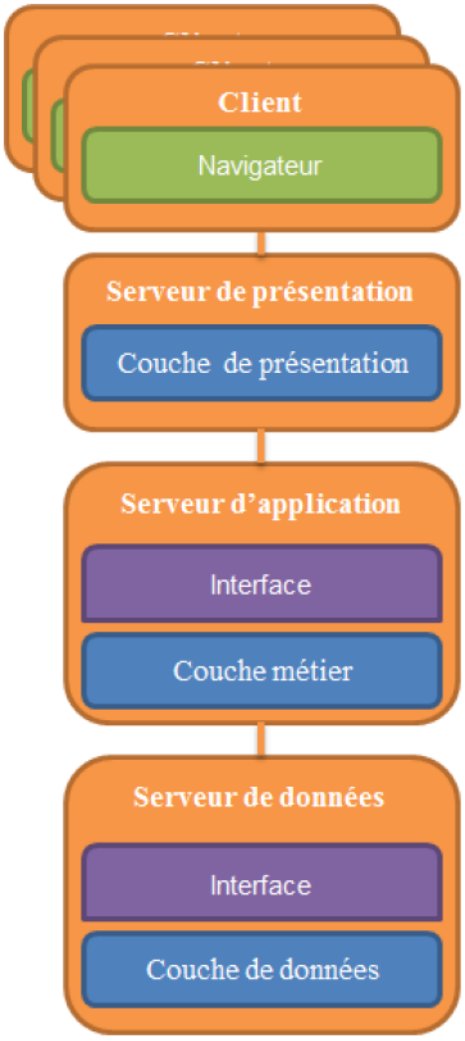


FIGURE 4.3 : ARCHITECTURE LOGICIELLE N TIERS

La principale différence avec l'architecture logicielle 3 tiers réside donc dans le fait que le lien entre chaque couche est défini et limité à des interfaces qui assurent la modularité et l'indépendance technologique de chaque couche.

- La couche applicative est l'interface entre la couche de présentation et la couche métier de l'application. Elle contient les actions représentant les règles métier. On peut citer comme exemple la création ou la suppression d'un utilisateur de l'application, ou encore la lecture de différents éléments [car13] [smb13].

- La couche d'accès aux données quant à elle est responsable de la création des objets métier de façon totalement transparente. Elle est indépendante du mode de stockage qui sera utilisé par l'application (bases de données, fichiers…) [car13] [smb13].

L'architecture logicielle n tiers, tout comme le modèle 3 tiers, met en avant l'utilisation d'un client léger. On constate depuis plusieurs années l'apparition de clients légers riches qui améliorent les interfaces.

Client riche

Le client riche est un client léger enrichi en puissance de traitement pour pouvoir effectuer des traitements locaux et ainsi soulager le serveur. Tout comme le client léger, il donne accès à l'application distante par le biais du navigateur installé sur le poste de travail de l'utilisateur [gro13] [zdn13]. On distingue deux types de clients riches :

- Application Internet riche ou « Rich Internet Application » (RIA). Ce premier modèle permet d'accéder à l'application via le navigateur Internet de l'utilisateur sans aucune installation préalable. En effet, tous les systèmes d'exploitation possédant une interface graphique proposent un navigateur installé par défaut nativement. Ce dernier est donc logiquement nommé client universel. Lorsqu'on parle de RIA, on associe souvent les technologies comme Ajax ou Flash. On peut citer comme exemple de RIA Google Documents [cam13] [zdn13] [cav13].

- Application riche du poste de travail ou « Rich Desktop Application » (RDA). Ce second modèle offre également un accès à l'application distante, mais via un programme client qui se télécharge grâce au navigateur Internet de l'utilisateur lors de la connexion à l'application. Une fois téléchargé, le navigateur n'est plus requis et le programme client s'exécute sur le poste de travail. Le téléchargement automatique de la nouvelle version du programme client permet d'éviter les procédures d'installation et de mise à jour. Mais contrairement au précédent, il exige l'installation d'un moteur d'exécution indispensable au fonctionnement de l'application cliente. Une installation est donc nécessaire sur le poste de travail de l'utilisateur [cav13] [zdn13]. On associe RDA avec les technologies suivantes : Java Web Start, Silverlight, …

Avantages

Tout comme l'architecture 3 tiers, l'architecture logicielle n tiers a le même avantage que le client léger, aucun module concernant l'application n'est installé sur le poste de travail qui ne demande donc qu'une configuration minimale. Le navigateur est utilisé comme client universel et est disponible par défaut sur la majorité des systèmes d'exploitation. Le déploiement est donc facilité, car l'application n'est installée que sur le ou les serveurs physiques hébergeant les couches métier et de données. Les coûts de déploiements sont donc largement diminués [car13] [smb13].

En raison du déploiement simplifié, les mises à jour de l'application le sont donc également. Lors de l'évolution de l'application, seuls les serveurs sont actualisés. C'est un avantage très important surtout dans le cadre du projet GreenLAB LabMonitor où l'évolutivité est le point fort mis en avant.

On constate également une nette amélioration de la sécurité dans ce type d'architecture. En effet, le poste client n'accède jamais directement aux données, seul le serveur d'application connait les informations nécessaires pour se connecter au serveur de données.

Inconvénients

De par la multiplication du découpage de l'application, on est amené à distribuer les différents modules sur plusieurs serveurs, la charge financière matérielle est donc plus importante, mais elle peut être compensée outre mesure avec des services d'informatique dématérialisée, pour ne louer

que l'utilisation réelle de serveurs indépendants sans acheter matériellement les infrastructures [car13] [smb13].

Bilan

L'architecture n tiers diminue grandement le couplage et favorise la modularité. C'est une architecture poussée à son extrême avec de très nombreux composants réutilisables et déployables sur une ou plusieurs machines. Elle offre de très nombreuses possibilités de distributions logicielles et matérielles qui conviennent fortement aux environnements hétérogènes dans lequel le projet LabMonitor peut possiblement être utilisé.

4.1.5 Bilan

Les contraintes de l'environnement du projet GreenLAB LabMonitor sont telles qu'il doit adopter une architecture très modulable, flexible et offrant les meilleures performances. Le tableau 4.1 permet de mettre en évidence les différences et les points forts de chaque architecture.

	Architecture logicielle 2 tiers	Architecture logicielle 3 tiers	Architecture logicielle n tiers
Source de données	Homogène	Hétérogène	Hétérogène
Réutilisabilité	Faible	Forte	Forte
Performance	Faible	Bonne	Excellente
Flexibilité	Limitée	Excellente	Excellente
Administration	Complexe	Moins complexe	Moins complexe
Relève en cas de panne	Faible	Bonne	Excellente
Type de client privilégié	Lourd	Léger/riche	Léger/riche
Utilisation du client universel	Non	Oui	Oui
Installation sur le client	Oui	Non/oui	Non/oui

TABLEAU 4.1 : TABLEAU COMPARATIF DES DIFFERENTS MODELES D'ARCHITECTURE LOGICIELLE CLIENT-SERVEUR

Pour répondre aux contraintes de performance et de robustesse imposées précédemment (cf. 2.3.5), il est essentiel d'utiliser une architecture modulaire au niveau logiciel comme au niveau matériel. L'architecture 2 tiers est une architecture de première génération qui ne facilite pas l'évolutivité de l'application. L'architecture 3 tiers offre quant à elle plus de possibilités, mais c'est bien

l'architecture n tiers qui se démarque par son énorme potentiel d'adaptabilité dans des environnements hétérogènes du fait de sa modularité poussée à l'extrême.

4.2 Architecture logicielle du projet LabMonitor

4.2.1 Proposition

En respectant les contraintes décrites dans le chapitre précédent, nous définissons l'architecture de l'application LabMonitor expliquée par la figure 4.4. Dans cette dernière, on identifie clairement les différentes couches de l'application web : la couche de présentation, la couche métier et la couche de données ainsi que les différentes interfaces entre les couches permettent un couplage très faible entre elles. On distingue également le moteur de supervision qui va se charger d'enregistrer les mesures relevées sur les différents éléments supervisés par l'intermédiaire des greffons.

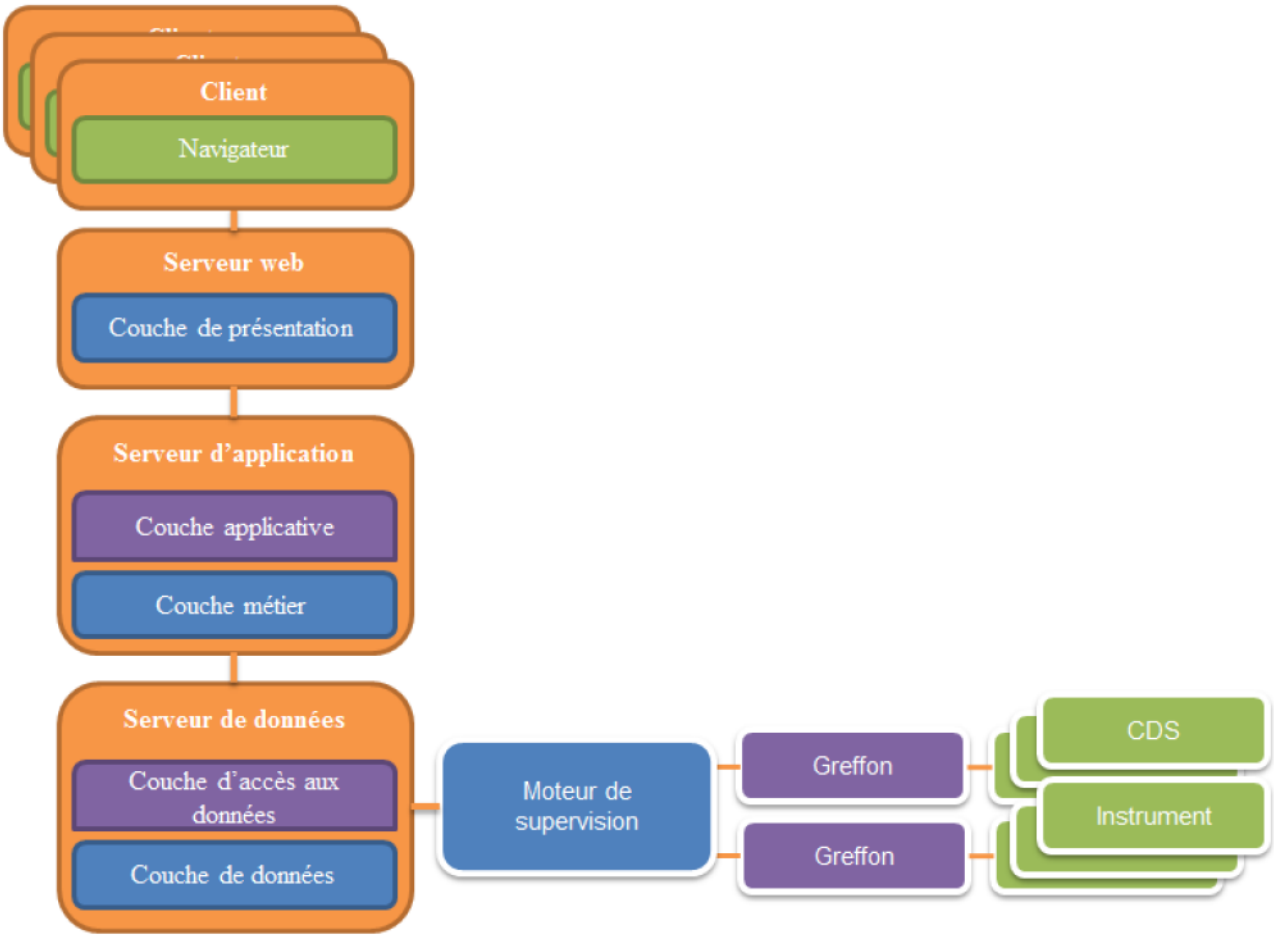


FIGURE 4.4 : PROPOSITION D'ARCHITECTURE LOGICIELLE DU PROJET LABMONITOR

La séparation maximale de chaque couche facilite le développement et l'évolution de l'application GreenLAB LabMonitor.

Couche de présentation

Elle offre à l'utilisateur de l'outil de supervision une vision à la fois globale et détaillée de son parc informatique ainsi que de toutes ses activités métier. Il peut d'un seul coup d'œil décrypter l'état de son système d'information, ainsi que toutes les statistiques issues de l'ensemble des données collectées. La couche de présentation permet donc de :

- Visualiser en temps réel de l'état du système d'information : on peut ainsi consulter instantanément les statistiques de chaque instrument d'un laboratoire afin de pouvoir optimiser la production,
- Intervenir à distance : en cas de problème, on peut déclencher le redémarrage d'un poste de travail sans avoir à se déplacer dans le laboratoire concerné,
- Visualiser les tendances du système d'information : on peut visionner les mesures de disponibilité du système en fonction du temps afin de l'améliorer continuellement,
- Générer des rapports : cela permet de mettre à disposition de futurs clients les performances de son laboratoire.

La vue de l'outil de supervision sera accessible depuis une simple page web. Les modules de la couche de présentation seront donc hébergés par un serveur HTTP. Ce dernier sera responsable de la génération des pages HTML qui seront interprétées par le navigateur du poste de travail client.

Couche métier

Au vu du nombre important de données qui peuvent être collectées lors de la supervision du système d'information d'une entreprise, la corrélation des évènements est un atout très important. En effet, l'ensemble des mesures effectuées sur les éléments supervisés doit être organisé afin de mettre en valeur les informations relatives à l'état de santé du laboratoire à superviser au travers de la couche de présentation. Un traitement doit être effectué en amont de l'affichage, c'est le rôle de la couche métier.

La mise en place de ce type d'outil dépend totalement des caractéristiques propres à chaque système d'information. Les règles doivent donc être conçues sur mesure. Et seuls le temps et l'expérience permettent de tirer parti de toute la puissance de la corrélation [Jan08].

Couche de données

L'historisation d'un outil de supervision est la solution de stockage qui conserve les différentes informations de chaque donnée collectée avec un suivi horaire. Elle est capable d'enregistrer un volume très important de données, car elles sont susceptibles d'être conservées plusieurs mois.

Les données collectées n'ont pas toutes la même valeur en fonction de leur date d'acquisition. La mesure de la charge d'un serveur une heure auparavant peut être utile pour identifier des problèmes détectés sur une activité métier. La même donnée relative au mois précédent a beaucoup moins de

valeur. Les plus anciennes sont toutefois nécessaires pour visualiser une évolution globale du système d'information, mais il n'est pas utile de conserver la même granularité pour celles-ci. Une agrégation des données est donc indispensable sous peine de devoir gérer un volume de données trop conséquent après seulement une année de supervision [Gab09].

Application mobile

Avec l'arrivée de nouveaux moyens de communication comme les terminaux de poche (plus connu sous le nom de « smartphone » en anglais) ainsi que les tablettes, de nouveaux affichages de logiciel de supervision sont en train de voir le jour. Ce nouveau moyen de surveillance fournit de nouvelles latitudes dans une entreprise du domaine de l'analyse chimique, désormais il n'est plus capital de rester face à son poste de travail pour surveiller le système d'information.

Un des principaux objectifs du projet LabMonitor est précisément l'utilisation de ce nouveau moyen de communication dans la vie interne du laboratoire. L'application cliente mobile ne nécessite pas l'utilisation de la couche de présentation de l'application web et communique donc directement avec la couche métier, via l'interface comme on peut le visualiser dans la figure 4.5.

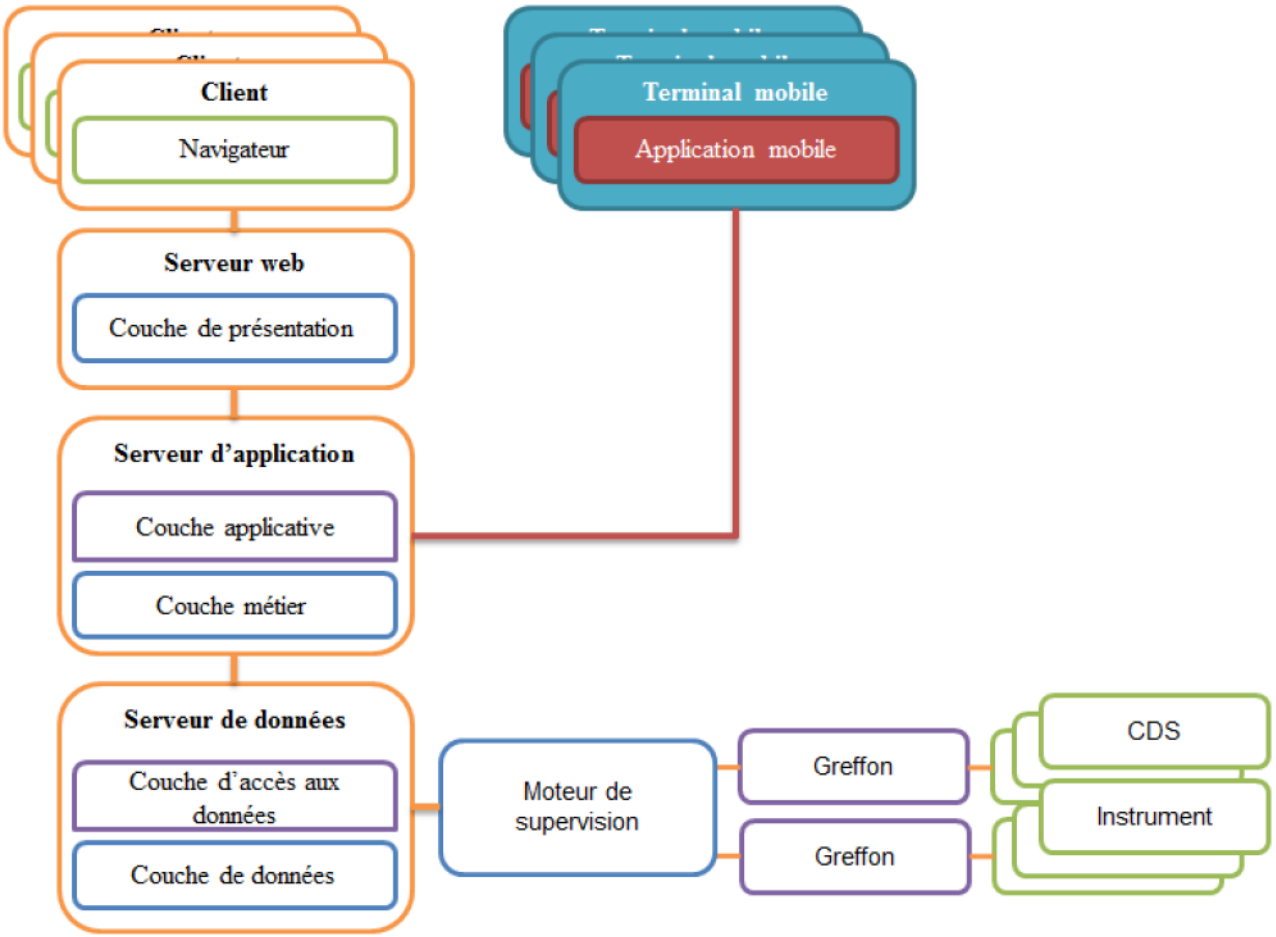


FIGURE 4.5 : INTEGRATION DE L'APPLICATION MOBILE DANS L'ARCHITECTURE LOGICIELLE

La communication entre l'application mobile et la couche métier, sans passer par la couche de présentation sera détaillée et définie dans le prochain chapitre lors du choix des technologies de développement pour l'application mobile du projet.

4.2.2 Bilan

Le choix de conception de l'architecture logicielle est une étape déterminante pour la vie d'un projet de développement. Si elle n'est pas adaptée, l'architecture peut mener une application à sa perte. L'architecture logicielle n tiers offre de nombreuses possibilités pour se plier aux différentes contraintes déterminées par le domaine de l'analyse chimique. Elle répond parfaitement aux exigences d'évolution et d'extensibilité voulues pour l'application LabMonitor. Le découplage de chaque couche logicielle annule également toute contrainte technologique.

L'architecture logicielle est une étape décisive dans la réalisation d'une application informatique. Mais il faut également réfléchir au déploiement de cette application et aux infrastructures matérielles qui sont nécessaires à son bon fonctionnement lors de la mise en production. En effet, il peut influencer le développement d'une partie de l'architecture.

4.3 Notions de déploiement logiciel et matériel

L'architecture logicielle décrite précédemment répond principalement aux contraintes de l'environnement, mais pas forcément aux contraintes de fiabilité et de performance. L'architecture seule ne peut résoudre ces contraintes, il faut optimiser le déploiement logiciel de l'application sur les infrastructures matérielles pour faire face à toutes les astreintes évoquées précédemment.

Chaque serveur ou chaque composant de l'application logicielle possède des limites de capacité de traitement. Bien que des optimisations logicielles les repoussent, il est vite nécessaire de multiplier les instances de chacun pour augmenter la capacité de traitement de l'application et de reprise à la panne. Mais il faut trouver un équilibre entre le travail de chacun. C'est le but de la répartition des charges [tar13].

4.3.1 Objectifs

La répartition des charges répond à plusieurs objectifs :

- Augmenter la capacité de traitement de l'application logicielle en utilisant plusieurs serveurs physiques en parallèle. C'est l'objectif principal de la répartition des charges [smi13].

- Equilibrer la charge pour utiliser de façon optimale les multiples serveurs physiques qui vont être mis en place lors du déploiement de l'application logicielle. En effet, un serveur ne doit pas être surchargé de traitement alors qu'un autre est inactif et disponible [smi13].

- Résister aux pannes, car la multiplication et plus particulièrement la duplication de serveurs permet d'exclure plus facilement un serveur qui fait défaut [smi13].

- Spécialiser les serveurs, on peut également répartir de façon différente les machines physiques, au lieu d'uniquement les dupliquer, on peut les spécialiser individuellement pour n'effectuer qu'une partie du traitement [smi13].

- Faciliter l'exploitation lorsqu'une machine a besoin d'une mise à jour ou d'une maintenance, la répartition des charges permet comme lors d'une panne, d'exclure temporairement un serveur de façon transparente à l'utilisateur de l'application logicielle [smi13].

L'utilisation d'une telle technique répond aux exigences de performance, de robustesse et d'extensibilité citées précédemment.

Un répartiteur de charge peut être logiciel, c'est une application installée sur un serveur qui se charge de répartir la charge entre les différents serveurs de l'infrastructure. Mais il peut également être matériel comme représenté sur la figure 4.6.

FIGURE 4.6 : REPARTITEUR DE CHARGE MATERIEL A10 NETWORKS AX2500

Il existe plusieurs techniques de répartition de charge qui agissent à différents niveaux de l'application.

4.3.2 Niveau DNS

Le « Domain Name System » (DNS) est le service qui permet de traduire le nom d'un domaine, par exemple l'adresse fictive www.greenlab.com en adresse IP qui désigne une machine physique.

La répartition de charge au niveau DNS fonctionne de la façon suivante : lorsque l'utilisateur se connecte au nom de domaine de l'application, son poste de travail envoie une requête au serveur correspondant pour demander son IP, c'est au niveau de ce processus que la répartition des charges intervient en fournissant l'IP du serveur le plus apte au traitement pour un même nom de domaine [smi13] [tar13].

Une des techniques de répartition de charge au niveau DNS est la GeoDNS. Elle consiste à orienter la requête du client vers le serveur responsable du traitement le plus proche géographiquement de l'utilisateur. Ce dernier est localisé grâce à son adresse IP.

Ce type de répartition de charge est utilisé au sein d'infrastructures de tailles très importantes qui sont parfois composées de plusieurs centres localisés sur des continents différents.

4.3.3 Niveau HTTP

La répartition de charge au niveau HTTP analyse chaque requête HTTP reçue du client pour décider ensuite du routage vers un serveur. Lors de l'analyse de la requête, deux éléments sont essentiellement consultés :

- Le contenu de la requête cliente qui peut contenir des informations de sessions utilisateurs. Une répartition par affinité de serveur est alors utilisée. Pour certaines applications, il est obligatoire de conserver en mémoire des informations, par exemple le contenu d'un panier sur un site d'achat en ligne [smi13] [tar13].

- L'identifiant uniforme de ressource plus couramment appelé « Uniform Resource Identifier » (URI) qui représente le traitement demandé par l'utilisateur. On peut ainsi utiliser une répartition avec spécialisation de serveur. En effet, des requêtes d'achats en ligne ou de simple consultation de pages peuvent être réparties sur des serveurs différents [smi13] [tar13].

4.3.4 Niveau TCP/IP

La répartition de charge au niveau « Transmission Control Protocol » (TCP) / « Internet Protocol » (IP) permet contrairement au niveau DNS de masquer plusieurs serveurs derrière une même adresse IP vue de l'extérieur. En effet, sur le réseau interne, chaque serveur possède sa propre adresse IP. Lorsque l'utilisateur entre le nom de domaine, il est automatique traduit vers cette IP globale qui pointe vers l'équipement de répartition des charges et qui va affecter la connexion entrante au serveur le plus approprié [smi13] [tar13].

Les solutions techniques possibles de répartition des charges TCP/IP sont :

- « Round-robin » : le traitement à effectuer est attribué à un serveur de manière cyclique,

- « Least connection » : le serveur sélectionné est celui qui a le moins de traitement en cours,

- « Priority activation » : cela implique l'attribution de requêtes à des serveurs secondaires si les principaux sont surchargés.

4.3.5 Bilan

Comme décrit précédemment, il existe plusieurs solutions quant à la répartition des charges qui équilibrent les échanges de données entre chaque couche de l'application logicielle. Le répartiteur de charge au niveau DNS est adapté à une infrastructure de très forte dimension. La répartition au niveau HTTP impose l'analyse de chaque requête envoyée à l'application, c'est un traitement beaucoup plus lourd que lors de la répartition de charge par TCP/IP. On préférera un répartiteur de charge TCP/IP plus réactif et plus flexible pour le projet LabMonitor afin de répondre aux contraintes de performance, robustesse et extensibilité.

4.4 Déploiement possible du projet LabMonitor

4.4.1 Proposition

Si le choix de l'architecture logicielle est une étape importante dans le développement d'une application, cette dernière ne représente pas l'utilisation en condition réelle dans un environnement ciblé. Lors de la mise en production de l'application, le système subit des montées en charge dues aux traitements demandés en parallèle par les utilisateurs. Pour que le système soit opérationnel, on doit être capable de multiplier les serveurs au niveau des couches pour supporter la charge et répondre au besoin des utilisateurs. Comme le montre la figure 4.7, il est important de savoir que la couche métier sera la plus chargée en termes de traitement et sera capable de se multiplier sur différents serveurs.

