

Département informatique
Université de France-Comté
Stage du 13 février au 4 juin
Année 2011-2012

Projet Yuukou II

Benoît MEILHAC



Rapport de stage

School of Electronics and Computer Science
University of Westminster
115 New Cavendish Street
London W1W 6UW

Tuteur de stage : M. Thierry DELAITRE
Responsable de stage : M. Jean-Michel HUFFLEN

Remerciements

Je tiens avant tout à remercier mon tuteur de stage M. Thierry DELAITRE pour son accueil à l'Université, pour m'avoir proposé un projet aussi intéressant, pour avoir mis à ma disposition tout ce dont j'avais besoin pour travailler correctement et aussi pour tout le temps qu'il m'a consacré.

Je remercie également mon responsable de stage M. Jean-Michel HUFFLEN pour m'avoir offert l'opportunité d'effectuer mon stage de fin d'études à Londres mais aussi pour ses conseils et l'aide apportée pendant la rédaction de ce rapport.

J'adresse aussi tous mes remerciements à toute l'équipe du CPC, avec qui j'ai partagé un bureau, pour m'avoir accueilli, pour leur aide et leur bonne humeur malgré la barrière de la langue en début de stage.

Merci encore à mes camarades francophones Damien HOSTACHE et Yacine MAGHEZZI venus me rejoindre en cours de stage pour les bons moments passés ensemble.

J'exprime également toute ma gratitude à Célia NASROUNE et Jérémie BOURSEAU pour le travail de relecture qu'ils ont effectué et l'aide apportée.

Table des matières

Remerciements	I
Introduction	1
1 Lieu de stage	3
1.1 University of Westminster	3
1.1.1 Présentation	3
1.1.2 Les différents campus et écoles	3
1.1.3 Quelques informations sur l'Université	4
1.2 School of Electronics and Computer Science	5
1.3 Équipes intégrées	6
1.3.1 <i>Infrastructure Team</i>	6
1.3.2 <i>Information Systems and Library Services</i>	6
1.4 <i>Centre for Parallel Computing</i>	7
2 Présentation du sujet	8
2.1 Le projet <i>Yuukou</i>	8
2.1.1 Présentation	8
2.1.2 Fonctionnement	8
2.1.3 Quelques chiffres	12
2.1.4 Changement vers <i>Yuukou II</i>	12
2.2 Le projet <i>Yuukou II</i>	13
2.2.1 Présentation du projet	13
2.2.2 Réflexions de l'équipe technique	13
2.2.3 Premières approches	14

2.2.4	Contraintes techniques	15
3	La notion de service Web	16
3.1	Service Web	16
3.1.1	Définition	16
3.1.2	Architecture	16
3.1.3	Fonctionnement	18
3.2	L'API JAX-WS	19
4	Le projet Yuukou II	21
4.1	Recherches	21
4.1.1	Architecture du projet	21
4.1.2	Nagios	23
4.1.3	L'IDE NetBeans	27
4.1.4	Le serveur Web GlassFish	27
4.1.5	Le SGBD MySQL	28
4.1.6	L'accès aux données des tables	28
4.1.7	Gestion de la base de données	29
4.1.8	Le format de retour	30
4.1.9	Complément d'information sur JSON	34
4.2	Communication avec Nagios	35
4.2.1	Le module MKLlivestatus	35
4.2.2	Requêtes pour Nagios	36
4.3	Fonctionnement du service Web	39
4.3.1	Le cycle principal	39
4.3.2	Les fonctions accessibles	41
4.3.3	Retour des fonctions	43
4.4	Fonctionnalités en place	44
4.4.1	Sécurisation du service Web	44
4.4.2	Génération de graphes d'utilisation	45
4.4.3	Gestion des emplois du temps	46
4.4.4	Catalogue logiciels des salles	47
4.4.5	Migration des données	47

4.5	Organisation du travail	48
4.5.1	Gestion du poste de travail	48
4.5.2	Travail en équipe	49
4.5.3	Tests du service Web	50
4.6	Exploitation du service Web	50
4.7	Problèmes rencontrés	52
5	Bilan	53
5.1	Bilan du travail réalisé	53
5.1.1	Récapitulatif du projet	53
5.1.2	Calendrier du stage	54
5.1.3	Bilan pour l'Université	55
5.1.4	Améliorations possibles	55
5.2	Bilan personnel	56
6	Conclusion	58
Bibliographie		59
Glossaire		60
Annexes		63
A Fichiers LQL Nagios		64
B Retours de requête LQL		66
C Base de données		68
D Exemples de retours JSON		74
Table des figures		76
Liste des tableaux		78

Introduction

Le cursus professionnel de deuxième année du Master informatique à l'Université de Franche-Comté de Besançon inclut la réalisation d'un stage en entreprise d'une durée minimum de quatre mois. L'Université donne, de plus, l'opportunité d'effectuer ce stage à l'étranger.

L'anglais est la langue prédominante dans le domaine de l'informatique mais c'est aussi un pré-requis indispensable dans certaines recherches d'emploi. C'est pour cela que j'ai saisi l'opportunité que m'offrait l'Université de réaliser mon stage au Royaume-Uni. C'est avec l'aide de M. Jean-Michel HUFFLEN, Maître de Conférences à l'Université de Franche-Comté, que j'ai pu intégrer pendant quatre mois l'Université de Westminster à Londres où j'ai été accueilli par M. Thierry DELAITRE, directeur des infrastructures à l'Université, afin de travailler sur le projet *Yuukou II*.

Ce projet est né d'une collaboration de deux équipes de l'Université : l'*Infrastructure Team* de la *School of Electronics and Computer Science* et les services centraux informatiques *Information Systems and Library Services*. De ce fait, j'ai intégré ces deux équipes tout en travaillant dans le *Centre for Parallel Computing*.

Yuukou, venant du japonais et signifiant validité, disponibilité, efficacité, est un projet mis en place dans le but de montrer l'utilisation des salles informatiques et d'en garder un historique. Ainsi, il permet de connaître la disponibilité des salles de l'Université et offre la possibilité de voir les ordinateurs disponibles, occupés ou encore éteints.

Cependant les principaux développeurs de cette application ne faisant plus partie de l'Université, le projet n'est plus maintenu. De plus, son principal intérêt serait de faire savoir à un étudiant les salles qui sont disponibles ainsi que les ordinateurs libres. Ce qu'il ne fait que très partiellement actuellement. En outre, l'Université change le fonctionnement de son infrastructure, ce qui aura, entre autres conséquences, de rendre l'application obsolète.

C'est dans ce contexte que s'inscrit le projet *Yuukou II*. Mon stage consiste donc à la mise en place d'un service Web permettant le suivi des différents ordinateurs de l'Université et pouvant retourner ces données pour un affichage sur un *smartphone*, ou encore un écran plasma se trouvant à l'entrée de chaque bâtiment par exemple.

Le présent rapport concerne le travail réalisé au sein de l'Université de Westminster. Une première partie présentera l'Université et les services que j'ai intégrés ainsi que ceux que j'ai côtoyés tout au long du stage. Une seconde partie traitera du projet *Yuukou*, avec une présentation de ses fonctionnalités et de ce à quoi il donne accès ainsi que des raisons qui ont conduit au projet *Yuukou II*. Cette partie permettra de découvrir le sujet de stage afin de comprendre les choix qui ont été réalisés dans les autres parties du rapport. La troisième partie mettra en avant la notion de service Web, le point central du projet. Ensuite, une quatrième partie présentera les différentes recherches qui ont été faites ainsi que le travail accompli et les problèmes rencontrés. Enfin, une dernière partie donnera un bilan des différents points retenus lors de ce stage avant de clore ce rapport.

NOTE : Les termes marqués d'un astérisque() sont définis dans le glossaire.*

1 Lieu de stage

1.1 University of Westminster

1.1.1 Présentation



FIGURE 1.1 – Blason de l'Université

L'Université de Westminster est une université publique de recherche située à Londres. À sa fondation, en 1838, elle portait le nom de *Royal Polytechnic Institution* et était la première école polytechnique ouverte en Angleterre. Son but était de fournir une institution où le public peut, à moindre coût, acquérir une connaissance pratique des divers arts et branches de la science en rapport avec les fabricants industriels, les opérations minières et l'économie rurale. Sa fondation est une réaction à la popularité grandissante de l'enseignement de type polytechnique en Europe continentale. Notamment avec l'Allemagne et la *Fachhochschule*, la France et l'*École Polytechnique* ou encore les États-Unis et la *Rensselaer Polytechnic Institute*.