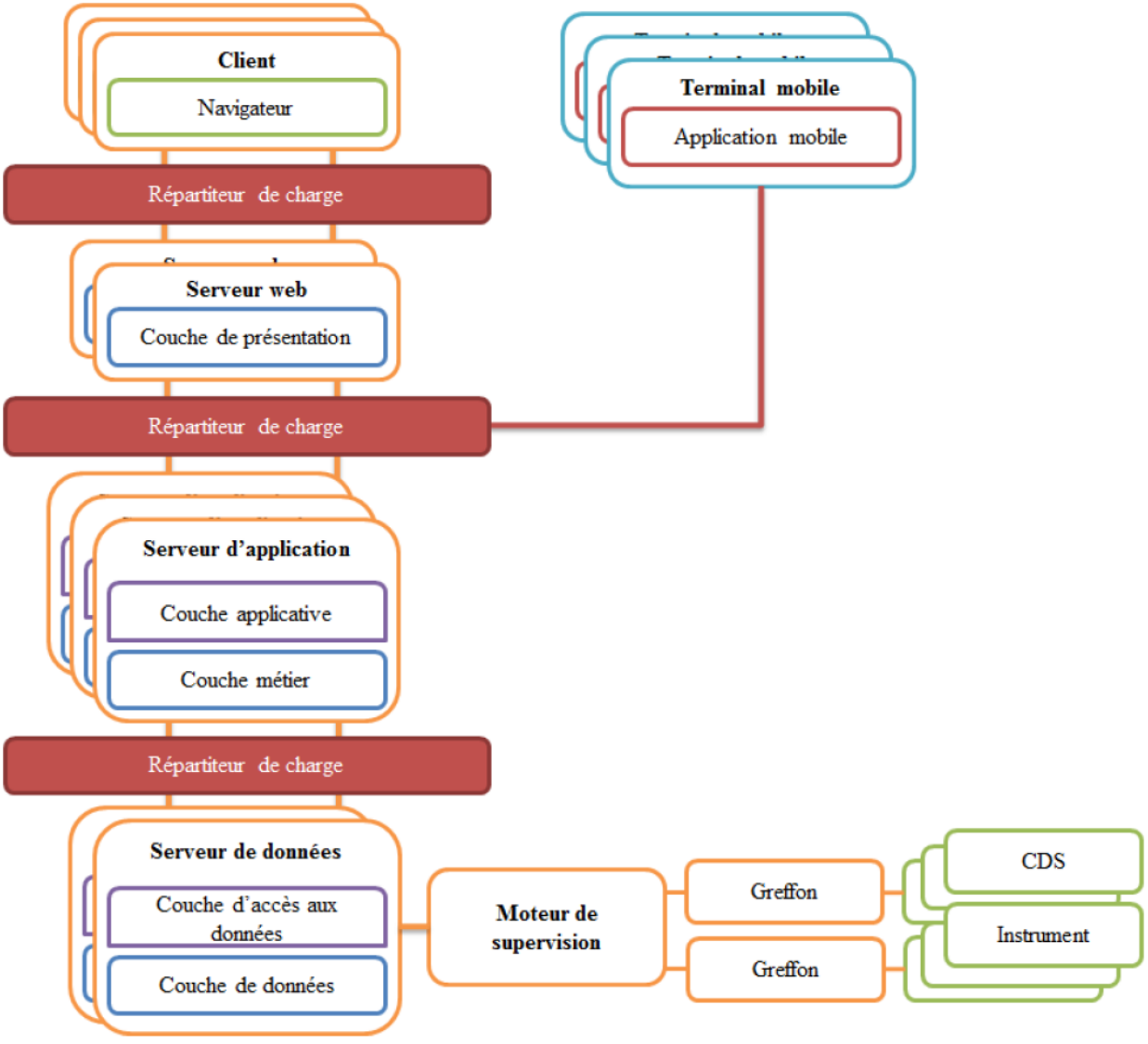


FIGURE 4.7 : DEPLOIEMENT POSSIBLE DE L'APPLICATION LABMONITOR

C'est une première proposition de déploiement qui sera affinée pendant le développement et les tests de performance. Elle est bien sûr fortement liée à l'environnement dans lequel l'application sera installée.

4.4.2 Bilan

Cette étude des notions de déploiement donne une vision claire de l'architecture logicielle en milieu de production. Ainsi, on constate que les contraintes de performances, robustesse et extensibilité peuvent être résolues par des moyens complémentaires au développement logiciel.

Les choix de conception concernant l'architecture logicielle et le déploiement envisagé du projet LabMonitor vont désormais permettre de définir les technologies de développement de l'application LabMonitor. Cette dernière reposant sur l'utilisation d'un outil de supervision existant, la partie moteur de supervision est désormais volontairement exclue des implantations logicielles réalisées. Le développement se concentre uniquement sur l'application web et mobile en charge d'interpréter les mesures de supervision effectuées par l'outil. L'épreuve TEST a mis en avant plusieurs outils de supervision existants appropriés à condition que des greffons soient réalisés pour communiquer avec l'élément à superviser.

Chapitre 5 Les choix technologiques

5.1 Couche de données

5.1.1 MySQL

L'étude des outils de supervision déjà existants fait apparaître quatre outils compatibles avec l'application que nous développons : OpenNMS, Nagios, Zabbix et Zenoss.

Zabbix et Zenoss utilisent par défaut un Système de Gestion de Bases de Données (SGBD) de type « Structured Query Language » (SQL) : MySQL pour stocker les mesures de supervision [zab13] [zen13]. Celles de l'outil OpenNMS sont enregistrées dans une base de données PostgreSQL [ope13]. Le logiciel Nagios, lui, stocke en temps normal sa configuration et ses mesures dans un ensemble de fichiers. Mais le greffon NDOUTILS au logiciel Nagios intervient pour transférer toutes les informations relatives à la configuration et à la supervision dans une base de données de type MySQL également [nag13].

Notre objectif est de réaliser les interfaces de supervision adaptées aux utilisateurs du laboratoire d'analyses chimiques. Le choix de l'outil de supervision existant n'étant pas fixé, nous utilisons donc au cours du développement du projet, un schéma de base de données simplifié.

Le SGBD MySQL est l'un des logiciels de gestion de bases de données les plus utilisés au monde dans le cadre professionnel ou personnel [mys13a] [clu13]. Il est en concurrence avec Oracle et Microsoft SQL Server. Il est compatible avec la plupart des systèmes d'exploitation existants (Windows, Linux, Mac OS X…) et est accessible par la majorité des langages de programmation (C, C#, Java, PHP). MySQL fait partie de l'ensemble de logiciels libres LAMP (Linux, Apache, MySQL, PHP) qui permettent de développer des applications Web.

Une des contraintes concernant la solution de stockage est la montée en charge. En effet, la supervision exige le stockage de mesures régulières, vingt-quatre heures sur vingt-quatre et sept jours sur sept. La technologie MySQL résout ce problème en proposant des outils de répartition de charges.

5.1.2 Répartition des charges

Réplication MySQL

Le principe de réplication est élémentaire, deux serveurs MySQL au minimum sont requis. Le premier tient le rôle de maître et héberge la base de données de l'application. C'est lui qui reçoit toutes les modifications. Le second dit esclave, avec une base de données de structure identique, se connecte régulièrement au serveur maître pour lire les dernières modifications et mettre à jour sa base de données. Un serveur maître peut posséder plusieurs esclaves. Ce sont les serveurs esclaves

qui réceptionnent les nombreuses requêtes de recherche et justifient plus de ressources. Un serveur esclave peut également être le serveur maître d'un autre esclave. On réalise ainsi une chaîne de réplication (cf. figure 5.1) [mys13b].

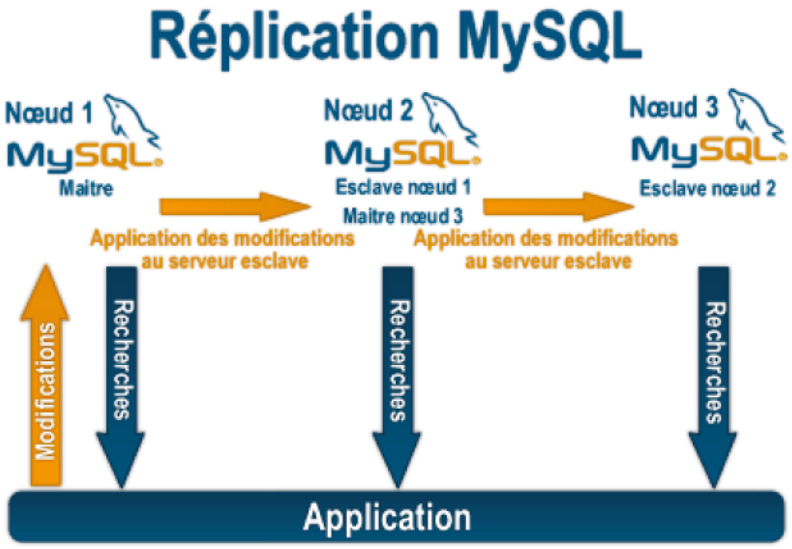


FIGURE 5.1 : REPLICATION MYSQL

Source : [mys13b]

Ce système possède cependant des limites :

- Lorsque le serveur maître cesse de fonctionner, il est possible de changer le rôle d'un esclave en maitre. Mais il est alors nécessaire de réorienter le flux de modification vers le nouveau serveur maître en modifiant un paramètre de l'application.

- Si le serveur esclave tombe en panne, le problème est identique. Il faut également réorienter les requêtes de recherche vers le serveur maître.

Lorsqu'un problème survient sur un ou plusieurs serveurs MySQL, une intervention humaine est nécessaire pour rétablir le fonctionnement de l'application. Ce processus engendre une indisponibilité de l'application. La répartition des charges est donc gérée manuellement par l'administrateur du logiciel [mys13b].

MySQL cluster

La technique de répartition des charges MySQL cluster diffère. Deux serveurs MySQL sont également requis au minimum pour dupliquer le stockage des données. Chaque serveur représente un nœud qui héberge la totalité de la base. Un des serveurs doit également jouer le rôle de répartiteur de charges pour aiguiller les requêtes vers un nœud disponible comme présenté sur la figure 5.2. Les requêtes de recherches sont plus rapides que lors de l'utilisation de la réplication MySQL [mys13c].

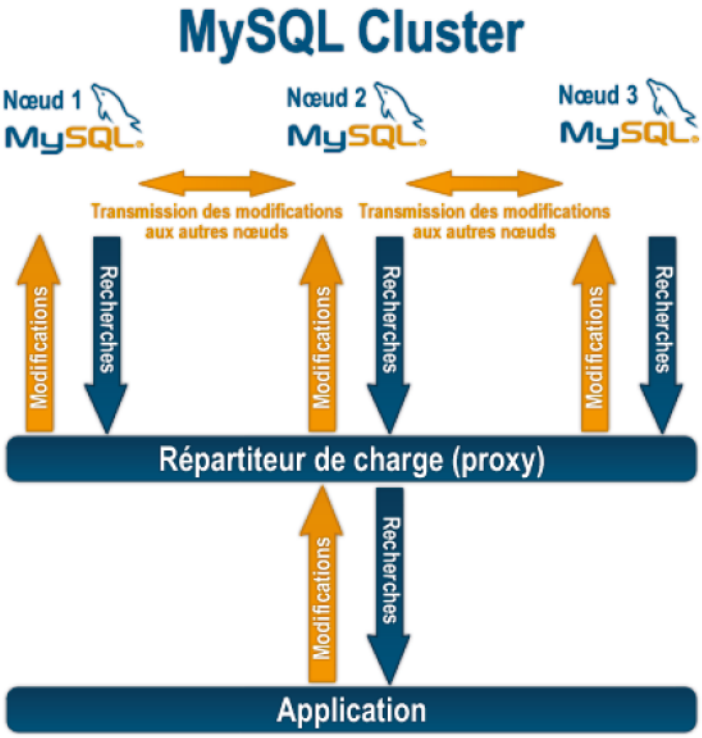


FIGURE 5.2 : CLUSTER MYSQL

Source : [mys13c]

Cette technique procure de nombreux avantages :

- Lors de la panne d'un nœud, les requêtes sont automatiquement redirigées vers les serveurs disponibles. Aucune intervention immédiate n'est nécessaire. Elle peut être temporisée. L'application est toujours disponible, sous réserve qu'elle puisse supporter la charge en l'état actuel.

- L'ajout d'un nouveau serveur est totalement transparent. Il n'est pas non plus nécessaire de stopper l'application, car tout est géré par le répartiteur de charge.

Cette solution est particulièrement intéressante pour une application qui demande une haute disponibilité et une sécurisation optimale des données [mys13c]. C'est le cas de l'outil de supervision qui doit offrir ces caractéristiques. On préfère donc l'utilisation de la technique de

cluster à la réplication MySQL pour une potentielle répartition des charges sur le système de stockage de la solution de supervision.

5.2 Interface d'accès à la couche de données

L'interface d'accès à la couche de données est l'interface entre la couche métier et la couche de données. C'est le point d'entrée de toutes les requêtes vers le système de stockage, qui en l'occurrence se trouve être le SGBD MySQL.

« Java DataBase Connectivity » (JDBC) est l'interface de programmation qui permet à des programmes Java d'accéder à une source de données. Elle est incluse dans l'environnement de développement Java ou « Java Development Kit » (JDK) [ora13].

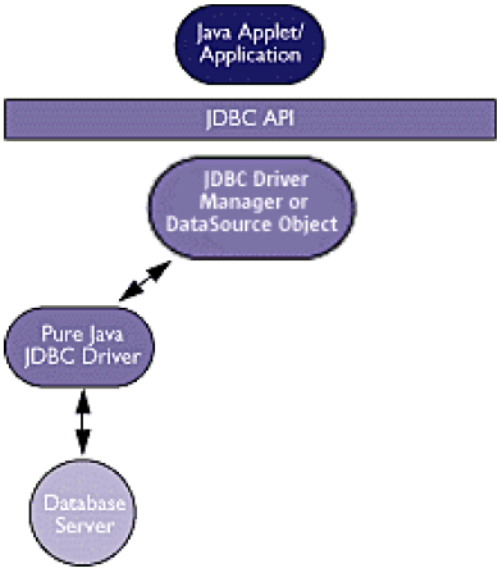


FIGURE 5.3 : PILOTE JDBC

Source : [ora13]

Cette interface de programmation autorise la connexion à une base de données pour exécuter des requêtes grâce à un pilote spécifique au type de base de données utilisée. Ce pilote ou « driver » (cf. figure 5.3) caractéristique à la base de données MySQL découple JDBC de la base de données. Il est utilisé lors du développement de l'interface d'accès aux données du projet GreenLAB LabMonitor pour améliorer le découplage entre la base de données et la couche métier.

5.3 Couche métier

Il existe un très grand nombre de langages de programmation. Java et .Net sont des langages très employés dans le développement logiciel [tio13] [red13]. Lors de développements web, Java est souvent préconisé. Il est indépendant du système d'exploitation, car les programmes développés dans le langage Java demandent un environnement d'exécution pour fonctionner. Cet environnement d'exécution, également appelé « Java Runtime Execution » (JRE) est disponible sur la plupart des systèmes d'exploitation comme Linux, Windows, OSX…

On privilégiera le langage Java couramment utilisé lors de développements web pour la réalisation du projet GreenLAB LabMonitor.

5.4 Interface d'accès à la couche métier

5.4.1 Types de service web existants

Les services web sont des composants logiciels qui rendent possible les échanges entre applications hétérogènes. Ils permettent de mettre à disposition les fonctionnalités d'un logiciel sur Internet ou un intranet. Les échanges entre applications hétérogènes sont permis grâce à l'utilisation de standards et de protocoles web [gar13]. Il existe deux principaux types de service web : WS-* et « Representational State Transfer » (REST).

WS-*

Les services web WS-* sont basés sur les technologies « Simple Object Access Protocol » (SOAP) et « Web Services Description Language » (WSDL). L'approche SOAP s'adapte à deux protocoles de communication : « HyperText Transfer Protocol » (HTTP) et « Simple Mail Transfer Protocol » (SMTP). Elle est indépendante des systèmes d'exploitation client et serveur ainsi que du langage de développement. Ce type de services web est basé sur des échanges utilisant le langage « eXtensible Markup Language » (XML). Les informations échangées respectant les standards SOAP sont riches et génèrent donc un volume conséquent qui transite à travers le réseau ce qui peut ralentir une application. Cette technique autorise donc l'échange d'informations complexes. Cette complexité se répercute sur le développement de ce type de services web [w3c13].

REST

Le type de service web REST est un modèle d'échange basé uniquement sur le protocole HTTP et donc sur l'emploi de ses méthodes : Get, Post, Put et Delete. Toutes les prestations fournies par ces services web sont identifiées comme une ressource. Cette dernière peut être une image, un document, un traitement métier… Toutes ces ressources sont accessibles à travers le réseau grâce aux « Uniform Resource Identifier » (URI) qui définissent à la fois leur nom et leur adresse. Le Web applique ces principes REST définis dans la thèse de Roy Fielding [sea13].

Ce type de service web découple fortement le client et le serveur contrairement à SOAP. Il est, de cette façon, plus simple à utiliser. Les réponses aux requêtes peuvent être formulées avec les langages XML ou « JavaScript Object Notation » (JSON).

5.4.2 Formats d'échange

Deux formats d'échange sont utilisés principalement par les services web : JSON et XML. XML est un langage de balisage qui permet de transmettre des informations structurées. Ce format de données nécessite des bibliothèques spécifiques pour être interprété par le programme qui l'utilise. JSON est au contraire un format de données textuelles. Il permet également de transmettre de l'information structurée. Il a l'avantage d'être beaucoup moins verbeux que XML. Le tableau 5.1 présente un exemple de réponse retourné par un service web dans les deux formats.

JSON	XML

`{`
`  "instrument": {`
`  "name": "Agilent7890_1",`
`  "ip": "10.190.200.123",`
`  "status": "error"`
`  }`
`}`
	`<?xml version="1.0"?>`
	`<instrument>`
	`  <name>Agilent7890_1</name>`
	`  <ip>10.190.200.123</ip>`
	`  <status>error</status>`
	`</instrument>`

TABLEAU 5.1 : TABLEAU COMPARATIF DES FORMATS DE DONNEES JSON ET XML

Comme on le constate, la représentation XML de la réponse est beaucoup plus verbeuse et implique des volumes de données échangés légèrement supérieurs. Différents tests de performance sont disponibles, mais aucun ne révèle lequel prend l'avantage. En effet, les résultats de ces tests dépendent fortement des informations échangées. Dans le cas d'informations simples, JSON prend l'avantage pour la vitesse de transfert et d'interprétation. Mais dès que les informations deviennent complexes, le format XML présente de meilleurs temps d'interprétation par le programme qui réceptionne la réponse [gee13] [eaw13].

5.4.3 Bilan

Le tableau 5.2 récapitule les différences entre les services web de type WS-* et REST.

	WS-*	REST
Utilisation	Complexe	Simple
Format de la requête	XML	URI
Volume de données échangé	Important	Moins important
Technologies associées	SOAP, WSDL	HTTP
Format de données	XML	XML, JSON

TABLEAU 5.2 : TABLEAU COMPARATIF DES TYPES DE SERVICES WEB REST ET SOAP

Nous avons choisi d'implanter des services web de type REST qui paraissent plus simples et plus adaptés au contexte du projet LabMonitor. Bien que de nombreux tests réalisés dans le monde du web tendent à prouver que l'écart n'est significatif qu'à partir d'un important volume de données échangé [gee13] [eaw13], nous avons décidé d'utiliser le format de données JSON pour les échanges réalisés avec nos services web.

5.5 Couche de présentation

La couche de présentation est l'interface graphique de l'application. Elle est affichée via le navigateur du poste de travail. Les langages utilisés pour développer des pages web sont HTML, CSS. Le premier est le format de données qui représente la page web, le second est en charge de la présentation des données. Ces deux langages sont interprétés par le navigateur pour afficher les pages web de l'application LabMonitor côté client. Mais si on se limite à leurs seuls utilisations, les pages web restent statiques. L'interface du logiciel LabMonitor se doit d'être dynamique pour afficher le statut des instruments d'analyses en fonction des mesures stockées dans la base de données.

Le serveur web héberge différentes ressources utiles à la production de ces pages web dynamiques. Il existe plusieurs possibilités pour créer ces dernières.

5.5.1 Langages existants

Les langages de production de pages web les plus employés sont : ASP .Net, Java et PHP [wib13].

Site	Fonctionnel depuis	Système d'exploitation des serveurs	Langages de programmation
Google.com	Novembre 1998	Linux	C, C++, Java, PHP
Facebook.com	Février 2004	Linux	C++, PHP
Youtube.com	Février 2005	Linux	C, Java
Yahoo.com	Août 1995	Linux	C, C++, Java, PHP
Msn.com	Août 1995	Windows	ASP .Net
Wikipedia.org	Janvier 2001	Linux	PHP
Amazon.com	Octobre 1995	Linux	C++, Java, J2EE

TABLEAU 5.3 : LANGAGES DE PROGRAMMATION UTILISES DANS LE MONDE WEB

Parmi les sites web les plus connus cités dans le tableau 5.3, on remarque plus particulièrement la prépondérance des langages Java et PHP. Les langages C ou C++ sont usité plus spécifiquement pour la couche métier. Notre choix s'oriente donc plus particulièrement vers les langages de programmation web Java et PHP.

5.5.2 Cadriciels de développement

Un cadriciel de développement, plus couramment appelé « framework » est un ensemble de composants logiciels structurés destiné à fournir un squelette d'application pour faciliter le travail du développeur. Pour le développement de la couche présentation de l'application LabMonitor, il est nécessaire de se poser la question de son utilisation pour organiser l'architecture interne de la couche.

Il existe entre autres des cadriciels pour encadrer le développement web avec les langages Java et PHP. Ils se nomment respectivement Play2 et Symfony2 [pla13] [sym13].

Le cadriciel destiné au langage de programmation Java a été choisi par les équipes Search et Workflow. Nous avons préféré découvrir le langage de programmation PHP afin d'élargir nos connaissances et confronter les deux langages à la fin des développements.

5.5.3 Bilan

Les cadriciels de développement assurent au développeur un environnement organisé pour développer son application. Ce sont des outils de travail complets qui fournissent de nombreuses fonctionnalités et qui permettent aux équipes de se concentrer sur les besoins du client. Ils présentent en outre de nombreux avantages comme la réutilisabilité des composants ou le gain de temps de développement. Mais ces cadriciels requièrent une bonne maîtrise du langage utilisé. En effet, l'utilisation d'un cadriciel n'implique pas forcément une qualité de réalisation.

Une certaine liberté nous a été accordée pour le développement de l'application LabMonitor. Nous avons donc décidé dans un premier temps de ne pas utiliser de cadriciel afin de nous concentrer sur l'apprentissage du langage PHP.

5.6 Application mobile cliente

5.6.1 Systèmes d'exploitation mobile existants

Bien que le choix de l'utilisation de l'iPhone ait été imposé par le service commercial pour le projet GreenLAB LabMonitor, il est important de connaître l'état du marché actuel. Il existe de nombreux systèmes d'exploitation mobiles dans le monde des terminaux mobiles. Les systèmes d'exploitation les plus utilisés sont :

- iOS développé par Apple Inc.,
- Android développé par Google Inc.,
- BlackBerry OS développé par RIM,
- Windows Phone développé par Microsoft.

Android est un système d'exploitation « open source » orienté mobile basé sur le noyau Linux. Il peut être utilisé sur des terminaux mobiles du type téléphone intelligent ou tablettes. A l'origine créé par une jeune entreprise ou « startup » du même nom, Android a été racheté par Google Inc. en 2005. Ce système d'exploitation libre est proposé gratuitement aux fabricants de terminaux mobiles, ce qui facilite sa progression dans le marché mobile.

Comme le montre la figure 5.4, le système d'exploitation le plus employé dans le monde en août 2012 est Android.

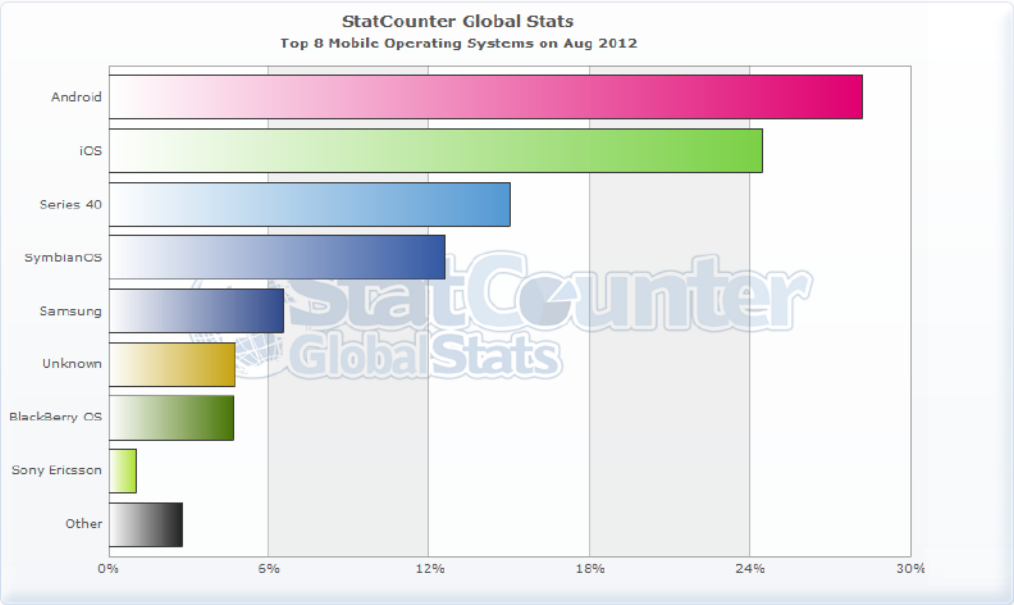


FIGURE 5.4 : REPARTITION DES SYSTEMES D'EXPLOITATION MOBILES EN AOUT 2012 DANS LE MONDE

Source : [sta13]

Le système d'exploitation mobile iOS développé par Apple contrairement à Android est exclusivement réservé aux terminaux Apple suivants :

- iPhone : téléphone intelligent, plus communément appelé « smartphone ». La date de sortie de la première génération d'iPhone correspond à la disponibilité de la première version du système d'exploitation iOS (version 1.0),
- iPod Touch : baladeur numérique à écran tactile, également considéré comme une console de jeux portable. Il est similaire à l'iPhone mais ne dispose pas des fonctionnalités de téléphonie et d'appareils photo numériques,
- iPad : tablette tactile. La tablette tactile est le chaînon manquant entre les téléphones intelligents et les ordinateurs portables. Elle est particulièrement orientée vers les médias tels que les livres, magazines, films, jeux, mais aussi vers la navigation internet,
- Apple TV : appareil numérique qui permet la communication entre un ordinateur et une télévision.