En 1970, la *Royal Polytechnic Institution* devient *Polytechnic of Central London* après avoir fusionné avec *Holborn College of Law, Languages and Commerce*. C'est en 1992 que le statut d'université fut attribué à Westminster qui devint *University of Westminster*.

1.1.2 Les différents campus et écoles

L'Université compte quatre principaux campus, trois dans le centre de Londres : Regent, Cavendish, Marylebone et le quatrième à Harrow, à l'ouest de Londres.

- **Regent** : situé au 309 Regent Street, c'est le campus le plus ancien de l'Université, il contient une école : *School of Social Sciences, Humanities and Languages*;
- **Cavendish** : situé au 101-115 New Cavendish Street, dans le quartier de Fitzrovia proche de West End de Londres (entre Marylebone, Bloomsbury et

au nord de Soho), ce campus contient trois écoles : *School of Electronics and Computer Science*, *School of Life Sciences* et *Westminster Exchange* dont le but principal est l'amélioration de l'éducation ;

- **Marylebone** : situé sur Marylebone Road, en face du célèbre musée de cire *Madame Tussauds*, il contient deux écoles : *School of Architecture and the Built Environment* et *Westminster Business School* ;
- **Harrow** : situé dans le village de style victorien Harrow-on-the-Hill surplombant Londres, ce campus contient une école : *School of Media, Art and Design*.



FIGURE 1.2 – Campus de Regent Street

Outre ces principaux campus, l'Université compte deux autres sites :

- **Little Titchfield Street** : situé au 4-12 Little Titchfield street, il contient une école : *School of Law* ;
- **Wells Street** : situé au 32-38 Wells street, juste au coin des autres bâtiments du campus de Regent, il offre une grande variété de salles de classe.

L'Université gère également le *Westminster International University* à Tachkent en Ouzbékistan ainsi qu'un campus satellite à Paris à travers l'Académie Diplomatique de Londres.

1.1.3 Quelques informations sur l'Université

L'Université se situe officiellement au 309 Regent Street London W1B 2HM. Elle compte en 2011 plus de 20 000 étudiants venant de plus de 150 nations différentes grâce à de nombreux programmes d'échange avec d'autres universités.

Parmi ses diplômés les plus prestigieux :

- Sir **Alexander Fleming**, biologiste et pharmacologue britannique, prix No-

bel de Physiologie ou Médécine en 1945 avec Ernst Boris Chain et Sir Howard Walter Florey pour la découverte de la pénicilline et de son effet curatif sur diverses maladies infectieuses ;

- **Christopher Bailey**, directeur de la création chez *Burberry* ;
- **Charlie Watts**, musicien et batteur des *Rolling Stones*.

Concernant les cours, l'Université donne accès à plus de 300 cours différents.

1.2 School of Electronics and Computer Science

Le stage s'est déroulé dans cette école se trouvant, comme vu au § 1.1.2, dans le campus Cavendish au 101-115 New Cavendish Street. La figure 1.3 montre une vue extérieure du bâtiment avec la BT Tower, tour de communication possédée par l'opérateur de télécommunications BT Group, se trouvant juste à côté.



FIGURE 1.3 – Campus de New Cavendish Street

L'école propose divers parcours, aussi bien dans le domaine de la recherche que dans celui des études, allant de l'ingénierie électronique et informatique, aux mathématiques appliquées. Ainsi de nombreuses matières peuvent être abordées dans la formation comme la gestion des systèmes d'informations, la programmation parallèle et distribuée, l'intelligence artificielle mais aussi le développement de jeux vidéos et bien d'autres.

Il existe quatre grands pôles de recherche au sein de l'école qui sont :

Electronic and Communication Engineering, qui se concentre sur le traitement de signaux ainsi que dans la conception de composants et circuits pour les systèmes de communication ;

Operational Research and Intelligent Systems, dont les activités sont principalement la modélisation quantitative des systèmes complexes pour soutenir des processus décisionnels, la gestion des données et des informations, les technologies de bases de données destinées au processus de gestion et d'interopérabilité dans les environnements

où les logiciels sont omniprésents ;

Parallel and Distributed Computing, qui s'intéresse à la recherche et au développement dans le domaine des calculs parallèles et distribués ;

Semantic Computing and Systems Engineering, regroupe des chercheurs de différentes disciplines, comme le génie logiciel, les interactions homme-machine, et aborde les aspects théoriques et pratiques de l'informatique sémantique.

1.3 Équipes intégrées

Le projet *Yuukou II* – sur lequel j'ai travaillé pendant mon stage – regroupe deux services de l'Université. D'une part l'*Infrastructure Team*, et d'autre part l'*Information Systems and Library Services* (ISLS). Le but de mon stage étant de fournir un pilote de ce qu'il est possible de faire afin que mon travail puisse être repris par les deux services et développé plus en profondeur.

1.3.1 *Infrastructure Team*

L'*Infrastructure Team* est une équipe de 13 personnes dans la *School of Electronics and Computer Science* qui a pour rôle de gérer les laboratoires spécialisés de l'école. Ce sont 34 laboratoires de l'école qui sont sous la responsabilité de cette équipe. Ces laboratoires sont dit spécialisés du fait des possibilités qui peuvent y être réalisées : l'installation de nouveaux systèmes d'exploitations sur les ordinateurs par exemple.

L'équipe apporte aussi un support pour les laboratoires d'enseignement et pour les groupes de recherches dans l'école. Elle reprend et étend les services fournis par les services centraux informatiques (voir ISLS au § 1.3.2) comme l'authentification, les quotas de disques, le réseau ou encore les outils d'enseignement.

1.3.2 *Information Systems and Library Services*

ISLS est un département dont le but est de contribuer à la qualité de l'éducation dans l'Université de Westminster à travers le développement et la livraison de services.

Le département est composé de cinq services qui sont :

- *Archive Services*, qui gère les archives de l'Université ;
- *Corporate Information*, qui développe et livre des applications qui prennent en charge le fonctionnement des activités (enregistrement des étudiants, emplois du temps, ...);
- *Infrastructure*, qui gère l'infrastructure informatique pour fournir des services informatiques et de télécommunications ;
- *Learning and Research Support*, qui gère la bibliothèque et les services informatiques pour le personnel et les étudiants ;

- *Resources and Planning*, qui gère les ressources et le planning du département.

1.4 Centre for Parallel Computing

Durant toute la durée de mon stage, j'ai travaillé dans le *Centre for Parallel Computing* (CPC). Le bureau se situe au 7^e étage de la *School of Electronics and Computer Science* et est composé de sept personnes. Cette équipe appartient au pôle recherche *Parallel and Distributed Computing*.

Ce centre se focalise sur la recherche dans la technologie et les applications de calculs parallèles et distribués. Les activités du centre incluent le développement d'outils et d'environnements pour soutenir le cycle de vie dans le génie logiciel comme la simulation d'événements discrets en parallèle. Les chercheurs travaillent en collaboration avec l'Université hongroise Sztaki.

Parmi les projets développés par le service, on peut citer :

- **Gemlca** (*Grid Execution Management for Legacy Code Applications*), solution générale dont le but est de déployer le code d'applications existantes, quelque soit le langage, comme un service de grille ;
- **NGS** (*National Grid Service*), solution dont le but est de fournir un accès électronique cohérent pour les chercheurs du Royaume-Uni à toutes les ressources de calculs et de données ainsi qu'à l'équipement nécessaire pour mener à bien leurs travaux, indépendamment de l'emplacement de la ressource ou du chercheur ;
- **SHIWA** (*Sharing Interoperable Workflows for large-scale scientific simulations on Available DCIs*¹), projet qui a pour but l'interopérabilité de différents systèmes de *workflow** européen (comme Moteur, P-Grade, Askalon ou Gwes) à l'aide de l'approche par granularité (grain fin ou grossier).

Pendant le stage, j'ai été rejoint par deux étudiants en troisième année de Licence Informatique à l'Université de Franche-Comté de Besançon : Yacine MAGHEZZI, travaillant sur la partie affichage du projet qui m'a été confié (plus d'explications au § 4.1.1) et Damien HOSTACHE, travaillant sur le développement d'une application pour le portail Web de l'Université de Westminster dans le but de gérer des objets en trois dimensions.