Mais lorsque l'on observe plus particulièrement la répartition sur le continent européen, c'est le système d'exploitation iOS qui se retrouve statistiquement en tête des systèmes d'exploitation mobile comme constaté sur la figure 5.5.

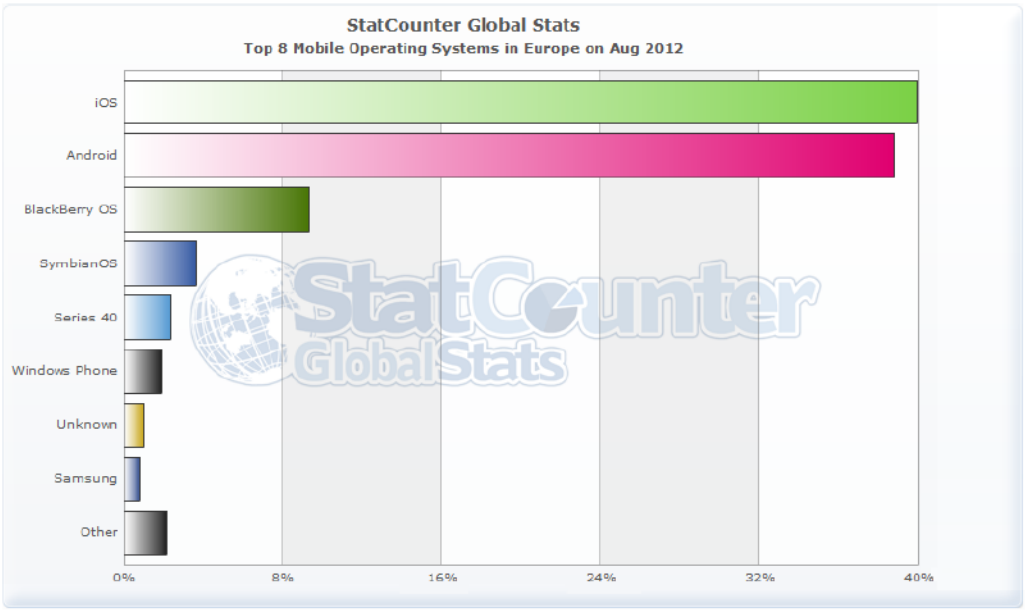


FIGURE 5.5 : REPARTITION DES SYSTEMES D'EXPLOITATION MOBILES EN AOUT 2012 EN EUROPE

Source : [sta13]

Les pourcentages calculés à un instant t ne peuvent être retenus et exploité pour entériner une décision quant au choix du système d'exploitation mobile pour le développement d'une application. Il est nécessaire de prendre en compte l'évolution de ces données en fonction du temps.

Ainsi, on constate que l'utilisation du système d'exploitation Android est en constante progression contrairement à iOS qui a tendance à reculer comme illustré sur la figure 5.6.

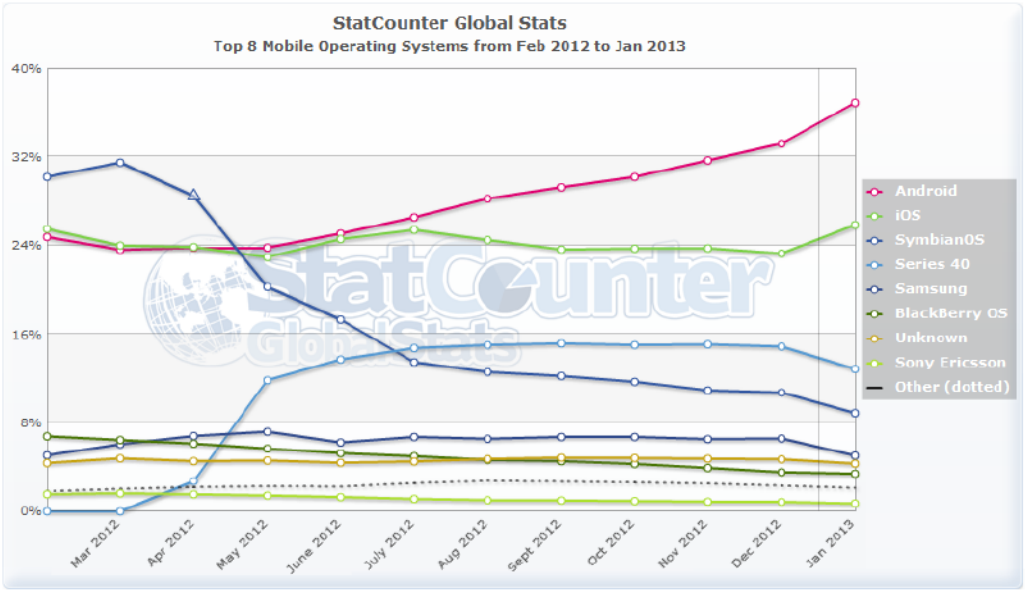


FIGURE 5.6 : EVOLUTION DE LA REPARTITION DES SYSTEMES D'EXPLOITATION MOBILES EN EUROPE

Source : [sta13]

La croissance du système d'exploitation Android est également mise en évidence lorsque l'on étudie la figure 5.7 qui représente la répartition des systèmes d'exploitation dans le monde en fonction du temps.

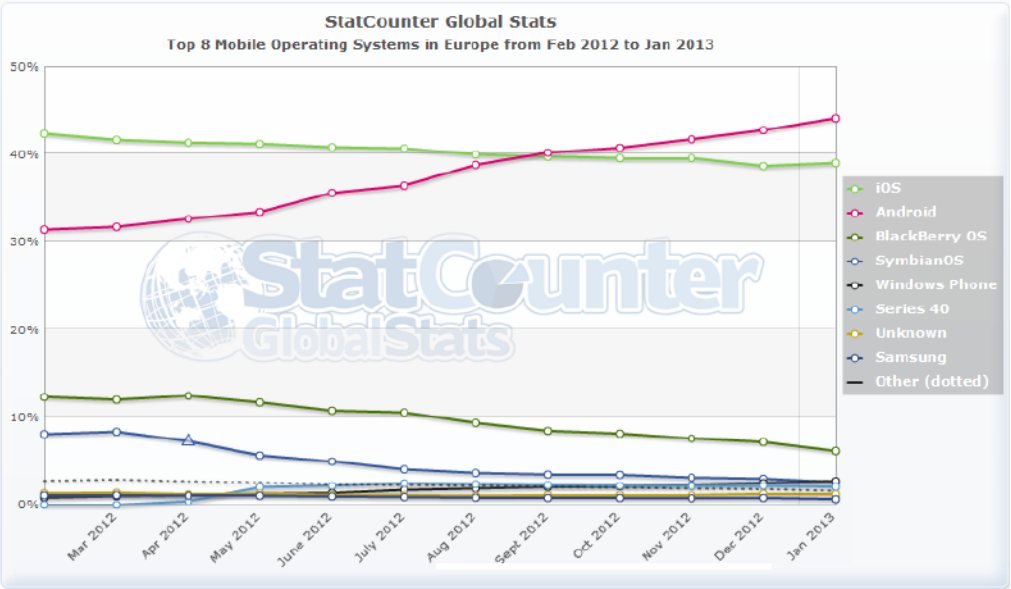


FIGURE 5.7 : EVOLUTION DE LA REPARTITION DES SYSTEMES D'EXPLOITATION MOBILES DANS LE MONDE

Source : [sta13]

Cette constante croissante résulte bien sûr de la gratuité de ce système d'exploitation incitant les constructeurs de terminaux mobiles tels que Samsung, HTC ou encore Sony à l'intégrer dans la plupart de leurs produits, téléphones intelligents ou tablettes. Mais elle découle également d'un nouveau phénomène « Amenez vos appareils personnels » plus communément connu sous le nom de « Bring Your Own Device » (BYOD). Cette pratique consiste à utiliser les équipements personnels des employés (plus particulièrement les terminaux mobiles) dans un contexte professionnel. Ce phénomène s'amplifie depuis 2012, mais pose de nombreux problèmes en termes sociaux, juridiques et de sécurité [ate13].

Le système d'exploitation Windows phone, trop jeune, n'a pas encore fait sa place dans le marché mobile. BlackBerry OS était très prisé auparavant par les entreprises mais a vu sa popularité baisser fortement avec l'arrivée d'iOS et d'Android en 2007. Toutefois, la sortie de la dixième version de ce système d'exploitation en février 2013 est à considérer. En effet, il propose une fonctionnalité intéressante, le « BlackBerry Balance » qui permet de définir deux espaces bien distincts et isolés l'un de l'autre dans le terminal mobile. Le premier est destiné à l'environnement du travail, le second à l'utilisation personnelle [zdn13b]. Il s'adapte donc particulièrement avec la nouvelle tendance de BYOD.

Le tableau 5.4 résume les différents systèmes d'exploitation mobiles existants ainsi que les terminaux mobiles compatibles.

Système d'exploitation	Licence	Modèles de terminal	Constructeurs
iOS Apple	Logiciel propriétaire	Apple (< 10)	Apple uniquement
Android Google	Distribution libre	Multiples (> 50)	HTC, LG, Samsung, Asus…
BlackBerry OS RIM	Logiciel propriétaire	BlackBerry (< 10)	BlackBerry uniquement
Windows Phone Microsoft	Logiciel propriétaire	Multiples (> 10)	Nokia, HTC …

TABLEAU 5.4 : TABLEAU COMPARATIF DES SYSTEMES D'EXPLOITATION MOBILES

Bien que le système d'exploitation mobile le plus usité dans le monde soit Android, les spécifications du projet GreenLAB imposent le développement de l'interface mobile sur un terminal iPhone. Ce choix s'explique en partie par les terminaux mobiles iPhone fournis par Persistent Systems au service commercial. Le projet LabMonitor a pour objectif d'être présenté aux clients potentiels afin de trouver des partenaires financiers. Il doit donc être démontrable sur les terminaux mobiles de l'équipe commerciale.

5.6.2 Type de développement mobile

Trois types de développement sont utilisables pour réaliser une application mobile : le développement HTML5, le développement natif ainsi que le développement hybride.

Développement HTML5

Le développement HTML5 d'une application mobile est tout simplement le développement d'une version mobile de l'interface web déjà existante pour le poste de travail. Les pages HTML sont générées sur le serveur et c'est le navigateur du téléphone qui se charge d'afficher l'interface de l'application. On peut citer comme exemple le célèbre moteur de recherche Google. Son interface destinée aux postes de travail classiques est accessible depuis l'adresse google.fr tandis que la version web mobile est quant à elle accessible à l'adresse m.google.fr [dev13] [jdn13].

On peut assimiler une application mobile HTML5 au client léger. Ce type de développement permet d'être compatible avec la plupart des terminaux mobiles existants qui possèdent un navigateur supportant HTML5. Mais le développement exclusivement HTML5 limite fortement l'utilisation des caractéristiques propres à un smartphone. Il est impossible d'accéder aux fonctionnalités spécifiques du téléphone, à savoir la puce GPS, les notifications, l'appareil photo numérique [dev13] [jdn13].

Développement natif

Ce type de développement est assimilé au client lourd. Il est spécifique à la plate-forme mobile choisie et utilise les outils de développements respectifs mis à disposition par les créateurs du système d'exploitation. Il permet de profiter pleinement de toutes les fonctionnalités du téléphone :

- Ecran tactile multipoint,
- Appareil photo numérique,
- Puce GPS,
- Accéléromètre.

Le développement natif permet donc de s'intégrer au mieux au système d'exploitation mobile. Mais un développement distinct est indispensable pour chaque système d'exploitation existant. Il faut donc multiplier les ressources humaines et matérielles puisque le code n'est pas réutilisable d'une plate-forme à l'autre. Les inconvénients en termes de temps de développement ainsi qu'en maintenance sont notables et à prendre en considération [dev13] [jdn13].

Développement hybride

Le développement hybride comme son nom l'indique combine le développement natif et HTML5. Il se caractérise de la façon suivante : l'essentiel de l'application est développé en HTML5, mais le code est ensuite embarqué à l'intérieure d'une application native très légère qui accède aux fonctionnalités du téléphone intelligent [dev13].

La plupart des développements hybrides se font au travers de divers cadriciels tels que Sencha Touch, PhoneGap ou encore Titanium qui facilitent le travail des équipes.

Bilan

Le domaine du développement mobile est en constante évolution. Les types de développements HTML5, natifs et hybrides possèdent respectivement des avantages et inconvénients recensés dans le tableau 5.5.

	HTML5	Natif	Hybride
Interface graphique	HTML/CSS	API natives	HTML/CSS
Performance	Moins rapide	Rapide	Moins rapide
Distribution	Web	Plate-forme de téléchargement officielle	Plate-forme de téléchargement officielle
Appareil photo	Non	Oui	Oui
Notifications	Non	Oui	Oui
Géo localisation	Oui	Oui	Oui
Technologies	HTML5, CSS, JavaScript	Langage propre au système d'exploitation	HTML5, CSS, JavaScript

TABLEAU 5.5 : TABLEAU COMPARATIF DES DEVELOPPEMENTS MOBILE HTML5, NATIF ET HYBRIDE

Le développement hybride a l'avantage de proposer toutes les caractéristiques du développement natif. En effet, il permet d'utiliser pleinement les fonctionnalités du terminal mobile et présente une facilité de développement par les langages utilisés : HTML5, CSS et JavaScript.

Cependant les terminaux iPhone étant une contrainte des services commerciaux, nous avons choisi de nous orienter vers le développement natif pour développer l'application mobile GreenLAB LabMonitor. Nous avons saisi l'opportunité d'étendre nos connaissances en nous familiarisant avec le langage de développement propre au système d'exploitation iOS.

Chapitre 6 Les implantations logicielles

Ce chapitre présente les différentes implémentations réalisées pour la mise en place de l'architecture logicielle frontale de la solution de supervision destinée au domaine de l'analyse chimique. Nous avons utilisé plusieurs méthodes et outils que nous présentons dans une première partie.

Les réalisations ont ensuite été organisées en quatre livraisons distinctes qui correspondent aux étapes majeures du développement de l'application LabMonitor. Chaque livraison se termine avec une démonstration aux responsables du projet et au service commercial pour ensuite fixer les objectifs de la livraison suivante et ainsi réorienter le développement des fonctionnalités en fonction des priorités imposées.

Dans un premier temps, nous avons implanté une première version de l'architecture logicielle du projet. Les différentes couches et interfaces logicielles sont créées pour obtenir une application simplifiée mais fonctionnelle.

Une seconde livraison présente plus particulièrement le déploiement de cette application dans l'informatique dématérialisée à l'extérieur de l'entreprise Persistent Systems et les contraintes de sécurité qui en découlent afin de protéger l'application ainsi que ses utilisateurs.

La troisième livraison se focalise sur la mise en place de l'environnement de développement destiné à la réalisation de l'application mobile cliente sur la plate-forme iOS d'Apple. Pour cela, nous nous sommes intéressés aux différentes possibilités et contraintes d'affichage qu'offrent les téléphones intelligents d'aujourd'hui.

La quatrième livraison, avec pour objectif une meilleure intégration des différentes applications qui composent la suite logicielle GreenLAB, décrit les travaux réalisés pour établir une communication entre les logiciels Workflow et LabMonitor.

Enfin, la dernière partie de ce chapitre résume les différentes estimations des travaux nécessaires à la mise en production de l'application LabMonitor, qui fournissent une base d'information précieuse au service commercial lors de leurs démarches clientèles.

6.1 Gestion du projet

6.1.1 Méthode Kanban

Le projet GreenLAB LabMonitor nécessite une architecture modulable et évolutive comme évoquée dans les chapitres précédents. De même, les choix techniques ne sont pas définitifs avant les différents tests d'utilisation, il faut donc pouvoir s'adapter aux changements futurs. Un mode de développement incrémental, avec des retours utilisateurs fréquents, s'impose donc. La méthode Kanban, connue pour avoir été utilisée précédemment dans des projets d'équipes, a été retenue pour la gestion de ce projet.

Définition de la méthode Kanban

Créé en 1950 par Toyota, l'outil Kanban a été mis au point pour optimiser le système de production. Il a pour but de créer une production sans gaspillage, également appelé « Lean Manufacturing ». Le mot Kanban signifie fiche ou étiquette. Cet outil a été conçu dans un premier temps pour les environnements de production industriels. Il permettait de suivre les lots de production. Sa mise en place entre deux postes de travail limite la production du poste amont aux besoins du poste aval. Kanban organise la production en flux tiré ce qui empêche les surproductions et donc les excédents de stocks entre chaque poste de la chaîne de production [KS13].

La méthode Kanban est désormais utilisée en développement logiciel grâce à sa transposition de la production au logiciel par Tom et Mary Poppendieck en 2003 dans le livre intitulé *Lean software development* [PP03]. Cette méthode est basée sur trois principes fondamentaux [CRI13] :

- Visualiser le flux de travail,
- Limiter le nombre de travaux en cours,
- Mesurer et optimiser les délais.

FIGURE 6.1 : EXEMPLE DE TABLEAU KANBAN

L'exemple de tableau Kanban de la figure 6.1 est composé de trois colonnes. Chaque fonctionnalité ou besoin du client est noté sur une fiche distincte. L'ensemble de ces fiches est stocké dans le carnet de produit. Une fiche circule de part et d'autre du tableau suivant le flux de travail. Elle passe successivement par l'état « A faire », « En cours » et « Terminé ». Cette technique permet à l'équipe de suivre très clairement l'évolution du travail.

Si les noms et nombres de colonnes ne sont pas limités, il est en revanche indispensable de borner le nombre de fiches par colonne. En effet, la méthode Kanban oblige à restreindre le travail en cours ou « Work In Progress » (WIP). Les équipes sont ainsi contraintes de se focaliser uniquement sur les fiches de la colonne « En cours » avant de passer à une fiche suivante. Les goulets d'étranglement sont aisément identifiés lorsqu'une colonne contient le nombre maximal de fiches

autorisé, alors que la suivante est vide. Par cette méthode qui met en évidence le WIP, on repère facilement une fiche immobilisée plusieurs jours consécutifs. Une solution peut être envisagée pour aider à la finalisation du travail.

Une partie importante de ce stage réside dans la recherche et l'étude des multiples techniques et technologies avant leur application au développement. La méthode Kanban permet de se concentrer uniquement sur un nombre limité d'objectifs (donc de fiches) tels que des estimations techniques ou technologiques. En fonction du résultat de ces dernières, le carnet de produit peut facilement être réorganisé ou rendu prioritaire par le responsable du produit sans impacter le travail des équipes. Cette méthode favorise une très grande réactivité des équipes au changement. Elle est particulièrement adaptée au projet LabMonitor alors qu'une grande partie du travail repose sur des estimations technologiques qui vont fortement impacter le développement de l'application.

La méthode Kanban nous a permis d'organiser le travail à réaliser tout au long de ce stage. Nous avons utilisé pour cela un outil de gestion de projet pour modéliser le tableau : VersionOne.

Outil utilisé : VersionOne

Nous avons déjà employé l'outil VersionOne dans différents projets pour organiser le travail au sein de l'entreprise. Ce logiciel est un outil de gestion de projet. Il est compatible avec la plupart des méthodes agiles de développement comme Scrum, Kanban, Lean ou XP. Il n'existe pas de contraintes quant à la définition du tableau. Chaque équipe peut adapter son tableau en fonction de son cœur de métier ou de ses contraintes. Avant de commencer les différentes études techniques et implantations logicielles, nous avons défini le tableau Kanban illustré par la figure 6.2.

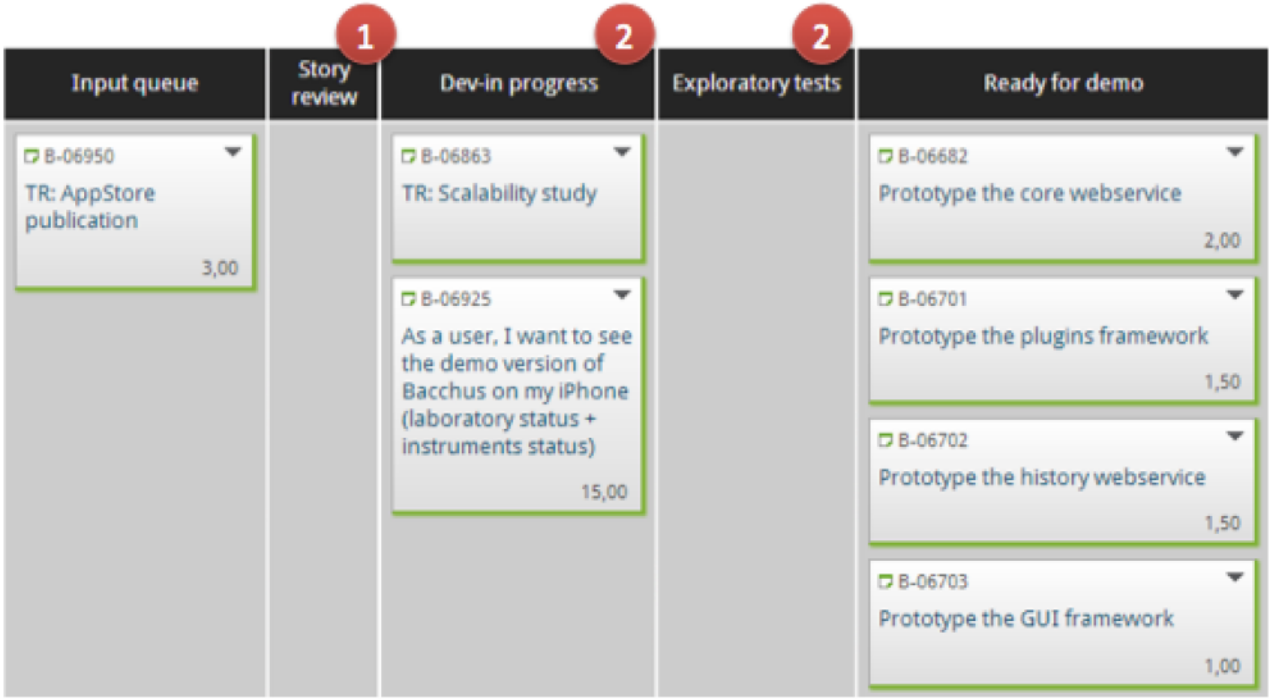


FIGURE 6.2 : TABLEAU KANBAN DU PROJET GREENLAB LABMONITOR

Ce tableau se compose de cinq colonnes distinctes :

- « Input queue » ou travail à faire : cette colonne représente, comme son nom l'indique, le travail à faire. Tant que la tâche se situe dans cette colonne, elle est sous le contrôle du responsable du produit et peut donc être mise en tête de liste pour être réalisée le plus tôt possible ou encore retardée pour favoriser du travail prioritaire,

- « Story review » ou revue de travail : cette étape permet de faire le point entre le développeur et le chimiste. Les critères sont revus en détail avant d'amorcer le développement.

- « Dev - in progress » ou en cours de développement : à partir d'ici, le développement de la fonctionnalité a commencé. Pour avancer vers la colonne suivante, il faut respecter certaines contraintes. Le code source doit être envoyé vers la plate-forme d'intégration continue pour construire les binaires de l'application. Cette construction doit être réussie ainsi que tous les tests automatiques associés au projet.

- « Exploratory tests » ou tests exploratoires : lorsqu'une tâche arrive dans cette colonne, le chimiste réalise des tests exploratoires sur la fonctionnalité réalisée pour vérifier les critères du point de vue utilisateur.

- « Ready for demo » ou prêt pour démonstration : cette dernière étape signifie que la fonctionnalité a été réalisée avec succès. Lors de la prochaine démonstration aux responsables du projet, celle-ci sera démontrée et validée définitivement.

6.1.2 Découpage

Avant de commencer le développement logiciel du projet, une réunion a été organisée entre ses différents acteurs : le développeur (moi-même), le représentant du client (chimiste) et le représentant commercial. Nous avons décrit les premières fonctionnalités de l'application et listé les besoins de chaque rôle utilisateur, identifiés précédemment (cf. 2.3.2).

Carnet de produit

La plupart des tâches du carnet de produit préliminaire représentent des estimations techniques qui doivent être réalisées avant de commencer les étapes techniques. Les quatre grandes étapes de réalisation sont les suivantes :

- L'architecture logicielle de l'application,
- L'informatique dématérialisée,
- L'application mobile,
- Les rapports d'optimisation du laboratoire.

La mise en place de ce carnet de produit nous a permis d'organiser le travail en quatre livraisons ou « release ». Chacune de ces dernières représente un ensemble de fonctionnalité suffisant pour livrer une application fonctionnelle.

Livraisons

La première livraison vise la mise en place des différents éléments qui composent l'architecture logicielle adoptée précédemment. Les différentes technologies choisies sont utilisées pour fournir une première version de l'application LabMonitor. Seule l'interface accessible via le navigateur est implantée.

La seconde livraison est centrée sur le déploiement dans l'informatique dématérialisée pour mettre à disposition l'application LabMonitor au service commercial en vue de démonstrations commerciales futures. Une amélioration de l'interface web avec l'utilisation de pages web élaborées par une équipe spécialisée est également réalisée.

L'application mobile est le principal objet de la troisième livraison. Elle est implantée et déployée sur les terminaux mobiles de l'entreprise pour permettre des démonstrations par le service commercial.

La quatrième livraison a pour objectif l'implantation des rapports d'utilisation des instruments analytiques et du laboratoire. Ces rapports sont consultables par les responsables du laboratoire grâce à l'interface web de l'application LabMonitor.

La figure 6.3 ci-après définit l'organisation des différentes étapes de la réalisation de l'application LabMonitor au cours des neuf mois du stage.

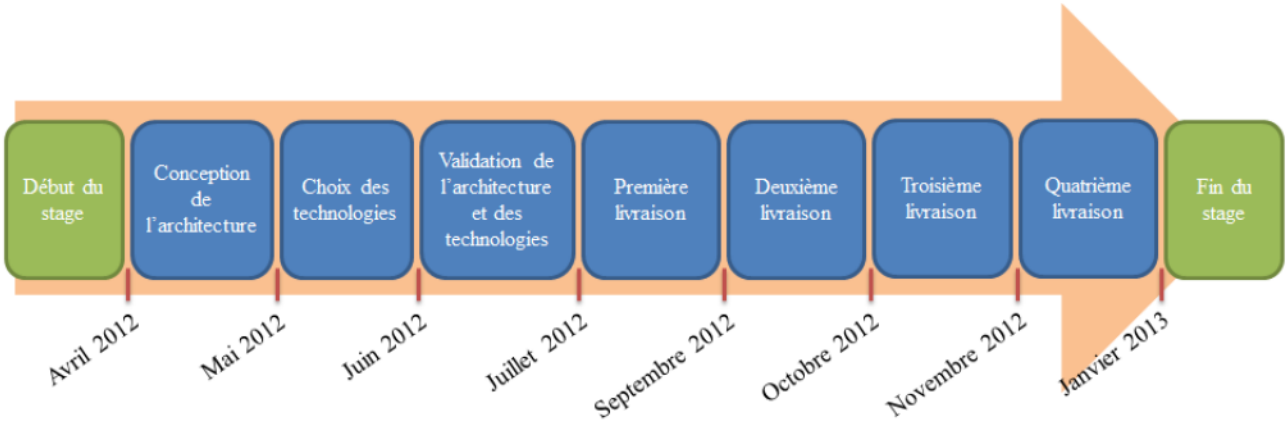


FIGURE 6.3 : CALENDRIER DU STAGE

6.2 Validation de l'architecture et des technologies

6.2.1 Prototypage

Dans l'optique de nous familiariser avec les technologies définies dans le chapitre précédent, M. Laurent Tardif (tuteur du stage) nous a demandé de prototyper très rapidement l'outil de supervision de l'élément supervisé à l'interface web. Le développement de ce prototype a été réalisé en une semaine et utilise une base de données MySQL, des services web REST développés en Java

et une page HTML générée par des scripts PHP sur le serveur HTTP. Cette application a été déployée sur un ensemble de serveurs pour mettre en évidence la faisabilité du projet.

6.2.2 Tests de performance

Après l'implantation de ce prototype, des tests de performance ont été lancés pour éprouver l'architecture et les technologies employées. Lors des tests de performance, l'outil PerfMon de Microsoft a été installé pour surveiller les utilisations du disque dur, de la mémoire vive, du réseau et du processeur.

Les caractéristiques du test sont : le simulateur installé sur 3 serveurs différents génère des mesures de supervision chaque seconde. Il simule 10 instruments analytiques. Le débit est donc de 10 mesures de supervision à la seconde. De même, des navigateurs présentant l'interface web du prototype sont lancés sur 3 postes de travail client avec un paramètre de rafraichissement de 5 secondes. La figure 6.4 illustre le déploiement logiciel et matériel du prototype.

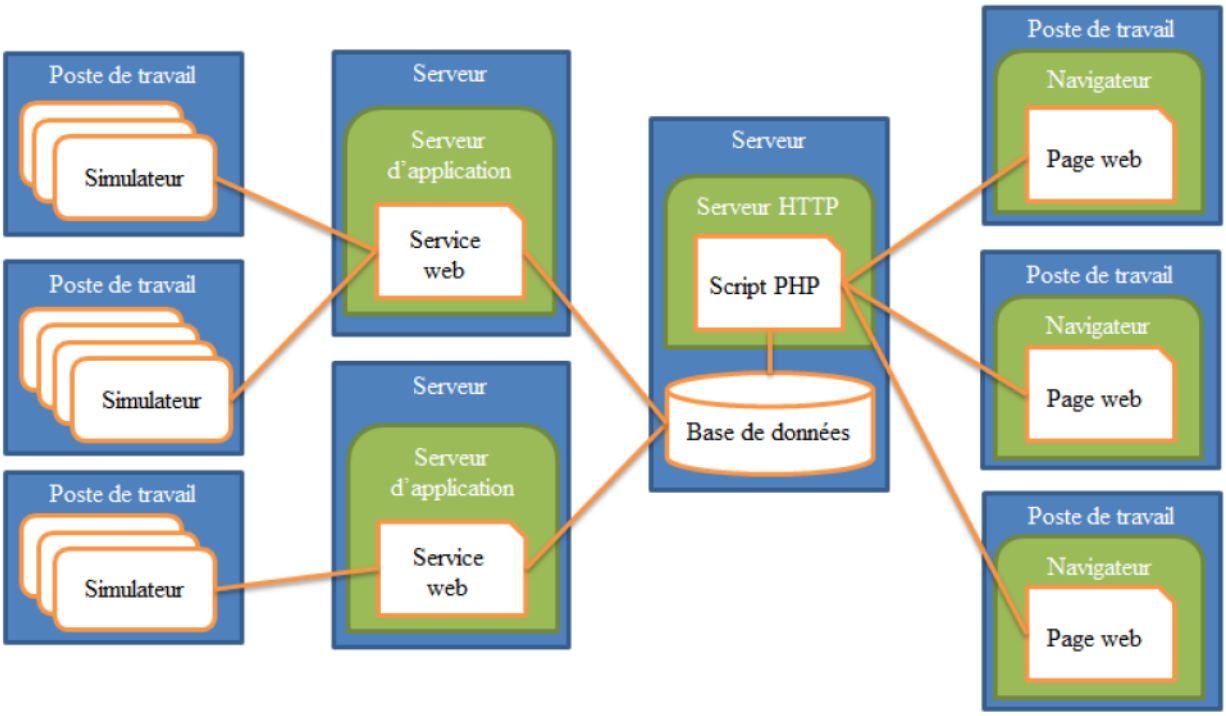


FIGURE 6.4 : DEPLOIEMENT LOGICIEL ET MATERIEL DU PROTOTYPE

Avant le lancement du test, la base de données était vide, aucune mesure de supervision n'était enregistrée, mis à part les 10 instruments. Le test de performance s'est déroulé pendant 8 jours sans interruption. Il a permis de générer environ 7 millions de mesures de supervision sans que les performances des serveurs soient impactées.

6.2.3 Bilan

Le prototypage de l'outil de supervision permet dans un premier temps une approche préliminaire des technologies qui sont utilisées pour le développement de chaque module de l'application LabMonitor. Dans un second temps, il met en évidence la modularité de déploiement d'une application reposant sur l'architecture n tiers. En effet, chaque couche étant distinctement séparée, elle n'impose pas de déploiement matériel. Les différents modules de l'application sont tout à fait installables sur un seul serveur comme sur trois serveurs différents, comme le montre le test précédent. L'accent est mis également sur la facilité de déploiement de ce type d'architecture logicielle.

Cette première étape de développement valide l'architecture logicielle frontale de l'application LabMonitor ainsi que les technologies élues précédemment (cf. Chapitre 4 et Chapitre 5).

6.3 Mise en place de l'intégration continue

Les premières bibliothèques développées pour le prototype ont été construites localement sur notre poste de travail de développement. Pour plus de rigueur dans la programmation de l'application LabMonitor, nous utilisons désormais une plate-forme d'intégration continue qui est en charge de la construction de chaque bibliothèque Java qui compose l'application GreenLAB LabMonitor.

6.3.1 Objectifs

L'intégration continue est un ensemble de pratiques utilisées lors d'un développement logiciel. Cela consiste à vérifier que chaque modification de code source réalisée par un développeur de l'équipe n'entraine pas de régression dans le logiciel. La mise en place d'une intégration continue implique :

- Un code source partagé,
- Une intégration quotidienne des modifications par les développeurs,
- Des tests d'intégration développés pour tester le logiciel.

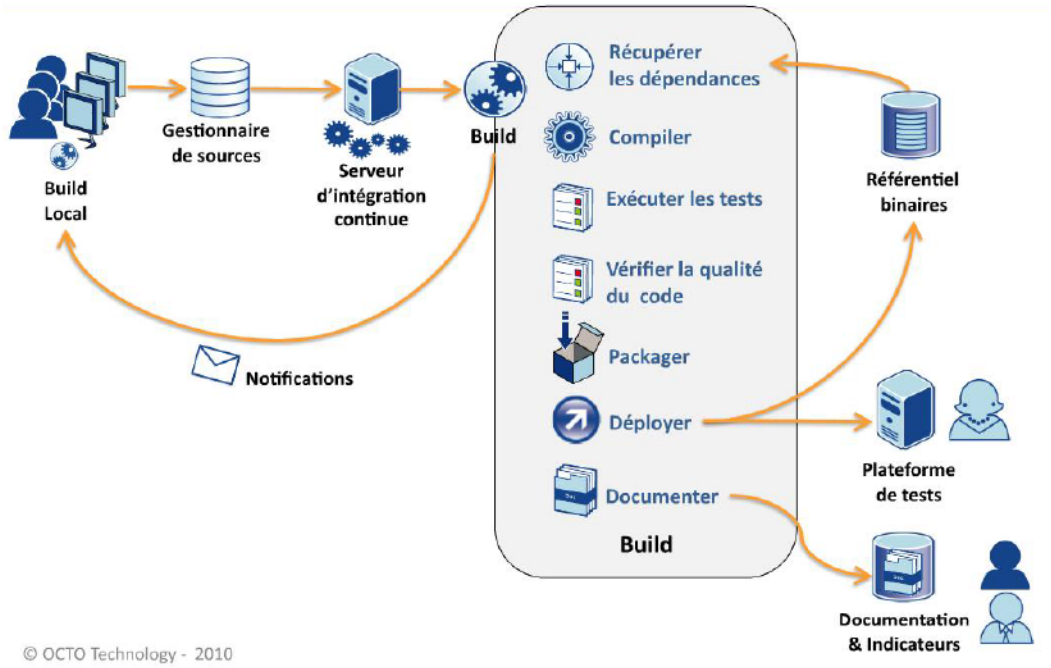


FIGURE 6.5 : INTEGRATION CONTINUE

Source : [oct13]

La figure 6.5 décompose le processus d'intégration continue. Le développeur effectue des modifications du code source de l'application. Il envoie ensuite ses modifications au gestionnaire de sources. La plate-forme d'intégration continue qui scrute les changements du gestionnaire de source déclenche alors la construction (plus couramment nommé « build ») de l'application. Cette dernière est découpée en plusieurs étapes :

- Récupération des dépendances dans le référentiel des binaires,
- Construction ou « compilation » des binaires,
- Exécution des tests des binaires construits,
- Vérification de la qualité du code source des binaires,
- Construction de l'installateur ou du paquet,
- Déploiement de l'ensemble des binaires sur une plate-forme de tests,
- Construction de documentations relatives au code source et à l'application.

Cette plate-forme d'intégration continue est basée sur un ensemble d'outils.

6.3.2 Outils

Outil d'intégration continue : Jenkins

Jenkins est un outil d'intégration continue permettant de mettre en pratique les principes de l'intégration continue, à savoir les trois cités précédemment : le code, l'intégration et les tests. Il est capable de s'interfacer avec le système de gestion de versions Subversion et exécute des projets basés sur Maven ou encore des scripts en Shell Unix ou batch Windows.

Gestionnaire de sources : Subversion

Le gestionnaire de sources, comme son nom l'indique, gère le code source du projet en cours de développement. Un serveur Subversion est installé dans les locaux de Persistent Systems France depuis plusieurs années. Il fonctionne de la façon suivante : chaque développeur possède sur son poste de travail un client à l'application Subversion (ici TortoiseSVN). Lorsque le développeur effectue une modification, il envoie le code source modifié au serveur Subversion via le client. Ce changement est listé dans l'historique de modification du fichier impliqué.

L'utilisation de cet outil au sein du site français est clairement identifiée. Chaque modification doit être obligatoirement associée à un numéro de fiche du tableau Kanban et à un message expliquant clairement le but de cette modification. Ces bonnes pratiques mises en place garantissent une grande traçabilité du code créé.

Construction des binaires : Maven

Le logiciel Maven sert à ordonnancer et à piloter de façon automatique l'ensemble des tâches qui à partir du code source, génère le logiciel opérationnel. Il est particulièrement adapté aux projets développés en Java. Des outils similaires comme Make ou Ant peuvent également être utilisés. Mais Maven est plus facile d'utilisation grâce à la standardisation de sa configuration, c'est-à-dire l'utilisation de fichiers XML.

Référentiel des binaires : Artifactory

Artifactory est un dépôt qui stocke et distribue des fichiers binaires en réponse à une demande. Cette demande peut être formulée par un développeur lors d'une compilation locale ou par la plate-forme d'intégration continue lors d'une construction de binaires. Artifactory permet de gérer les conflits de versions de bibliothèques ou binaires. Tous les binaires sont centralisés. Comme ce dépôt est local, le téléchargement est donc plus rapide lors de constructions.

6.3.3 Qualité logicielle

L'ensemble de ces outils permet d'assurer une traçabilité précise de l'application réalisée. Au-delà de cet objectif, la plate-forme d'intégration continue permet d'améliorer la qualité logicielle. Cette dernière ne possède pas de description précise. Elle peut être interprétée de la façon suivante :

- Qualité du code source,
- Qualité de conception du logiciel,
- Qualité perçue par le client final…

L'outil d'intégration continue Jenkins nous permet d'intervenir au niveau de la qualité du code source. Il autorise l'installation de différents greffons afin d'enrichir ses fonctionnalités. Nous avons donc installé plusieurs greffons destinés à vérifier la qualité du code source des applications.

Greffons CheckStyle et PMD

CheckStyle et PMD sont deux outils de contrôle de code source complémentaires. Le premier cible l'application des conventions associées au langage de programmation et le second, les éventuelles mauvaises pratiques rencontrées couramment.

Parmi les conventions associées au langage Java, on peut citer :

- Commentaires Javadoc pour les classes, attributs et méthodes,
- Conventions de nommage des classes, attributs, méthodes,
- Limitation du nombre de paramètres par méthode…

L'outil PMD contrôle plus profondément le code source. Il identifie plus particulièrement :

- Le code inutilisé,
- La duplication de code,
- La création d'objet inutile,
- La complexité cyclomatique.

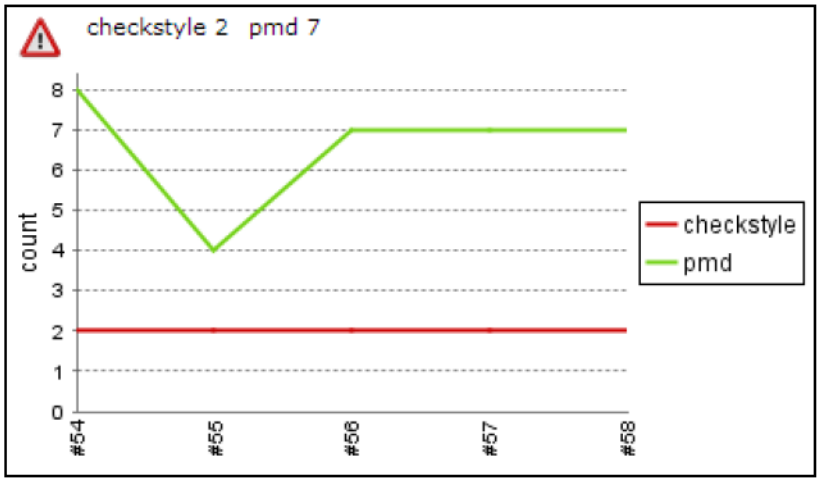


FIGURE 6.6 : RAPPORT GENERE PAR LES GREFFONS CHECKSTYLE ET PMD

Ces deux outils sont complémentaires et permettent d'assurer une certaine qualité de code source. Lors de chaque construction successive de binaire sur la plate-forme d'intégration continue, les fautes identifiées par ces deux greffons sont enregistrées et affichées. On peut ainsi suivre l'évolution de la qualité du code comme illustré dans la figure 6.6. On constate par exemple que lors de la 56e construction du binaire, le nombre de fautes identifiées par l'outil PMD (courbe verte) a augmenté. Le code source doit donc être corrigé pour assurer la qualité du projet LabMonitor.

Greffon Cobertura

L'outil Cobertura cible, à l'inverse des deux précédents, la couverture du code source par les tests. Il permet de calculer précisément en pourcentage le code qui est éprouvé par des tests automatiques. On peut ainsi connaître précisément le taux de couverture de test de chaque fichier, classe ou méthode du module construit par la plate-forme d'intégration continue (cf. figure 6.7).

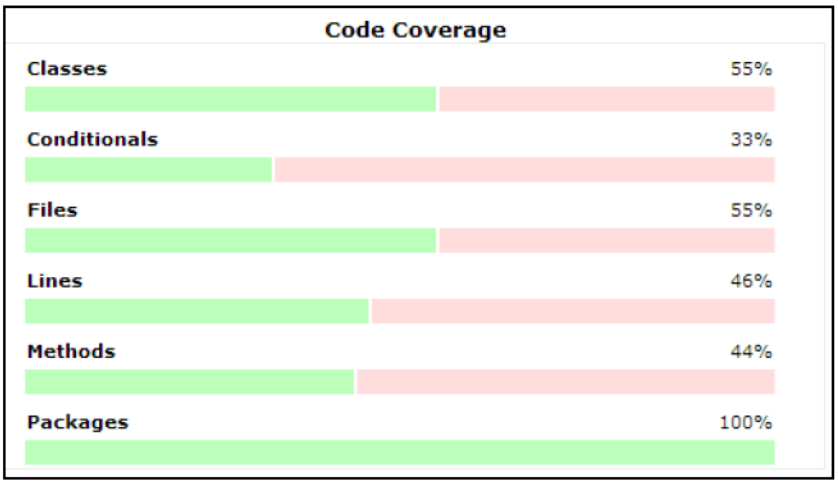


FIGURE 6.7 : RAPPORT GENERE PAR LE GREFFON COBERTURA

Un serveur d'intégration continue commun aux différents projets GreenLAB a été créé afin de vérifier régulièrement la qualité du code et d'automatiser les différentes étapes de la construction de la suite logicielle GreenLAB. Cette plate-forme facilite le travail quotidien des équipes.

6.4 Première livraison

6.4.1 Objectifs

L'objectif de cette livraison est de réaliser une première version de l'application LabMonitor. Cette partie décrit l'implantation des trois couches logicielles : données, métier et présentation. Les spécifications des fonctionnalités demandées de cette version sont les suivantes : l'interface web doit être capable d'afficher la liste des instruments analytiques supervisés ainsi que leur statut respectif. Elle doit également présenter une agrégation de ces différents statuts afin de fournir un statut global du laboratoire.

6.4.2 Définition et installation de la base de données.

Définition du schéma de la base de données

Pour réaliser le schéma de la base de données tronquée qui nous servira tout au long de la réalisation de l'application LabMonitor, nous devons recenser les différentes informations à conserver :

- Les instruments analytiques à superviser : un identifiant unique, un nom ainsi que d'autres informations complémentaires,

- Les mesures relevées relatives aux instruments : un identifiant unique, l'élément supervisé qui lui est associé, le statut du supervisé ainsi que la date et l'heure de la mesure.

Nous avons donc défini le schéma de base de données MySQL représenté dans la figure 6.8. Une entrée de la table « modules » représente l'ensemble des données associées à l'instrument. La seconde table « events » stocke chaque entrée représentant une mesure réalisée par l'outil de supervision. Le lien entre les deux tables est établi grâce à la clé étrangère « module_id ». Une entrée de la table « modules » peut posséder plusieurs entrées de la table « events ».

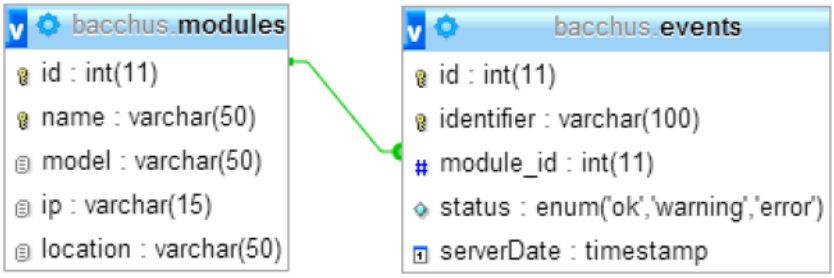


FIGURE 6.8 : SCHEMA DE LA BASE DE DONNEES

Une entrée de la table « events » représente une mesure de supervision. Elle est associée à un instrument et décrit le statut de l'instrument dont voici le détail :

- « ok » : l'instrument communique avec le serveur de supervision et est en état de marche,
- « warning » : la communication est établie, mais l'instrument signale une alerte,
- « error » : l'instrument n'est pas forcément joignable à travers le réseau ou renvoie une erreur.

Nous avons déployé cette base de données sur nos postes de développement en utilisant l'outil EasyPHP. Il s'agit d'une plate-forme de développement Web qui permet de faire fonctionner localement une base de données MySQL, un serveur web Apache et un interpréteur de script PHP. Cet ensemble d'outils fournit un environnement d'exécution local pour l'application web en cours de développement.

Simulateur de mesures de supervision

Des mesures de supervisions sont indispensables pour continuer le développement et plus particulièrement pour simuler les échanges entre les différentes couches de l'application. Nous avons donc développé un programme qui simule des mesures effectuées par le moteur de supervision (cf. figure 6.9). Réalisé en C#, il permet lors de l'appui sur le bouton d'un statut possible de l'instrument d'ajouter une entrée dans la base de données et plus particulièrement dans la table « events » décrite précédemment.

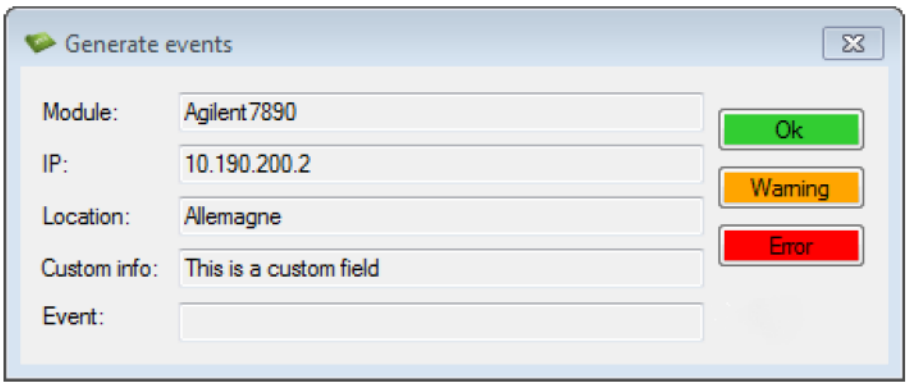


FIGURE 6.9 : SIMULATEUR DE MESURES DE SUPERVISION REALISE EN C#

Le choix du langage utilisé pour développer ce simulateur n'est pas issu d'une étude préliminaire, mais résulte uniquement de nos compétences antérieures au stage. Ce simulateur ne présente aucun intérêt pour l'utilisation du projet LabMonitor en mode de production, mais permet d'enregistrer des mesures de supervision afin de tester les couches logicielles métier et présentation. Nous avons donc retenu ce langage de programmation pour le réaliser le plus rapidement possible.

6.4.3 Implantation de la couche métier

La couche métier de l'application GreenLAB qui doit interpréter les mesures de supervision nécessite l'utilisation d'un serveur d'application fournissant un environnement d'exécution aux modules métier. Contrairement au serveur web nécessaire à la couche de présentation qui exécute des scripts PHP grâce à un interpréteur, il permet d'exécuter les modules logiciels Java.

Serveur d'application embarqué

Pour développer les modules Java et rendre leurs fonctionnalités accessibles via le réseau, nous avons mis en place un serveur d'application. Dans un premier temps, nous utilisons un serveur d'application embarqué dans le code source de l'application : Grizzly. Il permet de mettre en place très simplement et rapidement un serveur d'application pour utiliser les fonctionnalités de l'application LabMonitor.

```
60  /**
61   * Start application server
62   *
63   * @return the application server
64   * @throws IOException
65   */
66  protected static HttpServer startServer() throws IOException
67  {
68      final String packageName = ApplicationServer.class.getPackage().getName();
69      final ResourceConfig resourceConfig = new PackagesResourceConfig(packageName);
70      return GrizzlyServerFactory.createHttpServer(BASE_URI, resourceConfig);
71  }
```

FIGURE 6.10 : CREATION DU SERVEUR D'APPLICATION EMBARQUE

En quelques lignes, le serveur d'application démarre et se trouve accessible sur le réseau à travers l'URI défini basé sur l'IP du poste de travail qui l'exécute (cf. figure 6.10). Il présente plusieurs avantages : aucune installation n'est requise en dehors de l'environnement d'exécution Java ce qui permet de déployer très rapidement. Il peut facilement être utilisé pour la mise en place de tests d'intégration de l'application.

Modules métier

Dans cette première version de l'application, le rôle de la couche métier est très simple. Elle se contente de récupérer les informations nécessaires dans le système de stockage pour les retourner à l'interface. La première requête destinée à la base de données est de lister tous les instruments analytiques enregistrés ainsi que leur statut actuel qui est lui-même défini par le statut de la dernière mesure de supervision entrée dans la base. La seconde fonctionnalité souhaitée est l'agrégation de tous les statuts des instruments analytiques du laboratoire de la façon suivante :

- Si tous les instruments sont disponibles, le statut du laboratoire est « ok »,
- Si au moins un instrument présente une alerte, le statut du laboratoire est « warning »,
- Si au moins un instrument présente une erreur alors le statut du laboratoire est « error ».

Une simple bibliothèque Java utilisant l'API JDBC pour se connecter à la base de données a été créée et suffit à satisfaire les besoins actuels de l'application LabMonitor.