1. *Distributed Computing Infrastructure*

2 Présentation du sujet

2.1 Le projet *Yuukou*



2.1.1 Présentation

Le terme Yuukou comme abordé dans l'introduction de ce rapport, vient du japonais *Yuukou* et signifie validité, disponibilité, efficacité. C'est un système permettant la récupération des informations depuis les serveurs LDAP^{1*} de l'Université afin de comprendre et construire l'infrastructure des ressources et ainsi de voir sa facilité d'utilisation. L'architecture du système utilise un processus d'apprentissage simple pour déduire et maintenir la structure à jour avec un minimum de réglages initiaux. *Yuukou* a été créé pour montrer l'utilisation des salles informatiques et conserver un historique des informations sur le campus de New Cavendish.

2.1.2 Fonctionnement

Le but principal de *Yuukou* est d'afficher l'occupation des salles informatiques en se rapprochant autant que possible du comportement d'un système fonctionnant en temps réel. Pour ce faire, l'application a été divisée en deux parties comme le montre la figure 2.1.

1. *Lightweight Directory Access Protocol*

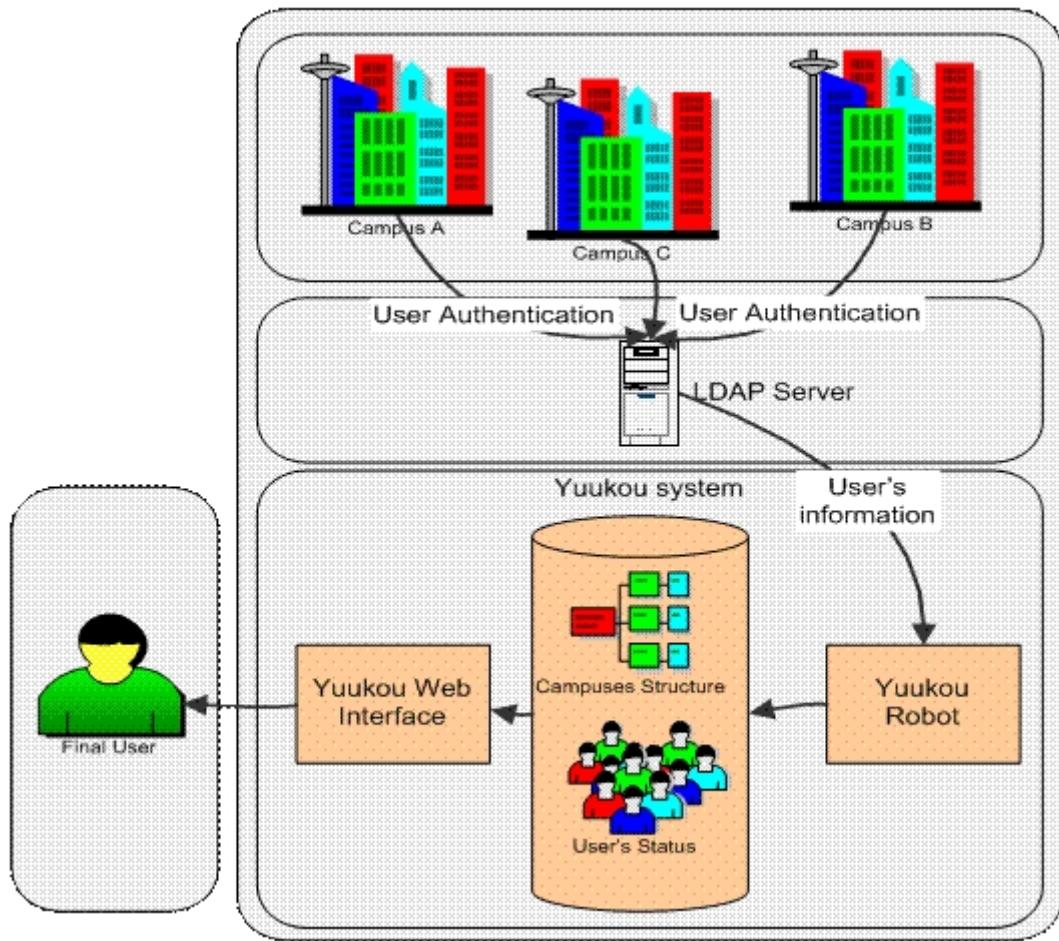


FIGURE 2.1 – Architecture de Yuukou

Première partie

La première partie est un programme écrit en Perl* qui récupère les informations de connexion des utilisateurs depuis un serveur LDAP* et qui se charge de construire, modifier ou mettre à jour l'architecture réseau de l'Université tout en stockant les informations dans une base de données relationnelle de type MySQL.

Le robot de Yuukou offre deux fonctionnalités : il permet de mettre à jour les données de connexion *via* le serveur LDAP* toutes les cinq minutes environ pour une utilisation normale, et de mettre à jour le statut des ressources (ordinateurs) toutes les demi-heures, là aussi pour une utilisation normale.

Deuxième partie

La seconde partie est un ensemble de pages Web écrites en PHP^{2*} et hébergées sur un serveur Web permettant de présenter les données collectées à l'utilisateur final. Ces pages sont de deux types : les pages publiques et les pages privées.

Les pages publiques sont accessibles par tous les utilisateurs de l'Université le désirant. Les pages privées, quant à elles, ne sont accessibles qu'aux administrateurs.

Les pages publiques

Les pages publiques permettent l'affichage des pages suivantes :

- une page contenant tous les campus et salles informatiques actuellement utilisées ;
- une page par campus avec les salles informatiques actuellement utilisées ;
- une page par campus et département avec les salles informatiques actuellement utilisées.

Les pages publiques offrent une vision générale de chaque salle : le nombre de ressources totales, disponibles, occupées par un utilisateur et dans un état inconnu. Il est à noter que chacunes des précédentes pages peut être affichées sur un écran plasma présent dans les différents bâtiments de l'Université. La figure 2.2 donne un exemple de page publique.

Les pages privées

Les pages privées permettent l'affichage des pages suivantes :

- une page d'identification *via* LDAP* pour l'administrateur ;
- toutes les pages publiques mais bénéficiant de fonctionnalités supplémentaires :
 - liste des ressources éteintes ;
 - une fenêtre permettant d'avoir des informations sur un utilisateur actuellement connecté à une ressource (identifiant, nom, photo, heure de connexion et durée de la session) ;
 - possibilité d'ajouter des commentaires sur les utilisateurs ;
 - liens vers les statistiques des salles informatiques.

La figure 2.3 donne un exemple des fonctionnalités supplémentaires auxquelles un administrateur a accès.

2. Personal Home Page ou PHP : Hypertext Preprocessor

Vue sur le produit

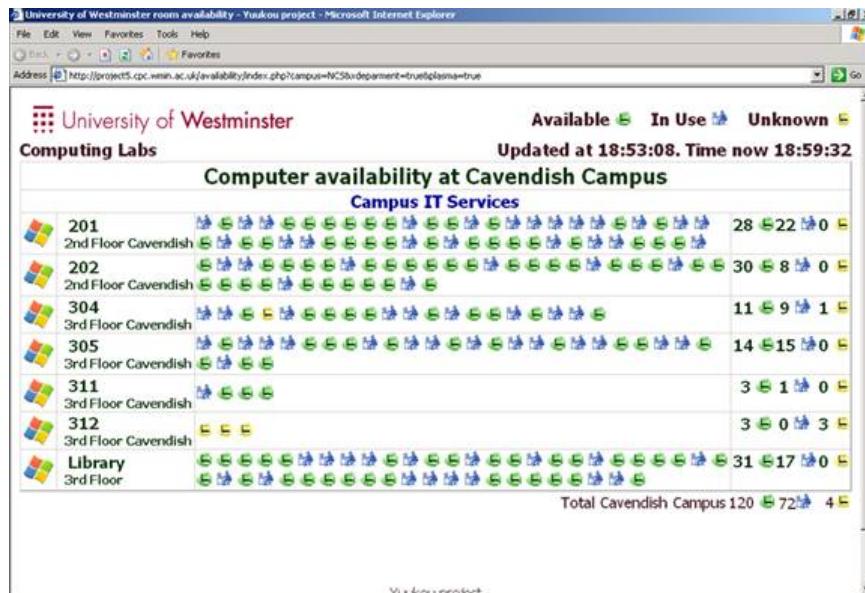


FIGURE 2.2 – Exemple de page publique de *Yuukou* repésentant un campus et l'utilisation des salles informatiques d'un département