Services web

Les services web permettent d'exposer sur le réseau les fonctionnalités des modules métier. Nous avons donc réalisé deux services web pour mettre à disposition les méthodes conçues précédemment. Pour cela nous avons utilisé une interface de programmation Java qui facilite le développement de services web grâce à des annotations.

```java
/**
 * InstrumentsService web service class.
 * @author dorine.mazeyrat
 */
@Path("/instruments")
public class InstrumentsService
{
    /**
     * Get instrument.
     * @param instrumentName instrument name.
     * @return HTTP response with JSON information.
     */
    @GET
    @Path("{instrumentName}")
    @Produces(MediaType.APPLICATION_JSON)
    public Response getInstrument(@PathParam("instrumentName") String instrumentName)
    {
        final Connection connection = connectionPool.getConnection();
        try
        {
            if (connection == null)
            {
                LOGGER.severe(Resources.ERROR_CONNECTION_DB);
                return Response.serverError().entity(Resources.ERROR_CONNECTION_DB).build();
            }

            final Instrument instrument = Manager.getInstrument(connection, instrumentName);
            return Response.ok(instrument, MediaType.APPLICATION_JSON).build();
        } finally
        {
            connectionPool.releaseConnection(connection);
        }
    }
```

FIGURE 6.11 : SERVICE WEB UTILISANT L'API JAX-RS

JAX-RS permet, en quelques lignes de code, d'exposer la méthode « getInstrument() » comme ressource identifiable et accessible via HTTP. Différentes annotations sont utilisées, on peut noter plus particulièrement celles de la figure 6.11 :

- « @GET » : c'est la méthode HTTP appliquée par la requête pour accéder à la ressource voulue,
- « @Path » : cette annotation définit le chemin relatif pour accéder à la méthode,
- « @Produces » : elle spécifie le format de la réponse du service web. Dans cet exemple le format de données utilisé est JSON,
- « @PathParam » : cette dernière fait le lien avec le paramètre « instrumentName » dans le chemin relatif.

La méthode est désormais accessible sur le poste de développement via la requête HTTP : http://…/instruments/{instrumentName} où {instrumentName} représente un paramètre d'entrée.

Lors des requêtes via le client universel vers le serveur, il est essentiel de standardiser les messages et plus particulièrement le statut des réponses. Pour cela, on utilise les codes HTTP qui prédéfinissent des statuts et des messages de réponses. On peut constater dans la figure 6.11, l'utilisation de deux statuts : « ok » et « serverError » qui correspondent respectivement aux codes HTTP 200 et 500. Le premier signifie que la requête a été traitée avec succès tandis que le second signale qu'une erreur s'est produite sur le serveur et qu'aucune réponse n'a pu être donnée. Dans le service web de notre exemple, un message d'erreur est retourné lorsque la connexion à la base de données a échoué.

La réponse utilise le format JSON qui permet de présenter et d'organiser très clairement les informations relatives à l'instrument, comme illustré sur la figure 6.12.

```
{
    "name": "Agilent7890GC_US_01",
    "model": "Agilent GC 7890",
    "location": "Los Angeles, United States",
    "ipAdress": "12.190.200.11"
}
```

FIGURE 6.12 : REPONSE DU SERVICE WEB AU FORMAT JSON

Lors du démarrage du serveur d'application embarqué Grizzly, celui-ci génère automatiquement un fichier « Web Application Description Language » (WADL). Ce fichier, représenté par la figure 6.13, décrit l'ensemble des services web disponibles de l'application. Cette description écrite au format XML énonce précisément comment utiliser le service web exposé. On retrouve dans l'exemple suivant le nom de notre service web, l'URL d'accès, la méthode HTTP appliquée et enfin le format de réponse.

```
▼<application xmlns="http://wadl.dev.java.net/2009/02">
  <doc xmlns:jersey="http://jersey.java.net/" jersey:generatedBy="Jersey: 1.14 09/09/2012 05:39 PM"/>
  <grammars/>
 ▼<resources base="http://10.190.200.139:9998/">
  ▼<resource path="/instruments">
   ▼<resource path="{instrumentName}">
      <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="instrumentName" style="template" type="xs:string"/>
    ▼<method id="getInstrument" name="GET">
     ▼<response>
        <representation mediaType="application/json"/>
      </response>
     </method>
    </resource>
   </resource>
  </resources>
 </application>
```

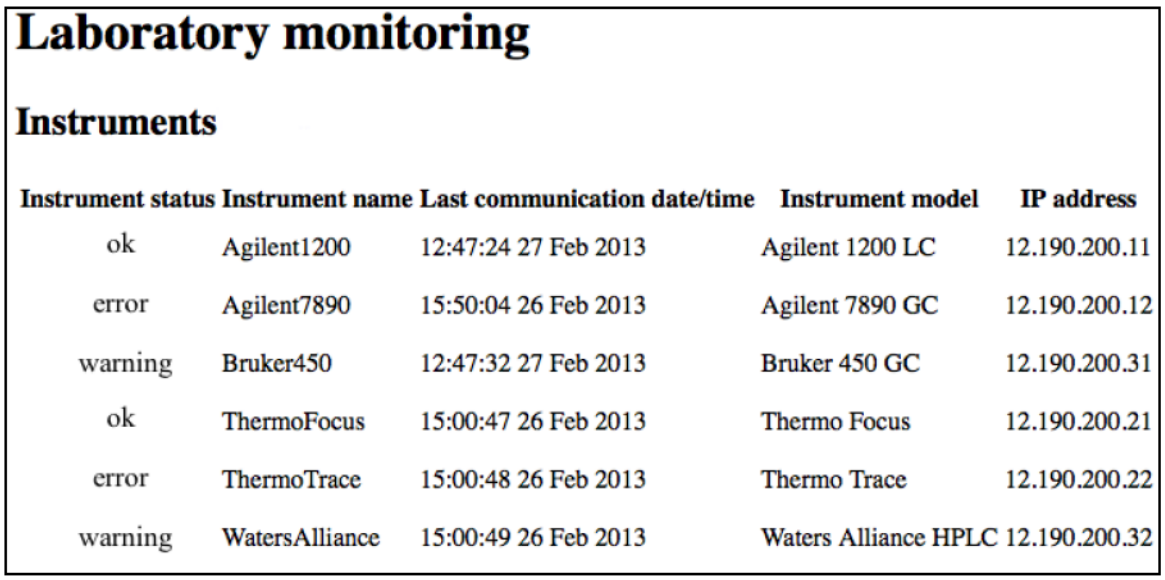
FIGURE 6.13 : DESCRIPTION DU SERVICE WEB

Les différentes méthodes HTTP existantes Get, Post, Put et Delete permettent respectivement de lire, d'écrire, de mettre à jour et enfin de supprimer une ressource.

Les méthodes de traitement sont désormais accessibles sur le réseau. Nous pouvons donc les utiliser pour concevoir la couche présentation responsable des interfaces utilisateurs.

6.4.4 Interfaces graphiques web

Le système de stockage et les services web étant implantés, nous pouvons commencer l'ébauche d'une interface web pour afficher les résultats de supervision à l'utilisateur. Dans un premier temps, nous nous consacrons au rôle du technicien du laboratoire. Pour rappel, le rôle du technicien est de s'assurer de la maintenance des différents instruments du laboratoire. Il doit donc pour cela être capable d'identifier rapidement les instruments en panne grâce à cette interface.



Laboratory monitoring

Instruments

Instrument status	Instrument name	Last communication date/time	Instrument model	IP address
ok	Agilent1200	12:47:24 27 Feb 2013	Agilent 1200 LC	12.190.200.11
error	Agilent7890	15:50:04 26 Feb 2013	Agilent 7890 GC	12.190.200.12
warning	Bruker450	12:47:32 27 Feb 2013	Bruker 450 GC	12.190.200.31
ok	ThermoFocus	15:00:47 26 Feb 2013	Thermo Focus	12.190.200.21
error	ThermoTrace	15:00:48 26 Feb 2013	Thermo Trace	12.190.200.22
warning	WatersAlliance	15:00:49 26 Feb 2013	Waters Alliance HPLC	12.190.200.32

FIGURE 6.14 : PREMIERE VERSION DE L'INTERFACE WEB

Nous avons réalisé une première version de l'écran d'affichage de la liste des différents instruments comme illustrée dans la figure 6.14. Cette page HTML est générée par des scripts PHP exécutés par l'environnement de développement local EasyPHP. Les scripts PHP développés utilisent la bibliothèque de requêtes aux URL de client ou « Client URL Request Library » (cURL) afin de communiquer avec les services web. Lors de la réception de la réponse au format JSON, les scripts PHP organisent les informations de supervision dans la page HTML. L'écran représenté énumère le statut de chaque instrument du laboratoire en l'identifiant, son adresse IP ainsi que des informations complémentaires comme la date et l'heure de la dernière communication.

6.4.5 Bilan

Cette première étape dans le développement du projet GreenLAB LabMonitor nous a permis de mettre en place les différents éléments de l'architecture logicielle de l'application GreenLAB LabMonitor grâce aux technologies définies dans le chapitre précédent. Ainsi, la base de données,

les modules métier, les services web et enfin l'interface web sont désormais disponibles dans une version simplifiée.

Lors de la démonstration de l'application aux responsables du projet, il a été décidé la mise à disposition d'un accès à l'application au service commercial pour des démonstrations clientèles. Pour cela, une nouvelle charte graphique pour la mise en valeur du produit doit être appliquée. La priorité de la seconde étape du projet sera donc les interfaces et la mise à disposition de l'application actuelle à l'équipe commerciale.

6.5 Deuxième livraison

6.5.1 Objectifs

Grâce à la première version réalisée et la démonstration du projet, l'orientation de cette deuxième étape a été fixée. L'accent est mis sur une interface web plus conviviale et ergonomique ainsi que sur le déploiement de l'application dans l'informatique dématérialisée afin de démarrer le plus rapidement possible les démonstrations commerciales en vue de démarcher des clients.

6.5.2 Application de la charte graphique

Les projets Workflow et Search avançant en parallèle, il a été décidé de l'utilisation d'une charte graphique commune aux trois différents projets GreenLAB. Les pages HTML et les feuilles de styles CSS ont été réalisées par une équipe spécialisée de Persistent Systems située à Goa, Inde.

Maquettes des interfaces graphiques web

Afin de préparer les conférences téléphoniques pour présenter nos différents projets et nos besoins en termes de développement, nous avons élaboré des prototypes d'écran grâce à l'outil Pencil. Cet outil permet de créer des maquettes d'interface graphique de logiciel facilement.

Ces différentes maquettes ont permis de spécifier précisément nos besoins aux équipes indiennes. Les différents fichiers reçus en retour n'ont demandé que de faibles modifications pour être intégrés dans l'application GreenLAB LabMonitor.

Voici, dans la figure 6.15, le prototype de l'écran réalisé qui affiche la liste des différents modules supervisés ainsi que des informations complémentaires les concernant.

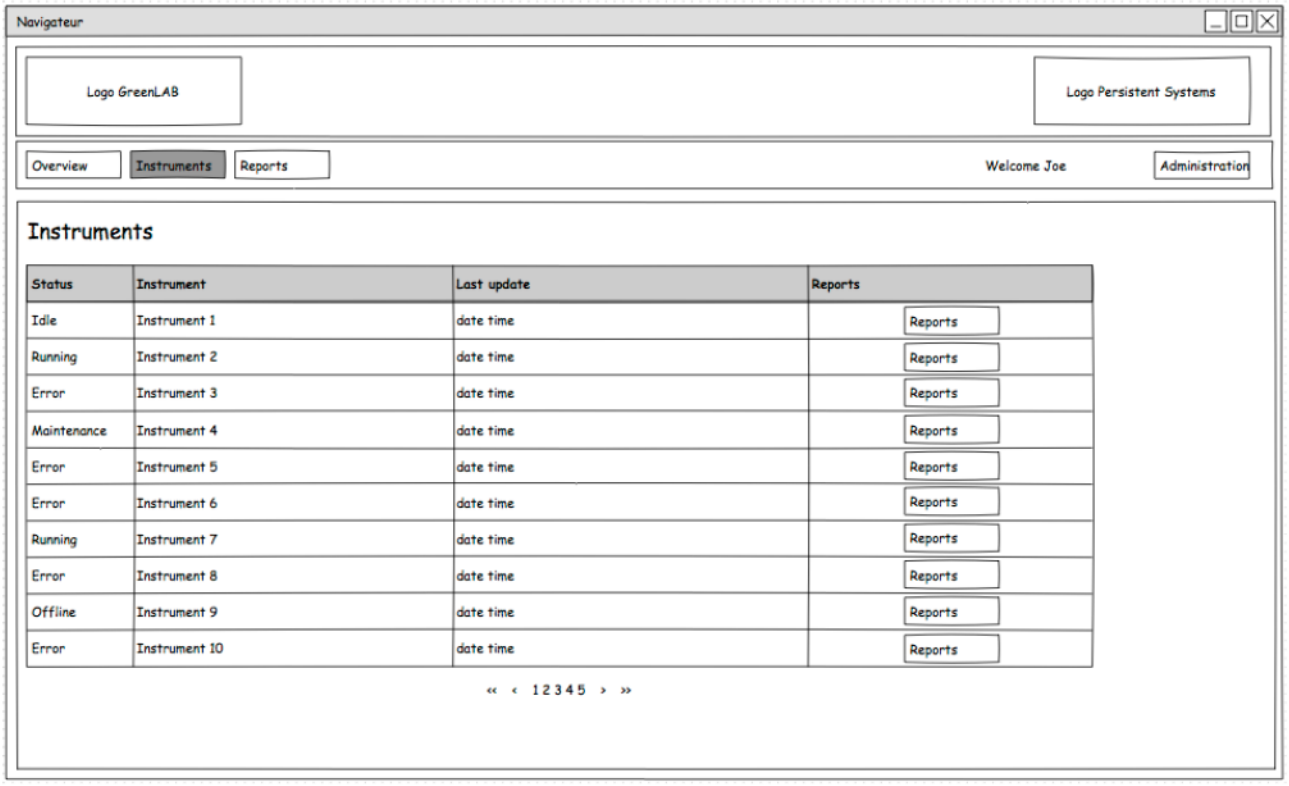


FIGURE 6.15 : PROTOTYPE D'ECRAN POUR L'INTERFACE WEB

Implantation des nouvelles interfaces graphiques web

L'implantation de ces nouvelles pages HTML et de ces feuilles de styles CSS n'a pas soulevé de réelles difficultés grâce à la préparation des différentes maquettes d'écran. En raison de la séparation en couche logicielle de l'application LabMonitor, le changement des interfaces n'a généré aucun impact sur les couches métier et de données.

FIGURE 6.16 : SELECTION D'ECRANS DE LA SECONDE VERSION DE L'INTERFACE WEB

Le résultat obtenu est visible sur la figure 6.16. Parmi les nouveaux écrans, on peut distinguer un écran de connexion. Un des objectifs de cette étape du projet est de rendre l'application disponible à l'extérieur de l'entreprise. En effet, le service commercial doit pouvoir démontrer le produit lors de ses déplacements professionnels. Il n'a pas toujours accès au réseau de l'entreprise. L'application doit donc être hébergée dans l'informatique dématérialisée et par mesure de sécurité, différents dispositifs sont installés pour garantir la confidentialité et la sécurité de l'application et de ses utilisateurs.

6.5.3 Préparation au déploiement dans l'informatique dématérialisée

Mise en place de l'authentification

Afin de restreindre l'accès à l'application web LabMonitor, un système d'authentification a été ajouté. Lorsque l'on souhaite mettre en place un accès sécurisé avec le langage PHP, une solution est l'utilisation des sessions PHP. Ce mécanisme permet de restreindre l'accès aux pages web de l'application aux utilisateurs authentifiés. Nous avons donc appliqué cette technique pour sécuriser l'accès de notre application. Dans un premier temps, nous avons enrichi la base de données existante (cf. figure 6.17) pour gérer les utilisateurs autorisés ainsi que leurs profils associés.

FIGURE 6.17 : SCHEMA DE LA BASE DE DONNEES UTILISATEURS

Une table « users » et une table « profiles » ont été ajoutées à la base de données. La première stocke l'identifiant de l'utilisateur, son nom, son mot de passe ainsi que la clé étrangère « profile_id ». Cette dernière fait le lien avec la seconde table. La table « profiles » enregistre les différents profils d'utilisateurs et indique si c'est un profil d'administration. Cette notion d'administration intervient pour limiter l'accès des pages de simulateur et de gestion des utilisateurs. L'accès aux différentes pages de l'application GreenLAB LabMonitor est désormais restreint aux seuls utilisateurs authentifiés. Mais il existe d'autres failles de sécurité dont peut être victime une application web que nous allons détailler ci-après.

Blocage de l'indexation par les moteurs de recherche

Les moteurs de recherche utilisent des robots d'indexation pour explorer quotidiennement le web. Ces derniers collectent des ressources comme les pages web, les images, les vidéos, les documents… Les moteurs de recherches indexent ensuite ce contenu. Afin de ne pas mettre à disposition le contenu de l'application web LabMonitor pour raison de confidentialité, nous avons mis en place le blocage de ces robots d'indexation. La première action de la plupart des robots d'indexation est de vérifier la présence d'un fichier robots.txt à la racine du site. Ce fichier décrit les autorisations d'accès du robot. Nous avons donc ajouté le fichier suivant représenté par la figure 6.18, afin d'en interdire l'accès.

```
# robots.txt file to prohibit indexation
User-agent: *
Disallow: /
```

FIGURE 6.18 : CONTENU DU FICHIER ROBOTS.TXT

La première ligne non commentée du fichier indique à quel robot s'adresse la restriction d'indexation. Dans notre cas, nous utilisons l'étoile (*) afin de cibler tous les moteurs de recherches respectant ce standard. La seconde ligne non commentée peut éventuellement affiner la restriction d'indexation. Mais nous avons choisi d'interdire totalement l'indexation de l'arborescence du site grâce à la barre oblique (/).

Protection contre les injections de code

Les injections SQL sont une des failles de sécurité d'une application web. Cette faille autorise une personne malveillante à modifier les requêtes envoyées à la base de données et compromet ainsi la sécurité et le fonctionnement de l'application. Les conséquences sont les suivantes :

- Les restrictions d'accès aux pages ne sont plus applicables,
- Les données stockées sont lues par la personne malveillante,
- La base de données est corrompue et l'application ne s'exécute plus.

Les injections SQL sont utilisées majoritairement lorsque l'application web implique une intervention utilisateur comme un champ de saisie d'utilisateur ou de mot de passe.

Pour se protéger de ce type de faille, on est dans l'obligation d'échapper les caractères spéciaux de chaque entrée utilisateur. Ainsi lors de l'utilisation de ces dernières nous appliquons systématiquement la méthode « mysql_real_escape_string($variable_a_proteger) » pour protéger les requêtes SQL utilisant ces entrées comme paramètres.

De même que les injections SQL, il existe également la faille de sécurité « Cross-Site Scripting ». Cette dernière vise à modifier le comportement de l'application web en injectant cette fois-ci du code directement dans la page HTML chargée. Les risques connus sont :

- Redirection de l'utilisateur vers une application web malveillante pour récupérer des informations personnelles,
- Vol de l'identifiant et du mot de passe de l'utilisateur, …

Ces différentes sécurisations de l'application web LabMonitor permettent de protéger son fonctionnement ainsi que ses utilisateurs.

6.5.4 Déploiement dans l'informatique dématérialisée

L'informatique dématérialisée permet d'accéder à des ressources informatiques mises à disposition par un fournisseur. C'est un service à la demande et en libre-service qui fournit au client des ressources informatiques partagées et configurables. On distingue quatre familles de services :

- Logiciel en tant que service (« Software as a Service », SaaS) qui met à disposition d'utilisateurs, des logiciels hébergés à distance. On peut citer comme exemple : Google Apps, Office Web Apps ou encore LotusLive,

- Données en tant que service (« Data as a Service », DaaS) qui permet au client d'externaliser le stockage de ses données,

- Plate-forme en tant que service (« Platform as a Service », PaaS) qui met rapidement à disposition un environnement d'exécution aux utilisateurs,

- Infrastructure en tant que service (« Infrastructure as a Service », IaaS) qui fournit des accès à distance aux infrastructures informatiques situées physiquement chez le fournisseur tels que des serveurs.

Il existe également trois modèles distincts de déploiement d'informatique dématérialisée :

- Nuages privés internes : ce sont des ensembles de ressources informatiques administrées en interne par une entreprise pour ses propres besoins,

- Nuages privés externes : ce modèle de déploiement est également destiné aux besoins d'une entreprise, mais sa gestion est externalisée chez un prestataire,

- Nuages publics : ces ressources informatiques sont gérées par des entreprises spécialisées qui louent leurs services.

Pour nous familiariser avec la notion d'informatique dématérialisée, nous avons choisi le fournisseur de nuages privés externes Amazon recommandé par nos collègues indiens et nous nous sommes orientés vers la famille de service : plate-forme en tant que service. Une unique instance de type « t1.micro » a été créée. L'utilisation de cette instance est gratuite lors de la première année

d'inscription aux services Amazon. Il s'agit d'un serveur virtuel Linux. Le forfait choisi permet d'utiliser l'instance 750 heures par mois et de disposer de 30 giga octets de stockage. Le déploiement réel de l'application LabMonitor nécessiterait au minimum la création de 3 instances distinctes, mais cela implique un coût financier non négligeable. Nous nous contentons donc de tester le déploiement multiserveur de l'application LabMonitor grâce à une configuration matérielle locale. Cette configuration dématérialisée gratuite est pour le moment suffisante pour l'utilisation de l'application lors de démonstrations aux clients potentiels.

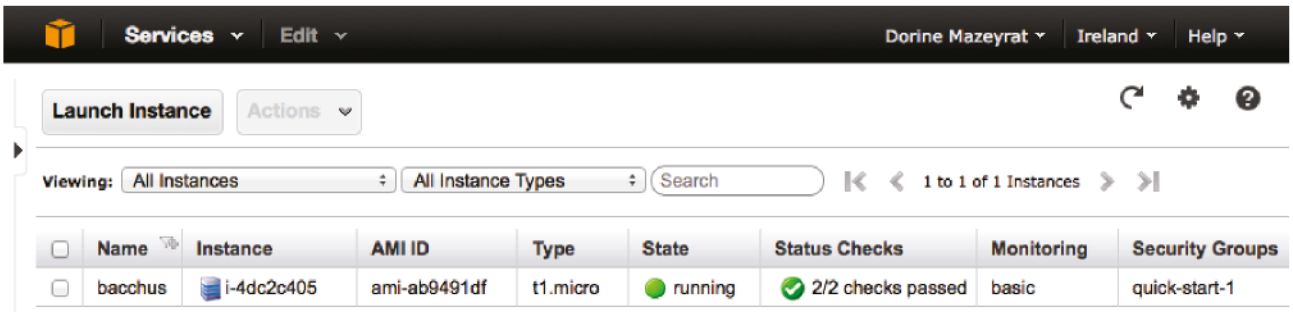


FIGURE 6.19 : INSTANCE AMAZON HEBERGEE EN IRLANDE

Les différents serveurs impliqués lors du déploiement local au sein des locaux de Persistent Systems utilisent le système d'exploitation Windows. Pour mettre en évidence l'indépendance de l'application LabMonitor vis-à-vis du système d'exploitation, nous avons donc opté pour une instance serveur Linux, système d'exploitation que nous n'avons jamais employé jusqu'à présent.

La particularité des serveurs Linux est de ne pas proposer d'interface graphique pour les interactions avec l'utilisateur. Sans cette interface qui est très gourmande en ressources, nous pouvons donc profiter au maximum des capacités matérielles des serveurs. Pour interagir avec le serveur distant, nous utilisons un terminal ainsi que le protocole de communication « Secure SHell » (SSH) (cf. figure 6.20).

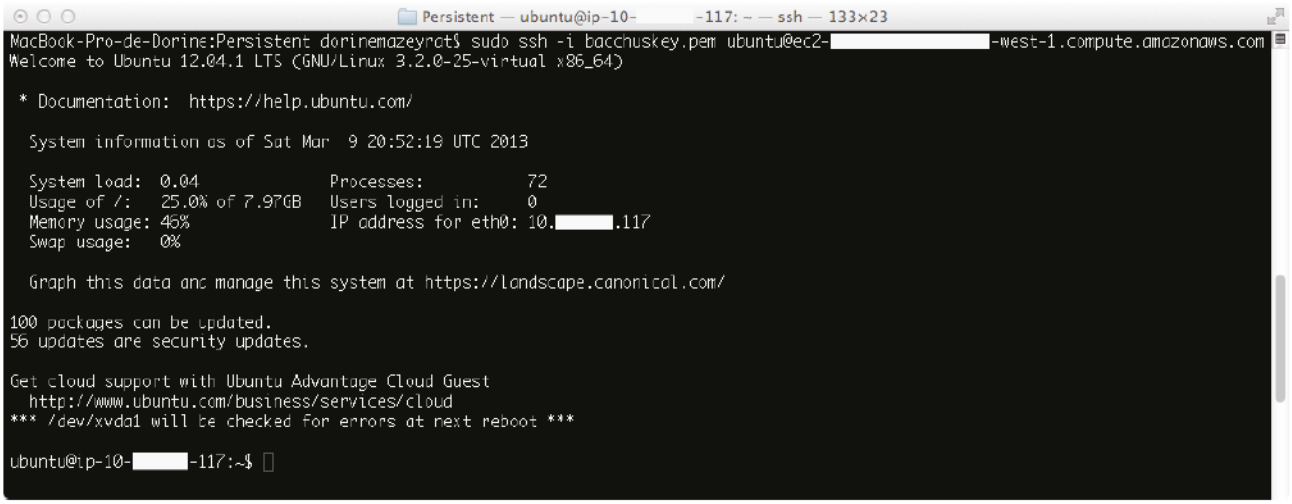


FIGURE 6.20 : TERMINAL UNIX DE L'INSTANCE AMAZON VIA SSH

La préparation au déploiement de l'application LabMonitor sur l'instance dématérialisée requiert tout d'abord l'installation des composants essentiels à l'exécution de chaque couche logicielle :

- Le SGBD MySQL pour le stockage des données,
- Le serveur web Apache et l'interpréteur de script PHP pour la gestion des pages web,
- L'environnement d'exécution Java pour le lancement du serveur d'application responsable des méthodes métier de l'application.

Des manipulations concernant la configuration de l'instance sont également requises pour le bon fonctionnement du logiciel. L'interface de gestion du compte Amazon nous a permis d'associer une adresse IP dite élastique à l'instance. Cette IP élastique est une adresse publique que nous avons explicitement réservée pour l'accès distant à notre serveur. Cette manipulation simplifie le déploiement. En effet, cette IP peut être associée, grâce à l'interface de gestion, à toute instance du compte Amazon. Nous nous prémunissons donc de tout problème lié aux changements de déploiement. Quels que soient ces derniers, l'application LabMonitor est toujours disponible pour le service commercial à la même adresse. En plus du pare-feu configuré par défaut par le système d'exploitation Linux, chaque instance est également protégée par le pare-feu général d'Amazon en amont. Il faut donc autoriser explicitement l'ouverture des ports « Transmission Control Protocol » (TCP) pour permettre les connexions SSH (port 22) ainsi que les échanges avec le serveur web HTTP (port 80).

Dès lors que ces prérequis ont été respectés, nous avons déployé et mis à disposition l'application LabMonitor à l'extérieur du réseau de l'entreprise, c'est-à-dire sur le réseau Internet.

Notre utilisation des services Amazon ainsi que de sa console de gestion nous a permis d'acquérir des connaissances sur l'informatique dématérialisée proposée par ce fournisseur. Nous avons donc rédigé une documentation sur l'utilisation du service « Elastic Cloud Compute » (EC2). Ce document a été mis à disposition des différentes équipes de Persistent Systems France. Elle permettra aux futurs utilisateurs du service de prendre en main plus rapidement les services fournis par Amazon.

6.5.5 Simulateur Web

Le déploiement de l'application LabMonitor s'est effectué vers une instance dématérialisée qui est un serveur Linux. Notre premier simulateur a été développé avec le langage C# qui est compatible uniquement avec le système d'exploitation Windows. Nous avons donc créé un script PHP accessible depuis une page web qui simule des mesures de supervision.

Il fonctionne de façon identique et possède une interface similaire comme le montre la figure 6.21.

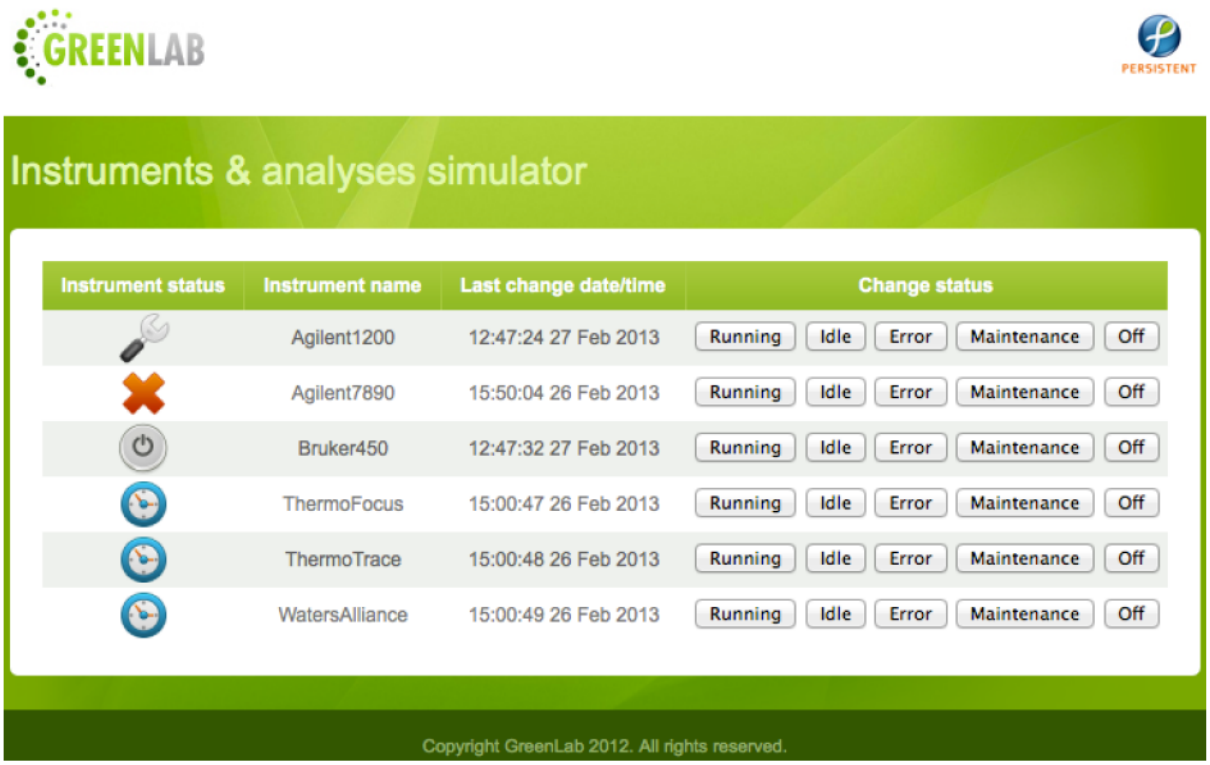


FIGURE 6.21 : SIMULATEUR WEB DE MESURES DE SUPERVISION

Dans un même temps, nous avons également enrichi les statuts existants des instruments analytiques :

- « ok - idle » : l'instrument est connecté mais aucun échantillon n'est en cours d'analyse,
- « ok - running » : l'instrument est connecté et une analyse chimique est en progression,
- « error » : l'instrument est en erreur,
- « warning - offline » : la connexion est impossible avec l'instrument,
- « warning - maintenance » : l'instrument est signalé en maintenance, l'intervention d'un technicien est en cours.

Ce simulateur envoie régulièrement des mesures de supervision pour chaque instrument analytique simulé dans la base de données. Il est, au même titre que l'application web LabMonitor, hébergé par l'instance Amazon.

6.5.6 Rafraichissement des pages web

Pour mettre à jour l'interface web sans intervention de l'utilisateur, il est nécessaire de rafraichir la page. Deux solutions sont envisageables : le rafraichissement automatique via HTML ou l'utilisation d'AJAX.

```
3  <head>
4      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5      <meta http-equiv="refresh" content="5"/>
6      <title>GreenLAB - LabMonitor</title>
7      <link href="css/greenlab.css" rel="stylesheet" type="text/css" />
8      <link href="css/greenlab-labmonitor.css" rel="stylesheet" type="text/css" />
9      <link rel="icon" type="image/png" href="images/favicon.png"/>
10 </head>
```

FIGURE 6.22 : RAFRAICHISSEMENT DES INFORMATIONS VIA HTML

Ces deux techniques ont le même dessein, celui de mettre à jour régulièrement les informations de supervision présentées sur l'interface de l'application web. La première est très rapide à mettre en place et ne nécessite que l'ajout d'une ligne unique dans l'entête de chaque page HTML comme on le constate à la ligne 5 de la figure 6.22. Dans cet exemple, le rafraichissement de la page s'effectue toutes les 5 secondes. Mais cette technique est très gourmande en ressources pour le client comme pour le serveur. En effet, la page rafraichie est rechargée entièrement.

```
97  /**
98  * Refresh instruments table.
99  */
100 function refreshInstrumentsTable(){
101     $('#div-table-instruments').load('getInstrumentsTable.php');
102     setTimeout(refreshInstrumentsTable, 10000);
103 }
```

FIGURE 6.23 : RAFRAICHISSEMENT DES INFORMATIONS VIA AJAX

La seconde solution, illustrée par la figure 6.23, affine le rafraichissement à une plus petite portion de la page. Dans cet exemple, on se contente de recharger uniquement la table contenant les informations relatives aux instruments du laboratoire. Pour cela nous utilisons la solution « Asynchronous JavaScript And XML » (AJAX). Cette technique utilisant le langage JavaScript dynamise la page sans nécessiter un nouvel affichage de celle-ci. En effet, les scripts JavaScript exécutés par le navigateur du client se chargent de demander les informations utiles au serveur et de mettre à jour une partie seulement de la page web.

6.5.7 Bilan

Désormais l'application LabMonitor est disponible sur le web et donc à l'extérieur du réseau interne de l'entreprise. Le simulateur, les bibliothèques métier, les services web et l'interface web

appliquant la nouvelle charte graphique ont été déployés dans l'informatique dématérialisée pour autoriser des démonstrations commerciales.

Une seconde démonstration a été réalisée aux responsables du projet. L'interface web développée actuellement permet d'afficher les écrans utiles au technicien de laboratoire. Mais comme expliqué précédemment, ce dernier se déplace régulièrement au sein du laboratoire. Il doit donc pouvoir consulter en temps réel l'état de santé de ses instruments. L'objectif principal de la prochaine étape vise donc le développement de l'application mobile.

6.6 Troisième livraison

6.6.1 Objectifs

La troisième livraison est la réalisation de l'application mobile cliente pour accéder à l'application LabMonitor. Pour ce faire, il est nécessaire de développer les services Web en charge du traitement des requêtes de l'utilisateur envoyées à partir de son téléphone intelligent. Nous avons développé de nouveaux services web adaptés au terminal mobile. Ensuite pour y accéder de l'extérieur du site, nous les avons déployés sur l'instance Amazon pour concevoir l'application mobile depuis notre poste de travail.

6.6.2 Modifications des services web

La problématique est la suivante : désormais les services web doivent être utilisés par les deux interfaces web et mobile. La différence entre les deux est principalement la taille disponible à l'écran pour l'affichage des informations. L'écran du poste de travail possède en général une diagonale de 15 pouces (= 38,1 cm). A l'inverse, un terminal mobile possède en moyenne un écran d'une diagonale de 3.5 pouces (= 8,89 cm). Les mêmes informations doivent être réorganisées et réagencées pour être affichées de façon optimale sur les écrans des terminaux mobiles.

Ajout de paramètres

Pour adapter les services web aux deux interfaces web et mobile, il est nécessaire d'utiliser des paramètres. Par exemple, lors de la requête concernant la liste des instruments du laboratoire, il est inutile de retourner la liste complète des instruments du laboratoire au terminal mobile. Celui-ci ne pourra pas tous les afficher à l'utilisateur de l'application mobile cliente.

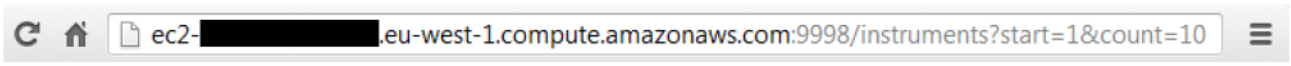


FIGURE 6.24 : PARAMETRES D'UN SERVICE WEB

Comme le montre la figure 6.24, l'ajout des paramètres « start » et « count » à l'URI qui désigne le service web filtre ainsi le nombre de résultats de la requête pour s'adapter au client. La configuration des réponses est primordiale. En effet, elle favorise l'interopérabilité, la réutilisabilité et la résistance aux changements des services web.

Présentation interne des applications web

Une présentation interne aux équipes de Persistent Systems France nous a été demandée afin de présenter quelques retours concernant le développement d'applications web. Le développement web n'a jamais été employé dans l'entreprise lors de la réalisation de projet. Cette présentation s'adresse à tout degré de compétence. Nous avons exposé les différentes technologies utilisées ainsi que les différents termes techniques appropriés à ce type de développement. Puis nous avons présenté les services web, points d'entrées au cœur métier de l'application et plus particulièrement certaines pratiques qui leur sont associées comme leur paramétrage cité ci-dessus.

6.6.3 Déploiement automatique de l'application dans l'informatique dématérialisée

Le premier déploiement de l'application LabMonitor sur l'instance dématérialisée s'est déroulé manuellement par simple copie des différents fichiers et bibliothèques qui composent le logiciel. Afin d'automatiser ce processus et d'éviter d'éventuelles erreurs, nous avons implanté ce processus de déploiement dans la plate-forme d'intégration. Lors de constructions réussies et après validation, le développeur déclenche manuellement via la plate-forme d'intégration continue, le déploiement automatique de l'application dans l'informatique dématérialisée.

Nous avons dû pour cela nous servir d'un greffon de l'outil d'intégration continue Jenkins. Celui-ci, nommé « Publish Over SSH », autorise les actions à distance sur l'instance Amazon via le protocole SSH.

6.6.4 Développement de l'application mobile

Les principales fonctionnalités de l'interface web étant réalisées, il faut maintenant développer l'application mobile qui permet de consulter l'interface de supervision. Comme vu précédemment (cf. 5.6.2), nous nous sommes orientés vers le développement natif.

Prérequis

Les outils requis pour ce type de développement sont :

- Environnement de développement pour le système d'exploitation iOS : XCode. Il n'existe actuellement que cet environnement de développement. L'utilisation de ce dernier est gratuite, seul le déploiement de l'application sur la plate-forme de distribution est payant,

- Poste de travail avec le système d'exploitation OSX d'Apple : un iMac a été mis à disposition du projet par l'entreprise Persistent Systems pendant la durée du développement de l'application. L'environnement de développement XCode fonctionne uniquement avec le système d'exploitation OSX,

- Terminal mobile avec le système d'exploitation iOS : un iPhone 4 avec le système d'exploitation iOS version 6.1 a également mis à disposition afin de tester régulièrement l'application en cours de développement.

Environnement de développement : XCode

Le langage employé pour le développement sur la plate-forme OSX est ObjectiveC. C'est un langage de programmation orienté objet basé sur le langage C. L'API plus particulièrement utilisé pour le développement mobile est Cocoa Touch. La programmation d'application pour la plate-forme iOS est basée sur le patron de conception Modèle-Vue-Contrôleur (MVC).

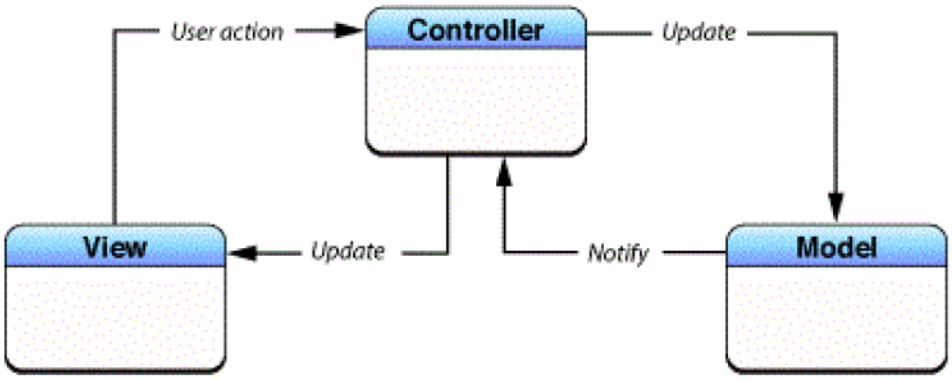


FIGURE 6.25 : PATRON DE CONCEPTION MVC UTILISE PAR COCOA TOUCH

Source : [app13a]

La figure 6.25 illustre le patron de conception MVC et son mode de fonctionnement. Celui-ci sépare les différents composants de l'architecture de l'application cliente mobile. Il regroupe les fonctionnalités dans trois catégories différentes :

- Modèle : elle correspond à la logique métier de l'application,
- Vue : cette catégorie réunit les fonctions relatives à l'interface utilisateur, elle transmet chaque interaction au contrôleur,
- Contrôleur : cette dernière gère les interactions entre les catégories vue et modèle.

Lorsque l'utilisateur interagit avec une vue de l'application, cette action est transmise au contrôleur qui se charge de requérir le traitement demandé au modèle. Lorsque le modèle termine le traitement, il notifie alors le contrôleur qui se charge de mettre à jour la vue de l'application.

Cette utilisation par la programmation iOS diffère de l'organisation habituelle du patron de conception MVC. En effet, dans le patron MVC standard, lorsque le modèle avise le contrôleur du traitement de sa requête, le contrôleur ne fait que notifier la vue pour qu'elle décide de se mettre à jour en questionnant à son tour le modèle.

Vues

La création des différentes vues de l'application mobile s'effectue via le constructeur d'interface ou « interface builder » du logiciel XCode. L'écran d'un terminal mobile étant de dimension limitée, on ne peut afficher toutes les informations relatives à la supervision du laboratoire sur une seule et unique page. Pour la première version de l'application, deux écrans sont créés via le modèle d'interface utilisateur maître-détail ou « master-detail ». Le premier affiche une liste maître, le second écran affiche les détails de l'élément sélectionné dans la liste précédente. Ces deux écrans permettent de présenter respectivement la liste des instruments et le détail des informations relatives à l'instrument sélectionné.

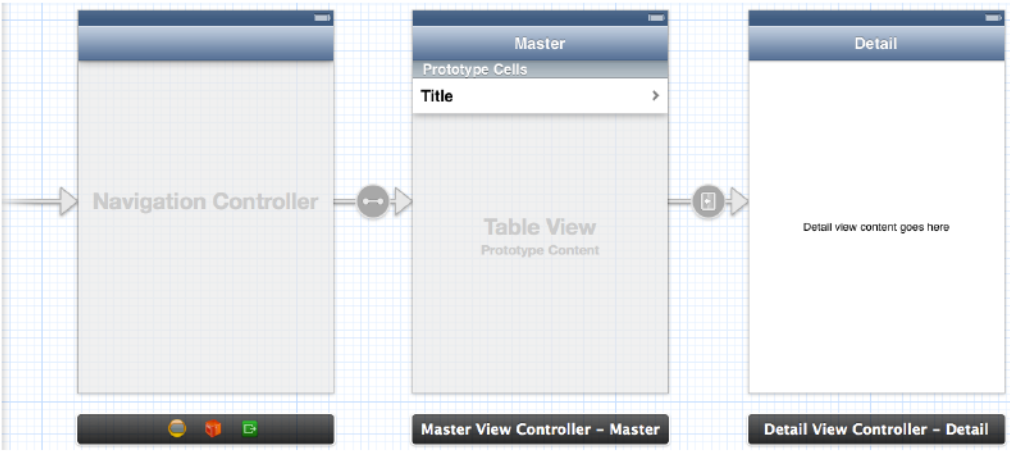


FIGURE 6.26 : MASTER-DETAILS

La figure 6.26 montre le « mainstoryboard » de l'application mobile. C'est l'interface de gestion des différentes vues successives. Il met en évidence l'enchaînement des écrans de l'application LabMonitor. On distingue le contrôleur « Navigation Controller » en charge des deux vues, maître et détail. On remarque aussi les différents liens entre chaque élément.

Le premier, situé à droite du contrôleur représente le point d'entrée de l'application (cf. figure 6.27).

FIGURE 6.27 : INDICATEUR DE LA VUE INITIALE

Le second situé entre le contrôleur et la première vue représente la relation contrôleur-vue (cf. figure 6.28).

FIGURE 6.28 : INDICATEUR DE RELATION CONTROLEUR-VUE

Le dernier, l'indicateur « push » (cf. figure 6.29) représente les transitions entre les différents écrans. On peut utiliser des transitions verticales, horizontales ou personnalisées. Une transition est l'animation qui permet d'enchaîner la disparition de la première vue pour laisser place à la seconde.

FIGURE 6.29 : INDICATEUR DE TRANSITION « PUSH »

Après l'implantation du modèle et du contrôleur, nous obtenons le résultat suivant pour l'application mobile cliente LabMonitor illustré par la figure 6.30.

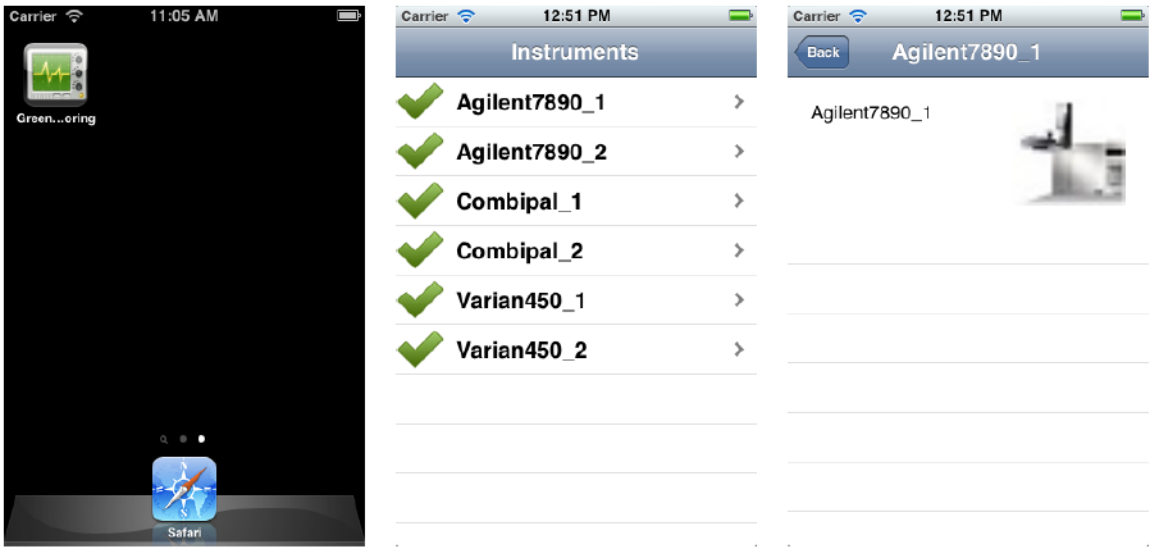


FIGURE 6.30 : PREMIERE VERSION DES ECRANS DE L'APPLICATION MOBILE CLIENTE LABMONITOR

6.6.5 Déploiement de l'application mobile

Le développement de la première version de l'application mobile achevé, nous devons installer celui-ci sur les différents terminaux mobiles du service commercial. Cette installation permettra de démontrer l'application LabMonitor le plus rapidement possible aux clients potentiels. Il existe trois

types de déploiement pour une application iPhone : le programme de développement individuel, celui en entreprise dit standard et le dernier, le programme de développement en grande entreprise comme illustré par le tableau 6.1.

	Licence individuelle	Licence entreprise	Licence grande entreprise
Nom du programme	« iOS developer program - individual »	« iOS developer program - company »	« iOS developer enterprise program »
Type de déploiement	USB ou Ad Hoc	USB ou Ad Hoc	In-House
Nombre de terminaux	100	100	illimité
Prix	99 $ par an	99 $ par an	299 $ par an

TABLEAU 6.1 : TABLEAU COMPARATIF DES PROGRAMMES DE DEVELOPPEMENT APPLE IOS

Source : [app13b]

- Licence individuelle : C'est une licence destinée aux particuliers. Elle permet de tester l'application sur cent terminaux mobiles et de distribuer l'application sur l'AppStore,
- Licence entreprise : Ce programme de développement offre les mêmes caractéristiques que la licence individuelle, mais est réservé aux entreprises,
- Licence grande entreprise : Cette licence autorise de déployer des applications développées par une entreprise uniquement sur les terminaux mobiles possédés par cette même société.

Une licence individuelle ne peut être achetée par une entreprise. Il faut donc se tourner vers les deux licences entreprises. Persistent Systems développe déjà de nombreux projets sur les plates-formes mobiles, dont le système d'exploitation iOS. Des licences sont donc déjà acquises. Ces dernières sont utilisées par des équipes de Pune, Inde. Sur notre demande et après présentation des besoins du projet LabMonitor, une licence grande entreprise nous a été attribuée. Les terminaux mobiles ciblés par le projet en cours sont les iPhones des services commerciaux et n'implique pas la publication de l'application mobile sur la plate-forme de distribution officielle d'Apple. Son utilisation est interne aux employés de la société. Nous avons donc déployé l'application mobile GreenLAB LabMonitor sur les iPhones du service commercial de Persistent Systems.

6.6.6 Bilan

Cette livraison intègre le développement et le déploiement de l'application mobile. Cette dernière communique avec l'instance dématérialisée dans les services Amazon. Tous les éléments de l'architecture sont désormais implantés et toutes les technologies précédemment choisies ont été abordées.

Lors de la démonstration du résultat de cette livraison aux responsables du projet, les exigences du service commercial ont permis d'envisager la prochaine livraison. Une demande a été faite pour

uniformiser la charte graphique de l'application cliente mobile et celle de l'application web, une seconde pour fournir un environnement de démonstration interne à l'entreprise qui met en évidence le lien entre chaque logiciel de GreenLAB.

6.7 Quatrième livraison

6.7.1 Objectifs

Cette quatrième livraison a pour objectif de superviser un laboratoire d'instruments analytiques via le logiciel GreenLAB Workflow. L'équipe Workflow a, de son côté, développé un logiciel de gestion de processus de laboratoire et plus particulièrement le processus d'analyse d'échantillon. Une connexion avec le logiciel de chromatographie OpenLAB CDS EZChrom edition a été établie. L'application Workflow peut ainsi récupérer les statuts des différents instruments analytiques pour planifier les analyses chimiques.

L'arrière-plan de l'application LabMonitor qui concerne le moteur de supervision n'a pas été implanté pendant le stage. Nous avons donc communiqué directement avec le logiciel Workflow pour recouvrer le statut des instruments analytiques afin de satisfaire les besoins du service commercial.

6.7.2 Supervision des instruments du laboratoire d'analyses chimiques

Implantation du service web de l'application Workflow

On observe dans la figure 6.31 l'architecture de l'application Workflow.

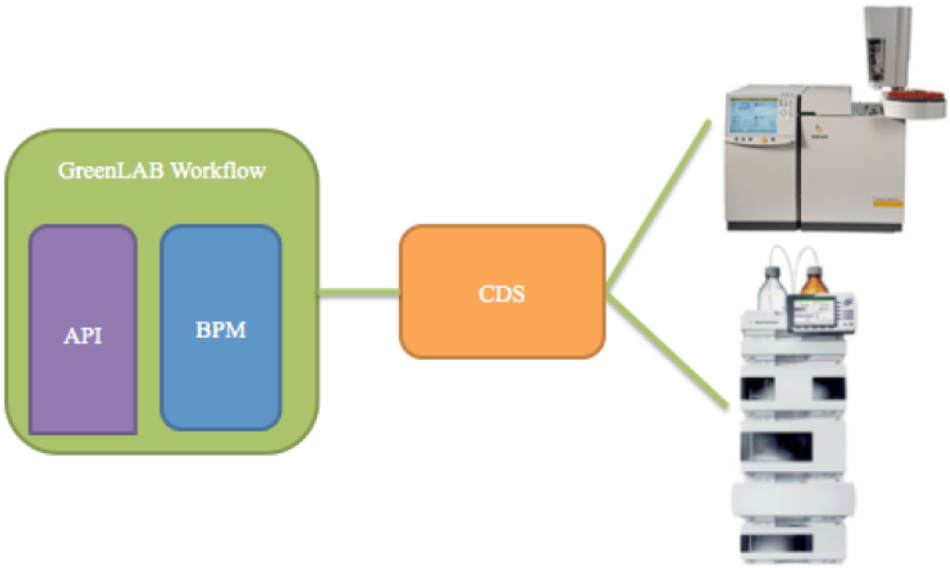


FIGURE 6.31 : ARCHITECTURE DE L'APPLICATION GREENLAB WORKFLOW

Pour interroger l'application GreenLAB Workflow concernant les différents statuts des instruments analytiques, nous avons implanté un service web comme point d'entrée pour le logiciel LabMonitor. Nous avons donc collaboré avec l'équipe Workflow pendant plusieurs jours pour concevoir cette implantation, les différents tests automatiques ainsi que la validation de la fonctionnalité.