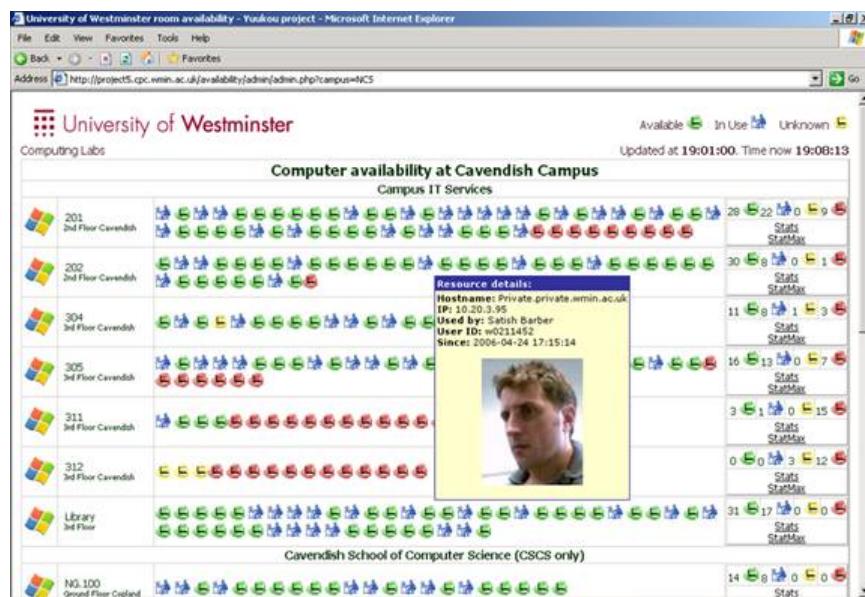


FIGURE 2.3 – Exemple de page privée de *Yuukou* montrant en particulier les données d'un utilisateur

2.1.3 Quelques chiffres

Le projet *Yuukou* permet de surveiller le réseau du campus de New Cavendish soit 43 salles informatiques, ce qui représente 661 PC. Le support des Macintosh n'étant pas pris en compte.

2.1.4 Changement vers *Yuukou II*

L'Université souhaite reprendre le principe du projet *Yuukou*, cependant elle est confrontée à différents problèmes. Le premier étant que les personnes ayant développé le projet ont quitté l'Université. Ce qui signifierait, pour la personne ou l'équipe qui serait en charge de reprendre le projet, de prendre du temps pour se former à Perl*, comprendre tout ce qui a été déjà réalisé et ensuite seulement, commencer à développer. Actuellement de nombreux projets sont en cours et il n'est pas possible pour une équipe de passer du temps sur l'existant.

Un autre problème est les changements importants, du point de vue infrastructure, qui sont en train d'être mis en place. En effet, l'Université qui utilisait eDirectory* de Novell pour gérer ses annuaires LDAP* a commencé à migrer toutes ses données vers le système Active Directory* de Microsoft qui est censé être plus efficace. Ce changement rendrait *Yuukou* obsolète.

Point suivant, la volonté de donner accès aux informations sur les salles, pas seulement en utilisant un navigateur Internet, mais aussi et surtout en utilisant un *smartphone* (iPhone ou autre par exemple) ou une tablette (IPad par exemple), chose que l'ancien logiciel ne peut pas fournir. De ce fait, un étudiant aurait à tout moment les informations sur les salles *via* son *smartphone* ou sa tablette, si tant est qu'il ait l'un ou l'autre.

À ces précédents problèmes, d'autres viennent s'ajouter :

- les Macintosh ne sont pas pris en compte ;
- le logiciel est assez monolithique, il deviendrait donc très difficile et complexe de vouloir l'étendre ;
- *Yuukou* ne prend en compte que les données en temps réel et ne garde pas un historique ;
- son utilisation prête à confusion car il n'a aucun interfaçage avec l'emploi du temps, de ce fait, quand une salle est présentée comme libre, il n'y a aucun moyen, avec le logiciel, de savoir si un cours s'y déroule ou non.

C'est en considérant tous ces points qu'il a été décidé d'abandonner le projet *Yuukou* afin de pouvoir mettre en place *Yuukou II* qui répondrait aux attentes de l'Université.

2.2 Le projet *Yuukou II*

Yuukou II a pour but de combler les lacunes de *Yuukou* et d'aller plus loin en termes de fonctionnalités qu'il peut offrir. Le projet sera tout d'abord présenté avec les principaux points le composant. Ensuite seront abordées les différentes réflexions que les équipes intéressés par le projet ont effectuées. Ces réflexions donneront lieu aux premières idées qui en ont émergées. Enfin, les principales contraintes techniques du projet seront décrites.

2.2.1 Présentation du projet

Dans son fonctionnement général, *Yuukou II* doit permettre de donner à un étudiant ou toute personne travaillant à l'Université et cherchant à utiliser un ordinateur, une vue globale des ressources qui sont disponibles actuellement. Les données devant être bien sûr exactes afin que la personne n'ait pas de mauvaise surprise en se rendant dans une salle qu'elle pensait libre. L'affichage pourra se faire *via* un *smartphone*, une tablette ou encore un navigateur Internet.

Le but du stage est :

- la conception d'un logiciel permettant la récupération de données concernant les connexions sur les différentes ressources de l'Université et cela en temps réel ;
- la gestion de la persistance de ces données ;
- la création d'un maximum de fonctionnalités retournant les informations utiles dans le but d'être exploitées pour l'affichage sur les différents supports.

La création d'applications permettant l'affichage des résultats ne fait pas partie de ce sujet de stage. Ici, seule la partie récupération, stockage et retour des données est abordée.

2.2.2 Réflexions de l'équipe technique

Différents acteurs de l'Université intéressés dans le projet *Yuukou II* ont commencé à fixer une liste des fonctionnalités qu'ils aimeraient voir avec l'application finale.

Concernant les ressources

- Macintosh et Windows ;
- connaître l'état de la ressource, éventuellement la démarrer à distance (WOL^{3*}) ;
- inclure une surveillance partielle du matériel et des logiciels ;

3. *Wake On Lan*

- utilisation des services d’Active Directory*.

Concernant les données récupérées

- mettre au point un formalisme avec les autres services de l’Université concernant les informations sur les salles, campus et départements ;
- stocker les informations dans une base de données SQL^{4*} ;
- mettre en place des outils pour générer des statistiques à partir des données stockées (utilisation d’une salle, nombre de connexions par jour dans un mois pour une salle, …).

Concernant les données retournées

- génération de flux RSS^{5*} pour retourner des informations ;
- l’affichage doit être en temps réel et doit aussi permettre d’avoir une vue globale dans le temps : utilisation des emplois du temps pour savoir quelle salle est libre et à quel moment.

Cette liste n’est pas complète étant donné qu’elle ne prend pas en compte la partie « affichage » des résultats du fait que le projet ne consiste qu’à la récupération, au traitement et au retour de données.

L’idée initiale était de développer un projet pilote qui permettrait d’avoir une vue sur ce qu’il est possible de faire et sur la façon de le faire. Il serait ensuite repris par les équipes de l’Université pour être terminé. Ce projet consisterait en la création d’un service Web (la notion sera expliquée plus en détail au § 3) écrit en C# et utilisant le *framework** .Net de Microsoft. Le service Web devra offrir un maximum de fonctionnalités et être exploité par une application mobile pour *smartphone* et par un site Web pouvant être projeté sur les écrans plasma à l’entrée de chaque site.

Cependant, après réflexion avec M. Thierry DELAITRE, la structure, les objectifs et le projet en général ont été revus. Dans l’idée initiale, pour connaître l’état d’une ressource, si elle est éteinte ou allumée par exemple, il aurait fallu interroger Active Directory*. De plus, n’est connu que l’état si un utilisateur est connecté. Il faudrait des traitements supplémentaires pour pouvoir fixer précisément l’état d’une ressource, avec un *ping* par exemple pour savoir si la ressource est éteinte ou non.

2.2.3 Premières approches

Une solution simple, rapide et fonctionnelle avait déjà été mise en place afin de « monitorer », c’est-à-dire surveiller le fonctionnement des différentes ressources de certaines salles dans l’Université. Nagios est une application permettant d’effectuer une

4. Structured Query Language

5. Rich Site Summary

surveillance système et réseau. Il permet de connaître l'état d'une machine ainsi que d'autres informations comme le système d'exploitation utilisé, la version de Java installée, la charge du processeur, ...

En partant de ce logiciel, le projet consiste en la récupération des données de Nagios, leur traitement et le développement des fonctionnalités permettant à une application extérieure de pouvoir afficher les informations.

Les objectifs principaux deviennent les suivants :

- création d'un service Web en utilisant l'API^{6*} Java JAX-WS⁷ ;
- mise en place d'une méthode de communication avec Nagios ;
- création de la base de données permettant l'archivage ;
- mise en place d'un cycle permettant de traiter les informations récupérées ;
- définition de fonctions utiles pour une application cliente ;
- choix d'une structure de retour des informations pour une application cliente ;
- faire un lien avec l'emploi du temps des différentes salles sous surveillance.

2.2.4 Contraintes techniques

Deux des principales contraintes du cahier des charges sont l'utilisation de logiciels libres et du langage de développement Java, tout en me laissant un maximum de liberté dans les autres choix.

Durant le stage, un ordinateur m'a été fourni avec un libre choix sur le système d'exploitation. De ce fait j'ai opté pour un Linux Mint 11 nommé *Katya*, que j'ai l'habitude d'utiliser.

Un autre ordinateur, lui contenant un Debian 6.0.5 nommé *Squeeze*, a été mis à ma disposition en tant que serveur distant hébergeant le service Web ainsi que les différents outils nécessaires à son fonctionnement. Le serveur distant contient le logiciel Nagios, le serveur Web permettant de faire fonctionner la dernière version stable du service Web ainsi que le système de gestion de bases de données (SGBD).

Les tests étant en premier lieu réalisés en local sur la machine de développement et ensuite, quand le fonctionnement était garanti, l'application était déployée sur le serveur distant. Des renseignements supplémentaires seront apportés au § 4.5.1.

6. Application Programming Interface
7. Java API for XML Web Services

3 La notion de service Web

Le projet repose donc sur la création d'un service Web. Son but étant de mettre à disposition d'un client, des méthodes retournant des informations pour qu'elles soient traitées et affichées. C'est pourquoi la notion de service Web sera expliquée dans un premier temps. Une définition sera donnée ainsi qu'une architecture basique et la façon dont fonctionne un service Web. Puis, dans un deuxième temps, une présentation de l'API* utilisée, JAX-WS, pour le développement du service Web sera faite. Elle permettra de donner un aperçu des méthodes de développement d'un service Web.

3.1 Service Web

3.1.1 Définition

Un service Web est une application accessible par le réseau, qui permet à un client de dialoguer avec le service Web et cela indépendamment de tout langage de programmation et de toute plate-forme d'exécution. Un service Web développé en Java peut donc être accessible par un client PHP* par exemple. C'est un système de messagerie standard utilisant le protocole HTTP pour communiquer à travers le réseau et échangeant des fichiers XML. Son intérêt résulte dans l'utilisation de normes (HTTP et XML) permettant l'interopérabilité des services. De plus, le fait que les services Web n'imposent pas de modèle de programmation spécifique permet aux vendeurs d'outils de développement d'offrir des méthodes différentes et donc de facilement différencier leurs produits de ceux de leurs concurrents.

3.1.2 Architecture

Lorsqu'on parle de services Web, on parle aussi de SOA, *Service-Oriented Architecture* ou architecture orientée services en français. Une architecture orientée services est un style d'architecture qui a pour objectif une interdépendance faible entre différents agents logiciels (modules, services). Les services Web possèdent trois normes principales

qui sont SOAP¹, WSDL² et UDDI³ qui permettent respectivement de connaître la façon dont les messages sont échangés, leurs descriptions et la façon de les découvrir sur le réseau.

Échange de messages

Dans la grande majorité des services Web, le protocole principal de communication est SOAP. Il offre le transport d'objets sérialisés (objets représentés par un flux d'octets) et autres données en XML, et l'appel de procédures distantes.

Description d'un service Web

Un service Web est décrit par un document WSDL. C'est un langage de description basé sur XML qui permet de donner de façon précise les détails concernant le service Web :

- son protocole de communication utilisé ;
- le format de messages requis pour communiquer avec lui ;
- les méthodes que le client peut utiliser ;
- sa localisation.

Un fichier WSDL permet ainsi de savoir comment communiquer avec le service Web.

Découverte d'un service Web

Pour utiliser un service Web, il faut savoir s'il existe. UDDI est une norme qui définit le mécanisme pour découvrir dynamiquement des services Web. C'est en fait un annuaire de services permettant de localiser sur le réseau le service Web demandé. Il comporte :

- *des pages blanches*, liste des entreprises ainsi que les informations les concernant ;
- *des pages jaunes*, liste des services Web de chacunes des entreprises sous le format WSDL
- *des pages vertes*, liste des informations techniques précises sur les services fournis.

1. *Simple Object Access Protocol*

2. *Web Services Description Language*

3. *Universal Description, Discovery and Integration Service*

3.1.3 Fonctionnement

Le fonctionnement des services Web s'articule autour de trois acteurs principaux illustrés par la figure 3.1.

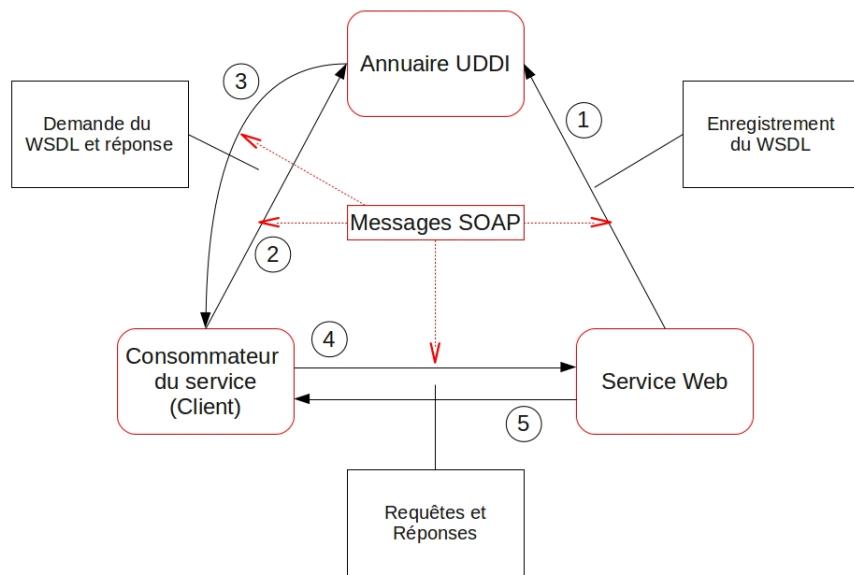


FIGURE 3.1 – Fonctionnement des services Web

Les étapes impliquant la mise à disposition et l'utilisation (consommation) d'un service Web sont les suivantes :

1. le service Web décrit son service en utilisant WSDL ; cette définition est publiée dans l'annuaire UDDI ;
2. le client envoie une ou plusieurs requêtes afin de localiser le service Web à l'annuaire et de déterminer comment communiquer avec ce service ;
3. une partie du fichier WSDL fourni par le service Web est passé au client. Le client peut maintenant savoir quelles requêtes et réponses envoyer au service Web ;
4. le client utilise le fichier WSDL pour envoyer une requête au service Web ;
5. le service Web fournit la réponse attendue au client.

3.2 L'API JAX-WS

Java API* for XML Web Services (JAX-WS) est une API Java permettant de simplifier la création, le développement et le déploiement de services Web clients et de services Web prestataires. Il fonctionne avec un système d'annotations spécifiques du code Java. JAX-WS fait partie de la plate-forme Java EE⁴ de Sun Microsystems. Anciennement appelée JAX-RPC⁵, ce changement de nom intervient lors de l'abandon du style RPC⁶, des appels de procédures distantes, pour des services Web de style document, avec une transmission de données au format XML définies dans un schéma XML.

Toute la partie communication de JAX-WS est gérée avec des messages de type SOAP à travers HTTP. L'API* permet de cacher toute la complexité des messages, le développeur choisit simplement la manière d'implanter le service. Pour cela, il existe deux méthodes : *top-down* et *bottom-up*.

Top-down

La méthode *top-down* permet de développer un service Web en partant d'un fichier WSDL. Ce fichier doit contenir la complète description de tout le service Web, ensuite le code Java est généré automatiquement. Les classes doivent ensuite être complétées avant déploiement.