En plus du statut de l'instrument supervisé, nous avons souhaité avoir accès aux différentes analyses chimiques en cours sur l'instrument concerné afin de proposer plus d'informations à l'utilisateur de LabMonitor. L'ajout de ces fonctionnalités impacte les différentes interfaces utilisateurs. Il faut désormais les adapter pour afficher les nouvelles informations concernant les analyses chimiques.

Implantation du nouvel écran de l'application web LabMonitor

Une nouvelle interface que l'on peut visualiser sur la figure 6.32 affiche ainsi les analyses en cours sur l'instrument analytique sélectionné. Ces analyses sont mises à jour régulièrement et correspondent aux analyses chimiques en attente, en cours ou en erreur de traitement par le logiciel de chromatographie.

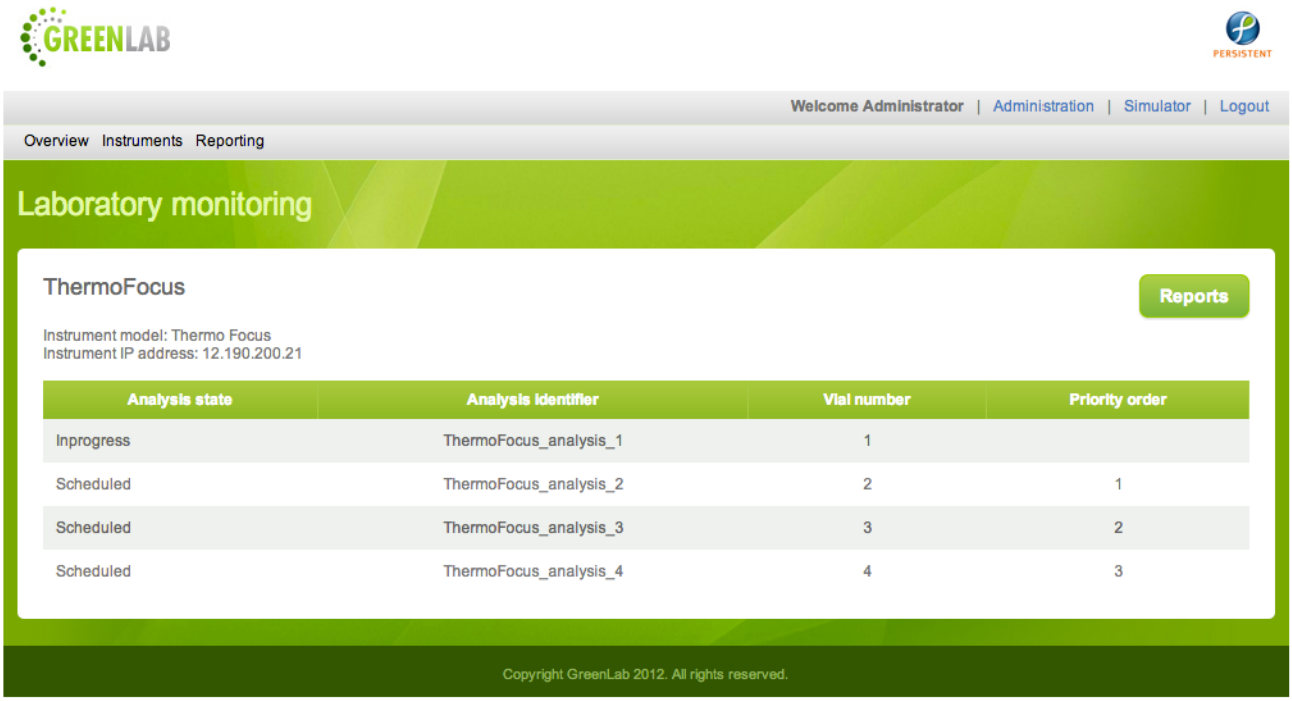


FIGURE 6.32 : PAGE WEB PRESENTANT LES ANALYSES D'UN INSTRUMENT ANALYTIQUE

6.7.3 Amélioration de l'application mobile

Réorganisation des vues

Le nombre d'informations à afficher a considérablement augmenté. Afin d'améliorer la visibilité et l'ergonomie de l'application cliente mobile, nous avons ajouté plusieurs écrans et réorganisé les vues comme on peut le voir dans la figure 6.33.

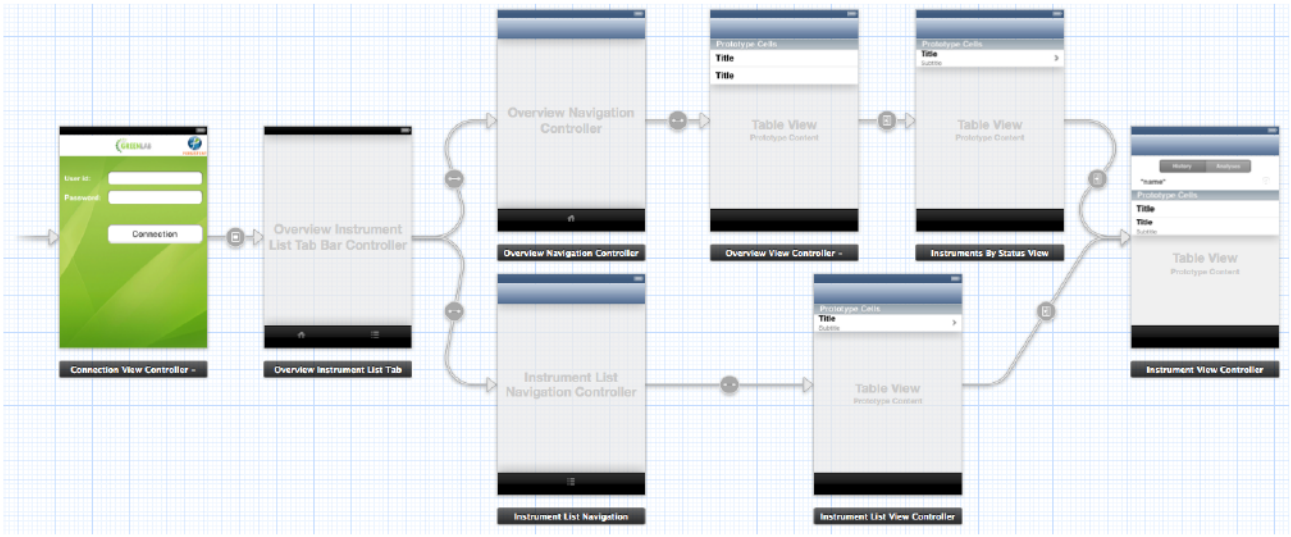


FIGURE 6.33 : SECONDE VERSION DU MAINSTORYBOARD

Nous avons dû tout d'abord, ajouter un écran d'authentification afin d'identifier l'utilisateur. L'écran d'accueil lorsque l'identification est un succès, a également évolué. Pour rappel, la diagonale moyenne d'un écran de terminal mobile est de 3.5 pouces (= 8,89 cm). Bien que jusqu'à présent cette taille d'écran ait été suffisante pour afficher les six instruments analytiques simulés, l'application se doit de supporter des laboratoires de plus grande ampleur.

Un nouvel écran a été implanté comme le montre la figure 6.34. Il regroupe les instruments analytiques par statut et autorise donc l'utilisateur à atteindre directement le ou les instruments qui posent problème. Il est basé également sur le modèle d'interface utilisateur maître-détail.

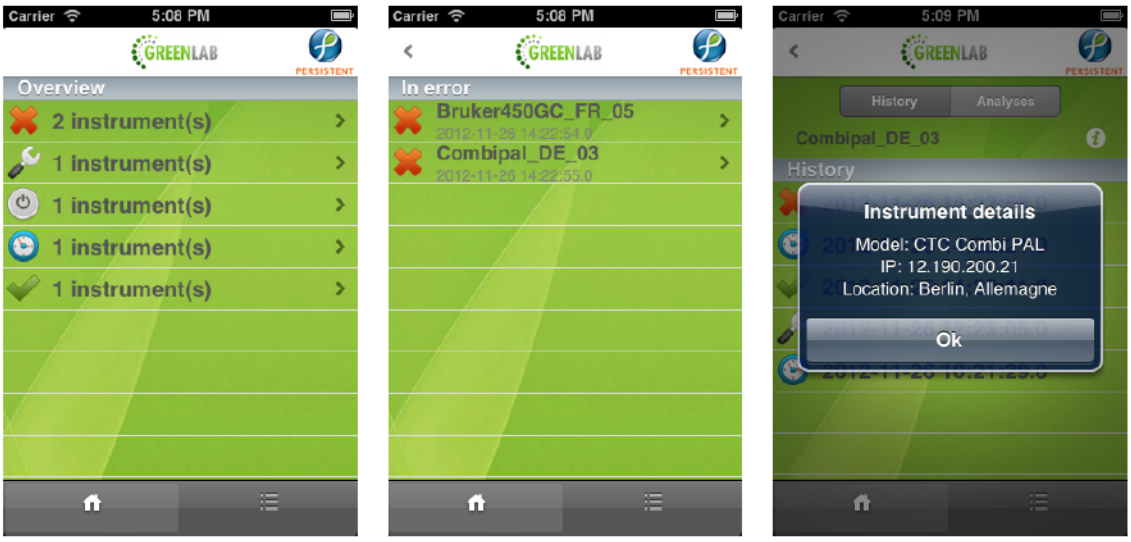


FIGURE 6.34 : ECRANS SUCCESSIFS DE LA SUPERVISION GLOBALE

Pour garder les deux vues implantées comme points d'accès aux instruments, la première expliquée ci-dessus et la seconde qui liste les instruments du laboratoire, nous avons utilisé un second modèle d'interface utilisateur : les onglets. Ces derniers, situés au bas de l'écran de l'application (cf. figure 6.34), permettent de passer successivement d'un mode de présentation à un autre.

Implantation de la charte graphique

Le type de développement choisi pour l'application mobile ne tolère pas la réutilisation des feuilles de styles CSS générées par les équipes indiennes de Persistent Systems. Les changements de la charte graphique ont dû être appliqués à chaque écran afin d'intégrer les logos GreenLAB, Persistent Systems ainsi que le fond d'écran. On peut visualiser le résultat de l'implantation dans la figure 6.35.

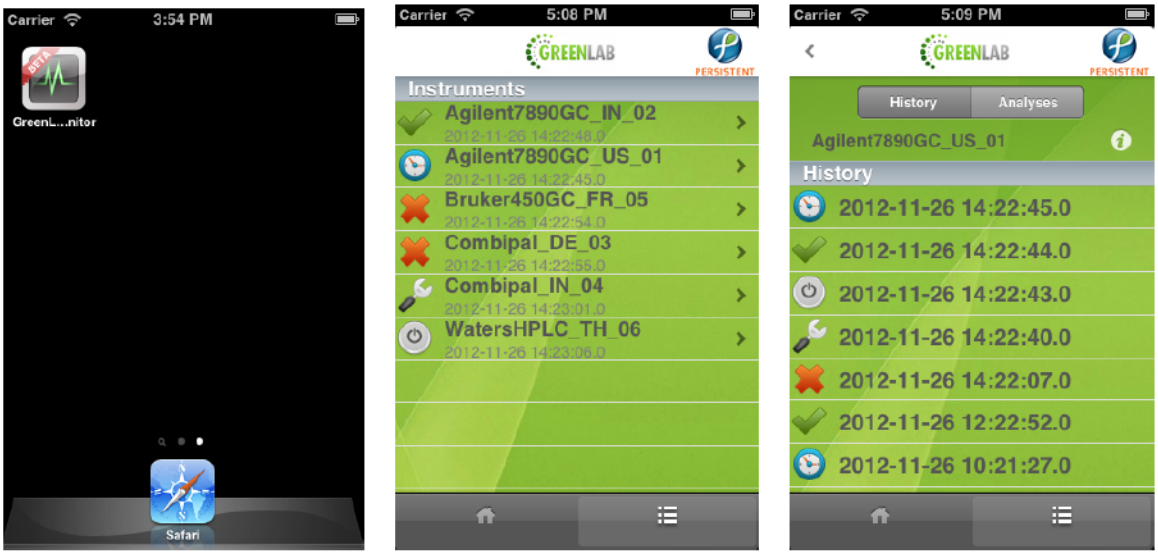


FIGURE 6.35 : ECRANS SUCCESSIFS DE LA SUPERVISION DETAILLEE

Ajout des analyses chimiques liées à l'instrument

La vue détaillée de l'instrument analytique a également été enrichie avec les analyses associées à ce dernier. Le même système d'onglets a été implanté, mais cette fois-ci, le changement d'onglet n'implique pas un changement de vue, mais uniquement un changement de données affichées dans le tableau représenté dans la figure 6.36.

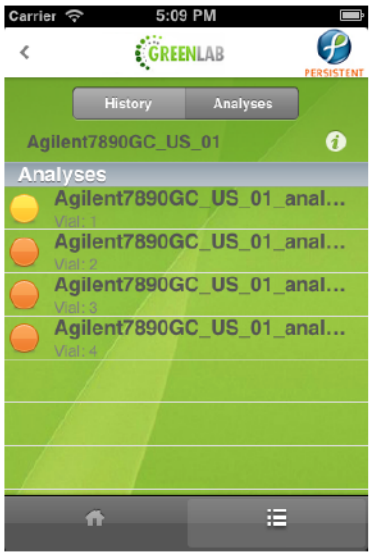


FIGURE 6.36 : ECRAN DU DETAIL DE L'INSTRUMENT SUPERVISE

Configuration de l'application mobile cliente

Pour une meilleure résistance aux changements susceptibles de se produire quant au futur déploiement de l'application web LabMonitor, nous avons défini dans l'application cliente mobile un écran de configuration accessible depuis la configuration générale du téléphone. Cette interface autorise la configuration de l'adresse du serveur hébergeant l'application LabMonitor ainsi que le port d'accès (cf. figure 6.37). On peut également envisager de renseigner l'adresse du serveur de répartiteur de charge matériel ou logiciel qui redirigera la requête vers un serveur d'application.

FIGURE 6.37 : CONFIGURATION DE L'APPLICATION MOBILE LABMONITOR

6.7.4 Réalisation des rapports destinés au responsable du laboratoire

Notre responsable produit a décidé de mettre en avant les rapports de supervision afin d'optimiser les laboratoires. Trois rapports ont été définis pour implémentation :

- Rapport d'utilisation d'un instrument sur les 30 derniers jours,
- Rapport d'utilisation moyenne quotidienne d'un instrument sur les 30 derniers jours,
- Rapport d'utilisation de chaque instrument du laboratoire sur les 30 derniers jours.

Nous avons utilisé la bibliothèque JavaScript D3 qui présente graphiquement les données calculées en amont par l'application LabMonitor.

Rapport d'utilisation d'un instrument sur les 30 derniers jours

L'objectif de ce premier rapport est de rendre manifeste le pourcentage d'utilisation de l'instrument visé. Les spécifications requises pour ce développement sont :

- Type de diagramme : circulaire,
- Période : 30 derniers jours,
- Autres : afficher le pourcentage sur chaque tranche du diagramme et sa légende.

Le responsable du laboratoire doit pouvoir consulter un diagramme circulaire indiquant en pourcentage la distribution des statuts instrument sur les 30 derniers jours d'utilisation. Nous devons donc calculer préalablement chaque distribution en pourcentage des différents statuts de l'instrument : offline, maintenance, error, idle, running. Un module Java a été réalisé pour calculer ces différentes répartitions. Nous pouvons constater le résultat dans la figure 6.38 ci-dessous.

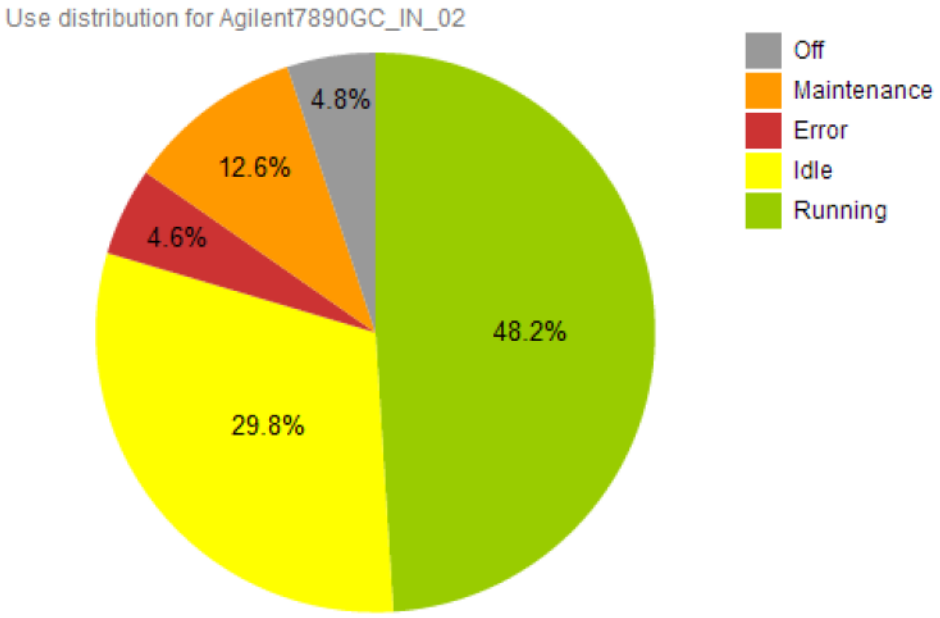


FIGURE 6.38 : REPARTITION DES STATUTS D'UN INSTRUMENT SUR LES 30 DERNIERS JOURS

Grâce à cette représentation graphique, le responsable de laboratoire peut répondre aux questions suivantes :

- Est-ce que l'instrument est sous ou sur utilisé ?
- Est-ce que l'instrument est souvent en panne ?

Ainsi pour l'optimisation de l'utilisation de l'instrument analytique, des alternatives peuvent être envisagées afin d'améliorer la productivité du laboratoire.

Rapport d'utilisation horaire d'un instrument moyennée sur les 30 derniers jours

Ce second graphique vise toujours l'optimisation d'un instrument analytique en particulier. Il permet de visualiser les tranches horaires pendant lesquelles l'instrument est peu ou pas utilisé. Les spécifications requises par le responsable du produit sont :

- Type de diagramme : linéaire,
- Période : 30 derniers jours.

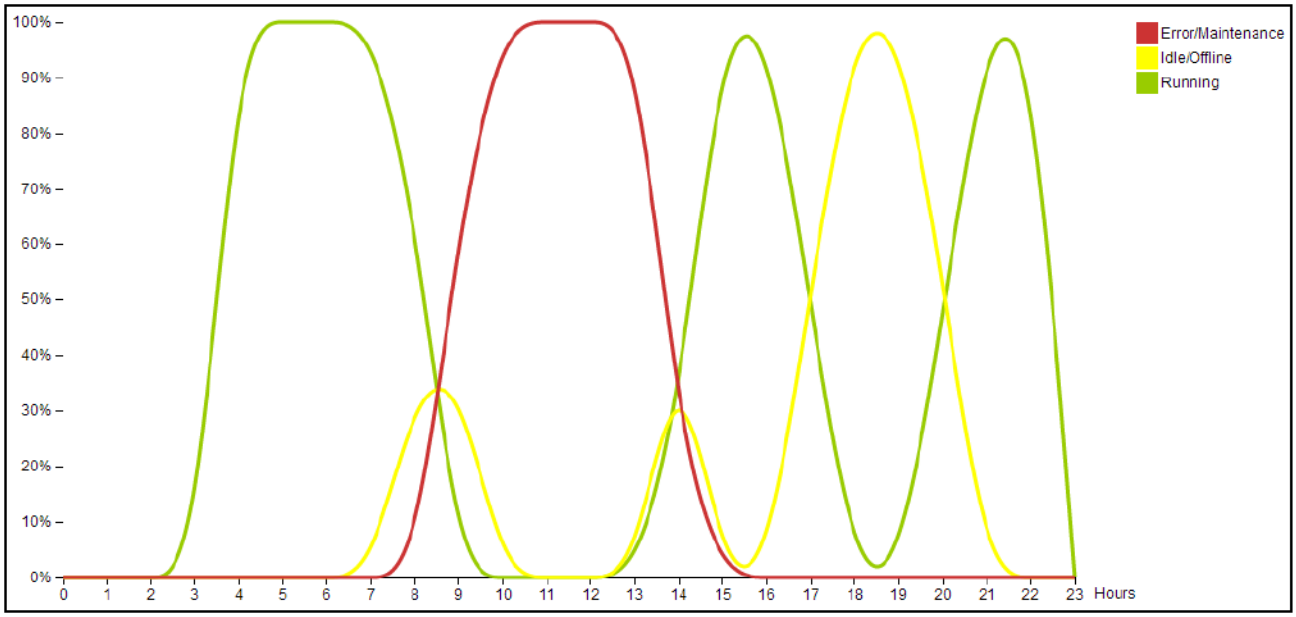


FIGURE 6.39 : RAPPORT D'UTILISATION HORAIRE D'UN INSTRUMENT MOYENNE SUR LES 30 DERNIERS JOURS

Cette courbe graphique des mesures de supervision identifie les périodes de temps d'inactivité ou d'arrêt de l'instrument analytique. Sur la figure 6.39 on constate une période d'inactivité de l'instrument entre 18h et 20h (courbe jaune) sur les 30 derniers jours d'utilisation. De même, on constate que l'instrument est constamment en erreur ou en maintenance entre 9h et 14h (courbe rouge).

Rapport d'utilisation de chaque instrument du laboratoire sur les 30 derniers jours

Ce rapport est similaire au premier réalisé. Il ne fait que regrouper la distribution du statut en pourcentage de tous les instruments du laboratoire. Il favorise ainsi la visibilité du responsable du laboratoire. Les spécifications sont identiques : pourcentage de distribution de chaque statut sur une période de 30 jours. Mais pour plus de visibilité, le type de diagramme utilisé est l'histogramme.

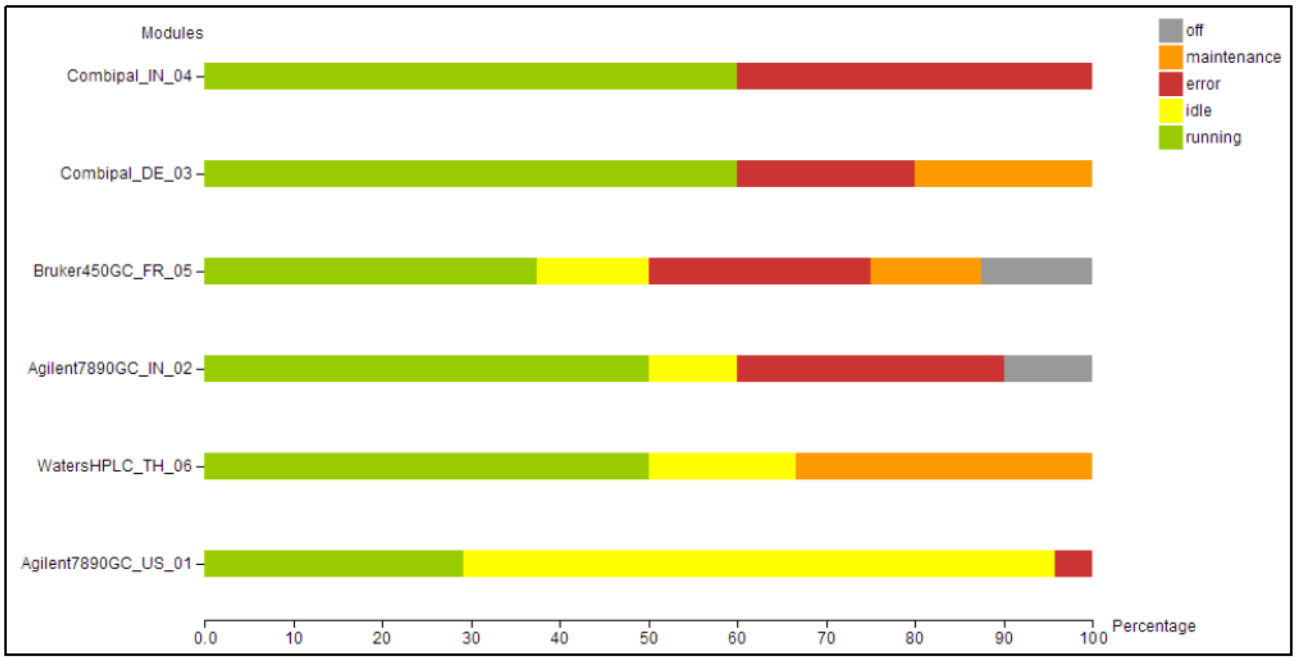


FIGURE 6.40 : REPARTITION DES STATUTS DE TOUS LES INSTRUMENTS DU LABORATOIRE

Chaque instrument listé dans ce graphique (cf. figure 6.40) peut ainsi être facilement comparé avec les autres éléments du laboratoire. Le responsable du laboratoire peut ainsi répondre aux interrogations :

- Quels sont les instruments les plus souvent en erreur ?
- Quel modèle d'instrument est plus performant ?
- Peut-on réorienter des analyses chimiques sur un instrument sous-utilisé ?

6.7.5 Bilan

Cette dernière livraison implante les fonctionnalités relatives au rôle du responsable de laboratoire pour optimiser son environnement de production. Ainsi, les principales fonctionnalités relatives aux deux rôles utilisateurs ciblés en début de projet sont réalisées. L'échange d'information avec le projet Workflow offre également l'opportunité au service commercial de présenter l'ensemble des logiciels GreenLAB de façon cohérente.

Les quatre étapes de réalisation de ce stage nous ont permis d'acquérir de l'expérience en ce qui concerne les techniques et les technologies relatives aux applications web et mobile. Ainsi, nous avons pu produire des estimations de fonctionnalités concernant la suite du développement du projet. Ces dernières permettent de se rendre compte des coûts et des risques pour la mise en production de l'application LabMonitor.

6.8 Passage de la preuve de concept au produit

6.8.1 Risques

Les technologies étudiées et employées ensuite ne présentent pas de risques réels. En effet, ce sont des technologies déjà éprouvées dans le milieu web depuis plusieurs années. L'architecture logicielle n tiers conçue pour l'application limite également les risques. En effet, cette architecture en couche isole chacun de ses composants. L'utilisation de protocoles de communication standards diminue également le couplage entre chaque couche. Ainsi, le changement d'un composant pouvant poser problème ne présente pas réellement de risque, car les impacts sont limités et maitrisés par les choix techniques.

6.8.2 Mise en production

Lors de ce stage, nous avons réalisé l'architecture frontale d'une solution de supervision destinée au domaine de l'analyse chimique. Pour envisager le passage à la mise en production de cette application, nous devons recenser les travaux restants.

Utilisation d'un moteur de supervision

Les différentes études de conception se sont focalisées sur la réalisation de l'architecture frontale de la solution de supervision LabMonitor. Pour obtenir un logiciel entièrement fonctionnel, l'outil de supervision existant qui sera choisi devra être intégré. Grâce à l'architecture logicielle en couches choisie ainsi qu'aux interfaces implémentées, le changement du schéma de la base de données pour celui de l'outil existant ne présente pas de réelle difficulté et implique des changements uniquement sur le module d'accès à la base de données.

Une étude plus approfondie des quatre outils OpenNMS, Nagios, Zabbix et Zenoss sera inévitable pour mieux cerner le fonctionnement de chacun ainsi que le schéma de leur base de données respective.

Implantation de greffons du moteur de supervision

Lorsque l'outil existant sera déterminé, il sera nécessaire d'implanter les divers greffons pour superviser les différents instruments analytiques. Le premier développement sera coûteux en raison du temps d'apprentissage vis-à-vis de l'instrument et du logiciel métier mis en place dans le laboratoire. Mais les suivants relativement similaires ne présenteront plus de difficultés.

Fonctionnalités supplémentaires

D'autres fonctionnalités seront essentielles pour rendre l'application LabMonitor utilisable au sein d'un laboratoire d'analyses chimiques. Parmi elles, on peut énumérer :

- Une mise en place de sécurités supplémentaires comme le chiffrement des communications ou un système d'authentification plus performant,

- L'utilisation d'un cadriciel de développement pour la couche de présentation de l'application. Cet outil de travail offre de nombreuses solutions au développeur ce qui lui permet de se focaliser sur les besoins du client,

- L'ajout de nouveaux rapports statistiques concernant l'utilisation des différents instruments analytiques. Il faut également étendre l'intervalle de temps utilisé pour les calculs pour fournir ainsi des informations plus complètes et détaillées au responsable du laboratoire,

- Le développement de nouvelles fonctionnalités concernant l'application mobile client. En effet, actuellement le technicien doit consulter l'interface de son terminal mobile pour constater une erreur sur un instrument analytique. L'implantation des notifications libérerait l'utilisateur de cette contrainte. Ainsi lors d'un problème au sein du laboratoire, un signal sonore retentirait pour alerter le technicien.

Chapitre 7 Conclusion

7.1 Rappel des objectifs

Dans le cadre du développement de la suite logicielle du projet GreenLAB visant à améliorer la performance d'un laboratoire d'analyses chimiques, la supervision est un point capital et déterminant quant à l'optimisation de l'utilisation des instruments métier. L'objectif du stage était de concevoir la première version de l'application LabMonitor, une solution de supervision destinée aux instruments analytiques. Elle informe les techniciens du laboratoire des états de fonctionnement en temps réel des instruments grâce aux terminaux mobiles. Elle offre également des outils d'analyses statistiques concernant l'utilisation des instruments d'analyses chimiques utiles aux responsables du laboratoire en vue d'optimiser l'activité métier. La solution de supervision devait répondre aux exigences du domaine, à savoir adaptable aux différents environnements, interface ergonomique et conviviale disponible sur poste de travail et iPhone ainsi que différentes contraintes comme la robustesse, la capacité d'évolution ou la sécurité.

Les défis représentés par ce projet sont multiples. Dans un premier temps, nous n'avions jamais été impliqués dans le choix architectural d'une application. Ensuite, les technologies et langages que nous avons étudiés, sélectionnés puis utilisés n'avaient jamais été employés auparavant par Persistent Systems France. Les systèmes d'exploitation des serveurs et des terminaux mobiles n'avaient encore jamais été utilisés. De même, la notion d'informatique dématérialisée nous était inconnue. Nous n'avions également jamais abordé le domaine de la supervision.

7.2 Bilan du stage

En premier lieu, nous avons conçu l'architecture logicielle de l'application LabMonitor. Pour cela nous avons réalisé une étude préliminaire d'outils de supervision existants afin d'identifier une architecture globale à ce type de solution : l'architecture logicielle en couches. Ensuite, nous avons comparé les différents types d'architecture pour aboutir sur celle qui a été utilisée pour le développement de l'application LabMonitor : l'architecture n tiers. Cette dernière offre les avantages de découplage, d'extensibilité, de réutilisabilité et de robustesse permettant de respecter les contraintes du projet fixées en amont.

Les outils de supervision destinés plus particulièrement au domaine de l'infrastructure informatique d'une entreprise procurent des possibilités d'adaptation à des environnements différents. Ainsi, les implantations du projet LabMonitor se sont orientées plus particulièrement vers l'architecture frontale d'une solution de supervision basée sur un outil existant chargé de la gestion de l'arrière-plan de l'application. Nous avons ainsi identifié les différents modules nécessaires à la réalisation du projet LabMonitor : les différentes couches logicielles ainsi que les interfaces pour favoriser le découplage. Cela nous a permis de recenser explicitement le rôle ainsi que les besoins et contraintes de chacun pour amorcer les choix technologiques.

La deuxième étape a consisté à déterminer les technologies pour le développement de chaque module de l'architecture logicielle frontale précédemment choisie. Pour cela, nous avons comparé différents langages et techniques afin de déterminer les plus appropriés et de les appliquer à la création des modules responsables des données, des méthodes métier et des interfaces utilisateurs accessibles par le navigateur Internet ou l'application mobile cliente installée sur terminal mobile. Cela nous a permis de faire le point sur les nombreuses technologies existantes dans le monde du web et de la mobilité. Une fois les technologies définies tels que le langage Java pour l'implantation des modules métier, le langage PHP pour la réalisation de l'interface web et le développement natif sous le système d'exploitation iOS pour la création de l'application mobile cliente, nous avons alors préparé les différents environnements et machines de développement pour commencer l'implantation des divers composants de l'architecture logicielle du projet.

Enfin, nous avons implanté et déployé les divers modules composant l'application LabMonitor grâce aux langages et techniques sélectionnés précédemment. Pour cela, nous avons réalisé une application web ainsi qu'une application mobile cliente destinée à la plate-forme iOS d'Apple. Pour nous familiariser avec la technique et satisfaire aux exigences de démonstration commerciale, nous avons également déployé ces applications respectivement sur une instance des services d'informatique dématérialisée fournis par Amazon et sur les terminaux mobiles du service commercial de Persistent Systems. Les différentes interfaces réalisées, destinées aux responsables et techniciens du laboratoire d'analyses chimiques, ont été adaptées à leur domaine métier.

Ces diverses études et réalisations ont fourni une preuve de concept du logiciel LabMonitor. En effet, cette démonstration de faisabilité ainsi que les estimations quant aux travaux nécessaires à la mise en production de l'application fournissent de solides bases aux services commerciaux pour leur permettre d'aborder les clients potentiels en leur présentant une version limitée, mais fonctionnelle du programme. Ainsi, cinq démonstrations commerciales ont pu être réalisées auprès de clients intéressés situés en Europe et aux États-Unis.

7.3 Avenir du projet

L'application LabMonitor, élaborée durant ces neuf mois de stage de stage au sein de Persistent Systems France, propose un ensemble de fonctionnalités concernant la supervision des instruments analytiques. L'outil BPM de la suite logicielle GreenLAB pourra donc se servir de ces fonctionnalités pour connaître et utiliser le statut courant des instruments en vue d'automatiser les activités métier du laboratoire d'analyses chimiques.

En fin de stage, différentes démonstrations commerciales ont été effectuées au cours du mois de mars 2013. Les clients impliqués situés en Europe et aux Etats-Unis se sont montrés réellement intéressés et nous avons eu des retours très positifs. Malgré cela, nous sommes encore en attente de partenaires financiers pour approfondir le développement de la suite logicielle GreenLAB.

Fin mars 2013, les différents travaux en cours ont été interrompus, les équipes GreenLAB réorganisées et réorientées vers de nouveaux projets déjà financés. La réalisation de l'ensemble des applications relatives à la suite logicielle GreenLAB fut une expérience très enrichissante pour les équipes impliquées. Elle a permis une montée en compétence dans les technologies web et mobiles qui n'étaient jusqu'à présent pas employées au sein du site français. Ces nouvelles compétences sont un atout majeur dans le développement logiciel car très demandées.

7.4 Bilan personnel

Ce stage de neuf mois au sein de Persistent Systems France est l'aboutissement de mes cinq années d'études au CNAM de Grenoble. Cette formation m'a permis d'évoluer sur le plan professionnel comme personnel. J'ai abordé les différentes étapes techniques de la réalisation d'une application logicielle expérimentale dans le domaine de l'analyse chimique.

J'ai consolidé mon expérience dans le domaine de l'analyse chimique en m'intéressant de plus près à la définition des besoins et des implications liés aux rôles des responsables et des techniciens de laboratoire. Cette expérience a été pour moi très enrichissante. En effet, pour réaliser les différentes interfaces de supervision qui leur étaient destinées, j'ai découvert et employé de nombreuses technologies web et mobile qui m'étaient jusqu'à présent inconnues.

J'ai réalisé et déployé l'architecture logicielle d'une application web. Dans cette démarche, j'ai été amenée à utiliser différents systèmes d'exploitation : OSX, Linux ou encore iOS alors que mon expérience précédente était restreinte aux technologies Microsoft. Je me suis formée à de nombreux langages de programmation comme Java, JavaScript, PHP ou ObjectiveC ainsi qu'à diverses techniques de développement comme les services web. Les services Amazon dans l'informatique dématérialisée m'ont permis une appropriation de la notion de « cloud computing » et de déployer l'application à l'extérieur de l'entreprise.

La grande liberté qui m'a été accordée quant à l'accomplissement de cette mission, bien que parfois déroutante, m'a enseigné une certaine rigueur et une certaine organisation. Cette expérience a nécessité une grande autonomie dans mon travail. En effet, les différents projets du site grenoblois dans lequel je travaille n'avaient jamais impliqué ces nouvelles technologies. De même, personne dans l'entreprise n'ayant les connaissances nécessaires, j'ai dû chercher moi-même les différentes solutions aux problèmes de développement rencontrés. Ce long travail de recherche et de mise en application de processus nouveaux m'a apporté non seulement une satisfaction dans la réalisation du projet LabMonitor, mais m'a aussi fait acquérir de nombreuses connaissances et m'a ouverte aux multiples technologies dans le monde du web et de la mobilité.

Ce projet m'a également permis de communiquer avec quelques-uns de mes nouveaux collègues indiens lors de conférences téléphoniques ou d'échanges de messages électroniques. J'ai ainsi découvert et apprécié des personnes motivées et très compétentes dans leur domaine respectif.

Ce stage a favorisé une première approche du développement logiciel à destination de terminaux mobiles et plus particulièrement du système d'exploitation iOS d'Apple. Cette expérience est un atout valorisant dans ma vie professionnelle future. En effet, les technologies mobiles, très récentes, offrent de multiples possibilités dans tous les domaines métier et m'intéressent particulièrement.

Glossaire

Cadriciel : plus couramment appelé « framework », le cadriciel est un ensemble organisé de composants logiciels. Il est employé pour créer les bases d'une partie ou de l'ensemble de l'architecture de l'application développée.

Chromatographie : la chromatographie est une méthode d'analyse de séparation d'éléments chimiques. On distingue deux types de chromatographie dans les laboratoires d'analyses chimiques : en phase gazeuse ou en phase liquide.

Couplage : le couplage désigne le niveau d'interaction existant entre plusieurs composants logiciels. Un couplage faible favorise l'évolution d'une application, car lors d'un changement, les impacts sont minimisés.

Déploiement : le déploiement est la mise en place d'une nouvelle application ou d'une nouvelle version d'un programme existant sur les instances matérielles concernées.

Evolutivité : c'est l'aptitude d'une application à évoluer. L'évolutivité est une caractéristique importante non négligeable à prendre en compte lors du développement d'un logiciel.

Extensibilité : l'extensibilité est la capacité d'un logiciel à s'adapter lors d'une montée en charge. Lorsque la demande augmente, l'application doit pouvoir maintenir son activité métier.

Informatique dématérialisée : plus couramment appelée « cloud computing », l'informatique dématérialisée désigne un ensemble de services à distance qui met à disposition des ressources informatiques configurables et partagées.

Interopérabilité : l'interopérabilité est la capacité d'un composant logiciel ou d'une application à pouvoir fonctionner avec d'autres logiciels. Les communications échangées doivent respecter des standards.

Intranet : l'intranet est le réseau interne d'une entreprise basé sur les mêmes protocoles qu'Internet.

Kanban : la méthode Kanban est une méthode de gestion de flux inventée par Toyota en 1950. Désormais appliquée au développement informatique, elle met l'accent sur le travail en cours et limite les gaspillages.

Montée en charge : la montée en charge désigne une augmentation importante et rapide des demandes de traitement à destination d'une application.

Patron de conception : en anglais « design pattern », un patron de conception est une suggestion logicielle, considérée comme une bonne pratique, qui répond à un problème rencontré couramment en conception informatique.

Répartiteur de charge : qu'il soit logiciel ou matériel, un répartiteur de charge a pour objectif de distribuer les traitements demandés entre les différentes applications ou serveurs d'un ensemble organisé.

Réutilisabilité : la réutilisabilité est un principe de développement qui consiste à réutiliser un code source ou un composant.

Serveur d'application : le serveur d'application est un programme qui fournit un environnement d'exécution pour des modules logiciels. Il est employé principalement dans la couche métier d'une application web.

Serveur web : ou serveur HTTP, il désigne le logiciel qui rend accessibles les pages HTML sur le réseau.

Service web : un service web est un composant logiciel dont les fonctionnalités sont exposées sur le réseau Internet ou intranet.

Spectrométrie de masse : la spectrométrie de masse est un procédé physique d'analyse en phase gazeuse qui permet d'identifier des molécules en mesurant leur masse.

Bibliographie

Références

[Gab09] Jean Gabès. *Nagios 3 pour la supervision et la métrologie*. Eyrolles. 2009.

[Jan08] Olivier Jan. *Nagios au cœur de la supervision OpenSource: de l'installation à l'optimisation*. Eni. 2008.

[KBS09] Michael Kresse, Markus Bause, Aurélie Schwaller. *Découvrir ITIL v3 Service Management Pocket Book*. Serview. 2009.

[Maz12] D. Mazeyrat. *Etude de l'architecture logicielle des outils de supervision*. CNAM de Grenoble. Epreuve TEST. 2012.

[PP03] M. Poppendieck et T. Poppendieck. *Lean software development: An agile toolkit*. Addison-Wesley Professional, 2003.

Sites Internet

Tous les sites Internet ont été consultés entre janvier et mars 2013.

[app13a] Apple. *Cocoa competencies*. 2013.
URL : https://developer.apple.com/library/mac/#documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html

[app13b] Apple. *iOS developer program*. 2013.
URL : https://developer.apple.com/programs/ios

[ate13] BYOD : *Pour conquérir l'entreprise, séduire les employés ou le management*. 2013.
URL : http://www.atelier.net/trends/articles/byod-conquerir-entreprise-seduire-employes-management

[cam13] Camille Roux. *Cours sur les RIA et Flex à Polytech'Nice-Sophia*. 2013.
URL : http://www.camilleroux.com/2008/12/21/cours-ria-flex-polytech-nice-sophia/

[car13] Eric Cariou. *Architecture client-serveur*. 2013.
URL : http://web.univ-pau.fr/~ecariou/cours/sd-m1/cours-architecture.pdf

[cav13] Fred Cavazza. *Après les RIA, voici les RDA*. 2013.
URL : http://www.fredcavazza.net/2006/10/15/apres-les-ria-les-rda

[ccm13] Comment ça marche. *Réseaux - client léger*. 2013.
URL : http://www.commentcamarche.net/contents/cs/client-leger.php3

[clu13] Clubic. *Interview de David Axmark*. 2013.
URL : http://www.clubic.com/actualite-84092-david-axmark-mysql-11-installations.html

[cri13] CRISP. *Kanban*. 2013.
URL : http://www.crisp.se/gratis-material-ochguider/kanban.

[dev13] Mario Korf et Eugene Oksman. *Native, HTML5, or hybrid: understanding your mobile application development options*. 2013.
URL : http://wiki.developerforce.com/page/Native,_HTML5,_or_Hybrid:_Understanding_Your_Mobile_Application_Development_Options

[dmt13] Distributed management task force, inc. *Common Information Model*. 2013.
URL : http://dmtf.org/standards/cim

[eaw13] Edward A. Webb. *XML vs JSON*. 2013.
URL : http://www.edwardawebb.com/tips/xml-json

[fda13] Food and drug administration. *Code of federal regulations title 21*. 2013.
URL : http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfcfr/cfrsearch.cfm?cfrpart=11

[gar13] Gartner. *Gartner IT Glossary*. 2013.
URL : http://www.gartner.com/it-glossary

[gee13] Blog d'un directeur technique. *INI vs JSON vs XML*. 2013.
URL : http://www.geek-directeur-technique.com/2011/03/03/ini-vs-json-vs-xml

[gre13] Gronoblog. *Client léger, riche ou lourd ?* 2013.
URL : http://gronoblog.blogspot.fr/2011/01/informatique-client-leger-riche-ou.html

[igm13a] Supervision monitoring. *La supervision*. 2013.
URL : http://igm.univ-mlv.fr/~dr/XPOSE2007/dmichau_supervision/supervision.html

[igm13b] La métrologie des réseaux. *Définitions*. 2013.
URL : http://igm.univ-mlv.fr/~dr/XPOSE2006/PICARD/definitions.htm

[jdn13] Nicolas Ressouches. *HTML5 ou Natif, quelle stratégie adopter pour vos applications mobiles*. 2013.
URL : http://www.journaldunet.com/ebusiness/expert/51796/html5-ou-natif--quelle-strategie-adopter-pour-vos-applications-mobiles.shtml

[KS13] H. Kniberg et M. Skarin. Kanban et Scrum - tirer le meilleur des deux. 2013.
URL : http://henrik-kniberg.developpez.com/mattias-skarin/livre/scrum-kanban/?page=partie1-comparaison.

[laa13] Laboratoire d'analyse et d'architecture des systèmes. *Réflexions sur la terminologie « Surveillance – Supervision »*. 2013.
URL : http://homepages.laas.fr/combacau/SPSF/sursup.html

[lar13] Larousse. *Architecture client-serveur*. 2013.
URL : http://www.larousse.fr/dictionnaires/francais/client-serveur/16523

[mon13] Monitoring-fr. *Qu'est-ce qu'est la supervision ?* 2013.
URL : http://www.monitoring-fr.org/supervision

[msd13a] Miscrosoft developer network. *Vue d'ensemble de la disponibilité*. 2013.
URL : http://msdn.microsoft.com/fr-fr/library/aa291543(v=vs.71).aspx

[msd13b] Microsoft developer network. *Using a three-tier architecture model*. 2013.
URL : http://msdn.microsoft.com/en-us/library/windows/desktop/ms685068.aspx

[mur13] Murphy's laws site. *Murphy's laws*. 2013.
URL : http://www.murphys-laws.com/murphy/murphy-laws.html

[mys13a] MySQL. *About MySQL*. 2013.
URL : http://www.mysql.com/about

[mys13b] MySQL. *Replication*. 2013.
URL : http://dev.mysql.com/doc/refman/5.6/en/replication.html

[mys13c] MySQL. *Cluster*. 2013.
URL : http://dev.mysql.com/doc/refman/5.5/en/mysql-cluster.html

[nag13] Nagios. *Nagios Core Documentation*. 2013.
URL : http://nagios.sourceforge.net/docs/nagioscore/3/en/toc.html

[oct13] Mikael Robert. *Toward a better software factory*. 2013.
URL : http://blog.octo.com/en/toward-a-better-software-factory/

[ope13] OpenNMS. *Official documentation*. 2013.
URL : http://www.opennms.org/wiki/Docu-overview

[ora13] Oracle. *JDBC overview*. 2013.
URL : http://www.oracle.com/technetwork/java/overview-141217.html

[per13a] Persistent Systems. *Cloud computing*. 2013.
URL : http://www.persistentsys.com/Technology/Cloud.aspx

[per13b] Persistent Systems. *Analytics*. 2013.
URL : http://www.persistentsys.com/Technology/Analytics.aspx

[per13c] Persistent Systems. *Collaboration*. 2013.
URL : http://www.persistentsys.com/Technology/Collaboration.aspx

[per13d] Persistent Systems. *Mobility*. 2013.
URL : http://www.persistentsys.com/Technology/Mobility.aspx

[pla13] Play documentation. *Play for Java developers*. 2013.
URL : http://www.playframework.com/documentation/2.1.1/JavaHome

[red13] RedMonk. *The RedMonk programming language rankings: january 2013*. 2013.
URL : http://redmonk.com/sogrady/category/programming-languages

[sea13] Margaret Rouse. *REST (Representational state transfer)*. 2013.
URL : http://searchsoa.techtarget.com/definition/REST

[smb13] SMB111. *Les architectures n tiers*. 2013.
URL : http://vrac.cofares.net/SMB111/composants/Cours%20-%20Architecture%20N-tier.pdf

[smi13] Smile. *Principe de répartition des charges*. 2013.
URL : http://architectures-web.smile.fr/Repartition-de-charge/Principe-de-repartition-de-charge

[sta13] StatCounter. *Top 8 mobile operating systems*. 2013.
URL : http://gs.statcounter.com/#mobile_os-ww-monthly-201203-201303

[sym13] Symfony. *The book*. 2013.
URL : http://symfony.com/doc/current/book/index.html

[tar13] Willy Tarreau. *Making applications scalable with load balancing*. 2013.
URL : http://1wt.eu/articles/2006_lb/index.html

[tec13] Techno-science. *Management information base*. 2013.
URL : http://www.techno-science.net/?onglet=glossaire&definition=10842

[tio13] Tiobe software. *Programming community index for march 2013*. 2013.
URL : http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

[w3c13] W3C. *Web services architecture*. 2013.
URL : http://www.w3.org/TR/ws-arch/#whatis

[wib13] WikiBooks. *Programmation Web*. 2013.
URL : http://fr.wikibooks.org/wiki/Programmation_Web

[zab13] Zabbix. *Zabbix Wiki*. 2013.
URL : https://www.zabbix.com/wiki/doku.php

[zdn13] Vincent Lieffroy. *Lourd, léger, riche : réflexions autour des clients*. 2013.
URL : http://www.zdnet.fr/actualites/lourd-leger-riche-reflexions-autour-des-clients-39601530.htm

[zdn13b] Frédéric Charles. *Blackberry : BB10 en approche pour réconcilier boulot et dodo*. 2013.
URL : http://www.zdnet.fr/actualites/blackberry-bb10-en-approche-pour-reconcilier-boulot-et-dodo-39786633.htm

[zen13] Zenoss. Zenoss community documentation. 2013.
URL : http://community.zenoss.org/community/documentation/official_documentation/zenoss-dev-guide/3.0v01

MEMOIRE D'INGENIEUR C.N.A.M. en INFORMATIQUE

Réalisation d'une plate-forme de supervision d'instruments analytiques dans les laboratoires à distance grâce aux nouveaux outils et technologies

Dorine MAZEYRAT **Grenoble, le 27 mai 2013**

Résumé

Les responsables des laboratoires d'analyses chimiques dans le domaine pharmaceutique ou pétrochimique sont chargés de l'optimisation des utilisations des instruments analytiques. Les solutions de supervision sont des outils indispensables pour améliorer les temps de disponibilité des éléments observés. Pour répondre aux besoins des responsables, Persistent Systems m'a chargée de réaliser une preuve de conception d'une solution de supervision destinée aux instruments analytiques. Après le choix de l'architecture logicielle et des technologies de chacun des modules, une application a été développée puis déployée dans l'informatique dématérialisée. Des services ont été implantés pour favoriser l'interopérabilité du programme et permettre ainsi la création de deux interfaces de supervision distinctes : l'interface principale accessible via le navigateur et l'interface mobile accessible via une application dédiée sur la plate-forme iOS d'Apple. Ces interfaces affichent en temps réel les statuts des instruments du laboratoire. Le technicien alerté en cas de défaillance peut ainsi intervenir rapidement pour résoudre la panne et relancer la production. L'interface principale génère également des rapports statistiques concernant les utilisations des instruments pour aider les responsables dans l'optimisation du laboratoire. Après des démonstrations commerciales du projet, les retours clients ont été très positifs et laissent espérer une suite du projet avec des retombées économiques.

Mots-clés : supervision, instrument analytique, architecture, dématérialisation, déploiement, mobilité, service

---

Abstract

The managers of chemical analysis laboratories in pharmaceutical or petrochemical fields are responsible for the optimization of the analytical instrument utilization. The monitoring solutions are essential tools to improve the uptime of monitored elements. To meet the managers' needs, Persistent Systems asked me to realize a proof of concept of a monitoring solution for analytical instruments. After choosing the software architecture and the technologies for every module, an application was developed and deployed in the cloud. Some web services were implemented to improve the program interoperability and to allow the creation of two separate monitoring interfaces: the main interface which is accessible through the browser and the mobile interface which is accessible through a dedicated application on the Apple iOS platform. These interfaces display in real time the status of laboratory instruments. The technician, who is alerted in case of failure, can quickly respond to solve the fault and restart production. The main interface also generates statistical reports on the instruments utilization to help managers in optimizing the laboratory. After commercial demonstrations of the project, the potential customers' feedback was very positive and gives hope for project continuation with economic benefits.

Keywords: monitoring, analytical instrument, architecture, dematerialization, deployment, mobility, service