Bottom-up

La méthode *bottom-up* permet, quant à elle, de développer un service Web en partant du code Java. Le code Java est d'une part annoté, ensuite le fichier WSDL est automatiquement généré. Le service peut enfin être déployé.

Choix de la méthode et exemple

C'est la méthode *bottom-up* qui a été employée dans le projet. Elle permet de ne se soucier que du code Java, le reste étant géré automatiquement par NetBeans avec la construction du service Web et le déploiement sur le serveur GlassFish. NetBeans et GlassFish seront décrits respectivement aux § 4.1.3 et 4.1.4. Le code 3.2 donne un exemple d'annotations sur un code Java.

4. Java Platform, Entreprise Edition
5. Java API for XML-based RPC
6. Remote Procedure Call

```
1 @WebService(name = "ExempleService")
2 public class ExempleService {
3     @WebMethod(operationName = "afficher")
4     public String afficher(
5         @WebParam(name = "texte") String texte){
6         ...
7     }
8 }
```

FIGURE 3.2 – Exemple de classe Java annotée avec JAX-WS

JAX-WS *Reference Implementation* est disponible sur son site Internet[1] en version 2.2.6.

4 Le projet Yuukou II

Une fois le sujet posé et la notion de service Web comprise, plusieurs recherches ont été nécessaires afin de pouvoir commencer le projet en lui-même. C'est seulement ensuite que la phase de développement a pu commencer. Le premier point traitera de la phase de recherche sur les outils et les choix qui ont été pris pour le projet. Ensuite un second point abordera la méthode utilisée pour accéder aux informations indispensables au fonctionnement du service Web. Les deux points suivants seront consacrés au fonctionnement et aux fonctionnalités qu'offre le service Web. Le premier de ces deux points permettra de découvrir le cycle qui permet de récupérer les informations de Nagios et de les archiver, ainsi que les méthodes auxquelles un client peut accéder pour récupérer une partie ou toutes les informations archivées par le service Web. Le deuxième de ces deux points expliquera les différentes fonctionnalités dont *Yuukou II* bénéficie. Suite à cela, un autre point aura pour but d'expliquer l'organisation du travail qui a été adoptée tout au long du développement du service Web. Une vue expliquant comment est exploité le service Web sera faite ensuite et montrera des captures d'écrans de ce qui a été réalisé. Enfin, un dernier point permettra de voir les principaux problèmes qui ont été rencontrés lors du stage.

4.1 Recherches

Après avoir compris comment fonctionne un service Web, il a fallu se concentrer sur les différents outils et la façon de les utiliser afin de développer le projet. La réflexion se portera d'abord sur une architecture du projet dont le but est de se faire une idée de comment toutes les pièces peuvent s'articuler les unes avec les autres. Ensuite il sera fait une description des outils utilisés durant le projet. Enfin les choix techniques opérés lors du stage seront décrits.

4.1.1 Architecture du projet

Le premier travail a été la réflexion à propos de la mise en place d'une solution pouvant communiquer avec Nagios dont une description sera faite au § 4.1.2. Les figures 4.1 et 4.2 présentent le fruit des recherches qui ont été faites sur la mise en place

du projet *Yuukou II*.

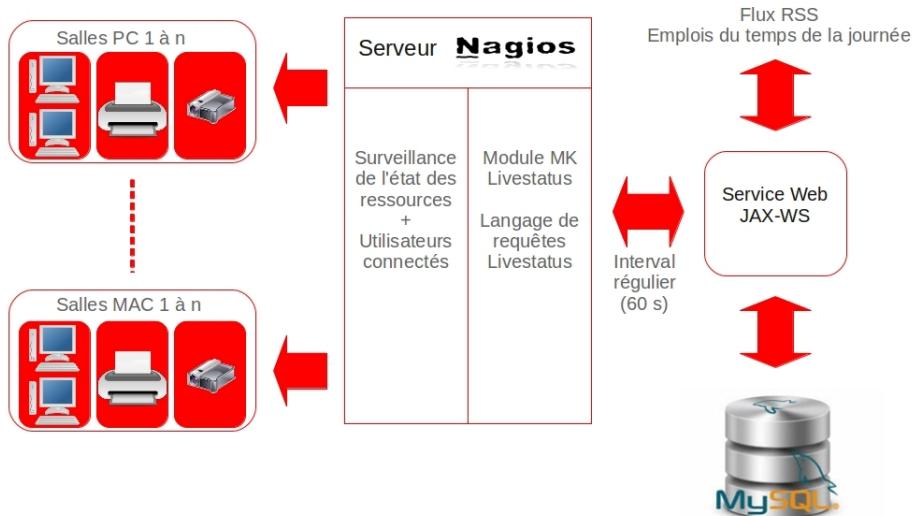


FIGURE 4.1 – Architecture du projet, partie service Web

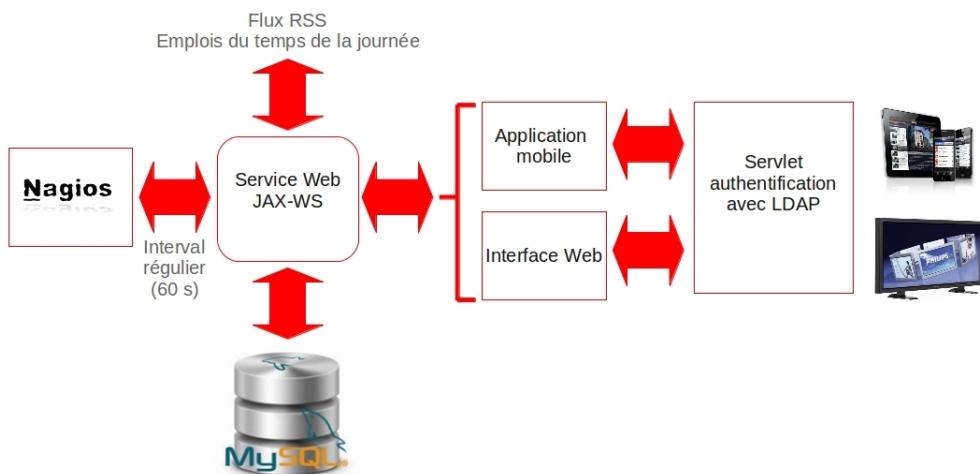


FIGURE 4.2 – Architecture du projet, partie affichage

Partie service Web

Basiquement, Nagios sert à surveiller les ressources auxquelles il lui est permis d'accéder. De ce fait, il doit garder des traces des informations qu'il récupère. Ces informations sont stockées dans un fichier. Cependant, il existe un module qui est directement

intégré dans le processus de fonctionnement de Nagios et qui offre une grande rapidité dans l'envoi d'informations. Le module *MK Livestatus* permet, à l'aide d'un langage de requêtes qui lui est propre, de récupérer les informations que garde Nagios.

Le service Web doit permettre, dans un premier temps, de récupérer toutes les informations utiles pour un archivage des données. Elles seront ensuite stockées dans une base de données MySQL. Ces informations sont :

- les données récupérées *via* le module *MK Livestatus*, la récupération doit être faite à intervalles réguliers ;
- les différents emplois du temps de toutes les salles que Nagios surveille, la récupération doit être faite une fois par jour ;
- les données sur un utilisateur inconnu (nom, prénom, rôle : étudiant par exemple, photo) *via* un serveur LDAP*.

Partie affichage

Dans un deuxième temps, le service Web doit pouvoir retourner ces informations triées à un utilisateur normal ou à un administrateur afin de garantir l'affichage sous la forme d'une application mobile ou d'une interface Web. La réflexion doit ainsi se porter sur le contenu des différentes méthodes auxquelles un utilisateur pourra avoir accès (en fonction qu'il soit administrateur ou non). Le but étant un affichage le plus rapide possible des informations demandées.

L'utilisateur doit s'authentifier *via* un *Servlet** qui communique avec LDAP* et qui donne un accès à l'application. Le reste de la communication sera sécurisée comme expliqué dans le § 4.4.1.

Le dernier point est de choisir un format pour les données qui seront échangées entre le service Web et l'application cliente.

C'est sur la partie affichage que Yacine MAGHEZZI est intervenu durant son stage.

4.1.2 Nagios



FIGURE 4.3 – Logo de Nagios

Présentation

Nagios est une application permettant la surveillance système et réseau de toute une infrastructure informatique. Nagios complète cette surveillance en offrant la possibilité d'alerter les équipes en charge de l'infrastructure en cas d'apparition de problèmes comme une panne ou encore un fonctionnement anormal. C'est actuellement la solution de surveillance la plus efficace du marché.



Nagios a été créé en 1999 et portait initialement le nom de *NetSaint Network Monitor*. Il est écrit en C et est conçu pour un environnement Unix. Le projet a été maintenu jusqu'en 2002 avant de changer de nom pour devenir Nagios en réponse à une contestation judiciaire par les propriétaires d'une marque similaire. N.A.G.I.O.S. est l'acronyme récursif de « *Nagios Ain't Gonna Insist On Sainthood* » où *Sainthood* est une référence à *NetSaint*.

Maintenant connu sous le nom de Nagios XI, Nagios est un logiciel libre sous la licence GNU GPL V2. Il est disponible sur son site Internet[2] en version 3.2.1.

Fonctionnement

L'architecture de base de Nagios est très simple, elle comporte :

- un ordonnanceur pour gérer les vérifications ainsi que les actions à prendre sur les différents incidents ;
- une partie graphique : visible à travers un simple serveur Web ;
- des sondes : greffons (ou *plugins* en anglais) dans Nagios, ce sont de petits *scripts* permettant d'effectuer diverses vérifications.

À la base, Nagios est un moteur d'ordonnancement de vérifications diverses et variées dont les vérifications sont effectuées *via* des greffons. Ces vérifications peuvent être la charge d'utilisation du CPU¹, l'espace disque utilisé ou encore les utilisateurs actuellement connectés. Dans le cadre de la vérification de l'infrastructure, deux types de machines sont observées : les ordinateurs dotés de Windows et les ordinateurs dotés de Macintosh.

Nagios étant installé et fonctionnel depuis un peu plus d'un an sur le site de New Cavendish, les greffons pour observer les deux types de machines ont déjà été développés. Pour les machines fonctionnant sous Windows, le greffon est en fait l'appel à la commande Unix *winexe* qui permet l'exécution de commandes à distance sur des machines Windows. Pour les machines fonctionnant sous Macintosh, le greffon effectue une connexion SSH², offrant une connexion sécurisée sur une machine distante pour ensuite exécuter la commande Unix *who* permettant l'obtention de l'utilisateur connecté. Pour

1. *Central Processing Unit* ou processeur en français
 2. *SecureShell*

les machines bénéficiant d'un double *boot* (d'un démarrage de l'ordinateur permettant de choisir entre deux systèmes d'exploitation) Windows-Unix, seul le système Windows est surveillé. La configuration actuelle de Nagios ne permet pas de récupérer avec exactitude l'utilisateur connecté sur le système Unix. Il est juste possible de savoir que l'ordinateur est occupé, mais pas par qui.

La configuration de Nagios s'articule entre différents concepts : les *hosts*, les *hostgroups* et les *services*. Un *host* représente un ordinateur, cet ordinateur possède des *services* comme le *service check_whoisloggedin* permettant de savoir si un utilisateur est actuellement connecté sur ladite machine. Ces *services* sont donc les greffons de Nagios. Les *hosts* font enfin partie d'un groupe d'ordinateurs *hostgroups* qui représente une salle informatique au sein de l'Université.

« *Monitoring* » à l'Université

Initialement, Nagios surveillait juste les machines se situant sur le campus de New Cavendish, soit 31 salles PC, soit environ un peu plus de 600 machines. Actuellement, ce sont 102 salles qui sont sous la surveillance de Nagios, soit 99 salles utilisant Windows soit 1920 PC, 3 salles utilisant des Macintosh soit 63 MAC, pour un total de 1983 machines à travers toute l'Université. Cela représente la grande majorité des ordinateurs de l'Université. Il faut noter que seuls les Macintosh de New Cavendish sont sous surveillance, les autres nécessitant d'être listés et des accès différents. De ce fait, le nombre de machines devrait augmenter par la suite.

Vue sur la surveillance de Nagios

Les figures 4.4 et 4.5 donnent un aperçu de l'interface graphique de configuration mais aussi de suivi de Nagios. La figure 4.4 offre une vue d'ensemble sur tous les *hosts* appartenant à tous les *hostgroups* surveillés. La figure 4.5, quant à elle, offre une vue pour un *hostgroup* spécifique : la salle pourtant le nom e-cg24.

Host	Status	Last Check	Duration	Status Information
e-cg24.pc-01	UP	2012-05-25 19:04:14	2d 12h 10m 25s	PING OK - Packet loss = 0%, RTA = 5.45 ms
e-cg24.pc-02	UP	2012-05-25 19:05:15	0d 12h 10m 5s	PING OK - Packet loss = 0%, RTA = 1.46 ms
e-cg24.pc-03	UP	2012-05-25 19:01:55	1d 12h 10m 45s	PING OK - Packet loss = 0%, RTA = 1.88 ms
e-cg24.pc-04	UP	2012-05-25 19:02:04	1d 12h 10m 15s	PING OK - Packet loss = 0%, RTA = 1.73 ms
e-cg24.pc-05	UP	2012-05-25 19:05:24	0d 12h 9m 55s	PING OK - Packet loss = 0%, RTA = 0.47 ms
e-cg24.pc-06	UP	2012-05-25 19:04:44	1d 9h 37m 25s	PING OK - Packet loss = 0%, RTA = 0.48 ms
e-cg24.pc-07	UP	2012-05-25 19:01:35	1d 12h 10m 25s	PING OK - Packet loss = 0%, RTA = 0.65 ms
e-cg24.pc-08	DOWN	2012-05-25 19:01:56	0d 3h 21m 5s	CRITICAL - Network Unreachable (10.7.65.17)
e-cg24.pc-09	UP	2012-05-25 19:05:05	0d 12h 10m 15s	PING OK - Packet loss = 0%, RTA = 0.41 ms
e-cg24.pc-10	UP	2012-05-25 19:02:34	1d 12h 9m 35s	PING OK - Packet loss = 0%, RTA = 0.46 ms
e-cg24.pc-11	UP	2012-05-25 19:04:55	3d 2h 37m 25s	PING OK - Packet loss = 0%, RTA = 0.51 ms
e-cg24.pc-12	UP	2012-05-25 19:05:35	0d 12h 9m 45s	PING OK - Packet loss = 0%, RTA = 0.42 ms
e-cg24.pc-13	UP	2012-05-25 19:02:04	0d 12h 9m 35s	PING OK - Packet loss = 0%, RTA = 1.45 ms
e-cg24.pc-14	UP	2012-05-25 19:02:04	3d 12h 6m 25s	PING OK - Packet loss = 0%, RTA = 1.72 ms
e-cg24.pc-15	DOWN	2012-05-25 19:05:15	14d 0h 18m 45s	CRITICAL - Network Unreachable (10.7.65.24)
e-cg24.pc-16	UP	2012-05-25 19:02:04	10d 12h 6m 45s	PING OK - Packet loss = 0%, RTA = 1.52 ms
e-cg24.pc-17	UP	2012-05-25 19:04:45	1d 9h 43m 25s	PING OK - Packet loss = 0%, RTA = 0.46 ms
e-cg24.pc-18	UP	2012-05-25 19:01:36	0d 12h 9m 5s	PING OK - Packet loss = 0%, RTA = 0.43 ms
e-cg24.pc-19	UP	2012-05-25 19:02:34	3d 12h 8m 5s	PING OK - Packet loss = 0%, RTA = 0.43 ms
e-cg24.pc-20	UP	2012-05-25 19:01:35	0d 12h 9m 15s	PING OK - Packet loss = 0%, RTA = 0.46 ms
e-cg24.pc-21	UP	2012-05-25 19:02:45	7d 10h 16m 55s	PING OK - Packet loss = 0%, RTA = 1.77 ms
e-cg24.pc-22	UP	2012-05-25 19:04:55	4d 2h 18m 45s	PING OK - Packet loss = 0%, RTA = 0.53 ms
e-cg45.pc-xx	DOWN	2012-05-25 19:02:34	0d 11h 52m 35s	CRITICAL - Network Unreachable (10.7.50.11)
e-cg45.pc-22	DOWN	2012-05-25 19:05:24	70d 20h 6m 18s	check_ping: Invalid hostname/address - server
e-cg45.pc-01	UP	2012-05-25 19:01:45	0d 12h 3m 15s	PING OK - Packet loss = 0%, RTA = 0.56 ms
e-cg45.pc-02	UP	2012-05-25 19:02:55	1d 12h 3m 55s	PING OK - Packet loss = 0%, RTA = 0.53 ms
e-cg45.pc-03	UP	2012-05-25 19:02:45	1d 12h 4m 5s	PING OK - Packet loss = 0%, RTA = 0.53 ms
e-cg45.pc-04	UP	2012-05-25 19:02:44	15d 0h 13m 25s	PING OK - Packet loss = 0%, RTA = 0.22 ms
e-cg45.pc-05	UP	2012-05-25 19:03:25	1d 12h 4m 5s	PING OK - Packet loss = 0%, RTA = 1.70 ms
e-cg45.pc-06	UP	2012-05-25 19:03:25	1d 12h 3m 25s	PING OK - Packet loss = 0%, RTA = 0.49 ms

FIGURE 4.4 – Exemple d'affichage de Nagios, vue sur tous les *hostgroups*FIGURE 4.5 – Exemple d'affichage de Nagios, vue sur tous les *hosts* d'un *hostgroup* spécifique

4.1.3 L'IDE NetBeans



FIGURE 4.6 – Logo de NetBeans

NetBeans est un IDE^{3*} *open source* développé par Sun Microsystems permettant le développement en utilisant les langages de programmation tels que Java, JavaScript, PHP*, C, C++, et autres. Il est écrit en Java et fonctionne sous Windows, Mac OS, Linux, Solaris et d'autres plates-formes du moment qu'elles possèdent une JVM^{4*} compatible.

NetBeans permet de développer et déployer rapidement des applications graphiques Swing, des Applets, des JSP/Servlets et des architectures J2EE⁵. Il possède toutes les fonctionnalités recherchées dans un IDE* moderne (coloration syntaxique, refactoring, débogueur, ...) et ajoute, dans le cas d'un développement d'un service Web comme pour ce projet, un support avec la dernière version de GlassFish, permettant notamment de faire des « *deploy on save* », de déployer les applications Web sur un serveur distant et de contrôler le serveur (suivre la sortie de *logs*, démarrer, arrêter le serveur). Il donne accès à une gestion simple du serveur d'application GlassFish qui sera utilisé dans le projet. Sa première version date de 1996 et portait le nom Xelfi. Il est disponible sur son site Internet[3] en version 7.1.2.

4.1.4 Le serveur Web GlassFish

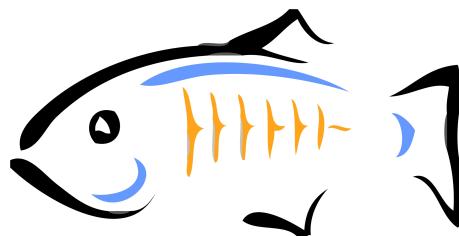


FIGURE 4.7 – Logo de GlassFish

GlassFish est un serveur d'application *open source* développé par Sun Microsystems pour les plates-formes Java EE 5 et 6, et est maintenant maintenu par Oracle Corporation. Il dispose de nombreux outils pour faciliter le développement, le déploiement et la maintenance d'application. Un de ses avantages est qu'il est particulièrement

-
- 3. *Integrated Development Environment*
 - 4. *Java Virtual Machine*
 - 5. *Java Platform, Enterprise Edition*

bien intégré à NetBeans, ce qui permet un déploiement très rapide des applications ainsi qu'une trace d'exécution lors du fonctionnement. GlassFish est disponible sur son site Internet[4] en version 3.1.2.

4.1.5 Le SGBD MySQL

Un des objectifs du projet *Yuukou II* est un archivage temporel des données. Le but étant de générer des statistiques d'utilisation des salles informatiques des différents campus par exemple.



FIGURE 4.8 – Logo de MySQL

MySQL est le Système de Gestion de Base de Données (SGBD) le plus populaire qui fonctionne comme un serveur fournissant un accès multi-utilisateurs à des bases de données de type SQL*. Il a été créé par MySQL AB en 1995, racheté par Sun Microsystems et est maintenant maintenu par Oracle. Il présente les avantages d'être *open source*, gratuit, fiable, rapide et facile à utiliser. Il est très souvent utilisé avec le langage de création de pages Web dynamique PHP*. C'est donc avec cet outil que les données de *Yuukou II* seront archivées. MySQL est disponible sur son site Internet[5] en version 5.5.24.

4.1.6 L'accès aux données des tables

Par souci de performance, c'est-à-dire, limiter les accès multiples à la base de données mais aussi faciliter la manipulation des données lors des traitements, le *design pattern** *Data Access Object* (DAO) a été utilisé. Un *Data Access Object*, ou objet d'accès aux données en français, est un objet qui constitue une représentation en mémoire des données d'une base de données. En Java, un DAO est une classe représentant une table dont les attributs sont les champs de la table comme le montre la figure 4.9. Des classes représentant des listes sont généralement développées pour contenir les DAO et simplifier l'accès aux données.

Lorsqu'il sera nécessaire d'effectuer des traitements répétés sur les données d'une table, même si ces traitements ne portent pas sur la totalité de la table, il est préférable de charger en mémoire la totalité des éléments utiles avec, dans le cas idéal, une seule requête SQL*.

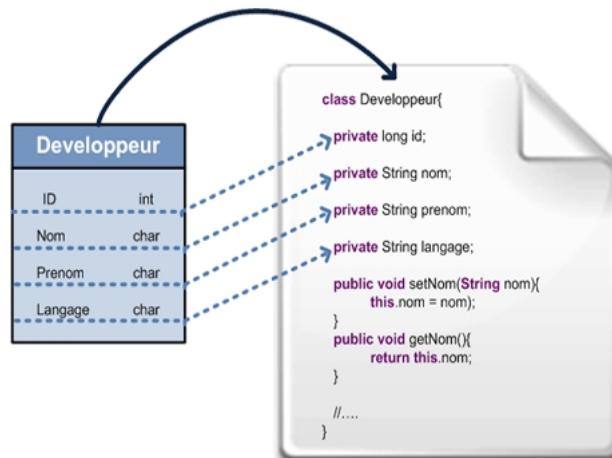


FIGURE 4.9 – Exemple de *mapping* d'une table avec sa classe DAO associée (*source : Cyrille Herby*)

Dans le cadre du projet, les chargements sont effectués lorsqu'un client appelle une méthode du service Web. Les données sont chargées dans une liste, elles sont ensuite traitées puis retournées au client.

4.1.7 Gestion de la base de données

La base de données a évolué tout au long du projet en fonction des fonctionnalités qui ont été demandées. L'annexe C permet d'avoir une description des différentes tables composant la base de données, une vue générale de ce à quoi elle ressemble, et des vues découpées des différentes parties.

La base de données peut être découpée en différentes grandes parties. La partie archivage a pour but de conserver les données dans le temps. Pour ce faire, elle est composée de deux tables centrales de sauvegarde d'informations. La première est la table `yuukou_who`, son but est de simuler la commande UNIX `who`. Cette commande permet de donner tous les utilisateurs connectés sur la machine où elle est exécutée. De ce fait, si elle est portée au niveau de Nagios et du projet, la table contiendra tous les utilisateurs connectés sur un ordinateur et depuis quand ils le sont. La deuxième table est `yuukou_last`, son but est aussi de simuler une commande UNIX, la commande `last`. Cette commande permet de donner la liste de tous les derniers utilisateurs sur la machine sur laquelle elle est exécutée. De ce fait, si elle est portée au niveau de Nagios et du projet, la table contiendra toutes les connexions des utilisateurs sur un ordinateur, l'heure de

début et de fin et la cause de la déconnexion. En fait, elle contiendra les données de la table `yuukou_who` et ajoutant les informations du temps de fin de connexion et la cause de la déconnexion.

La partie logicielle a pour but de stocker la configuration logicielle de chaque salle informatique. La récupération de cette configuration est expliquée au § 4.4.4. Cette partie est composée de plusieurs tables. Une salle informatique peut avoir des groupes de logiciels, `yuukou_rooms_groups` et aussi des logiciels indépendants de tout groupe, `yuukou_rooms_software`. Un groupe de logiciels contient des logiciels, `yuukou_groups_software`. Les descriptions des groupes et logiciels sont contenues dans les tables `yuukou_groups` et `yuukou_software`. Avec ces tables, il est possible de facilement reconstruire la configuration logicielle d'une salle.

La partie emploi du temps a pour but de stocker tous événements des différents emplois du temps de l'Université. La récupération de cette configuration est expliquée au § 4.4.3. Cette partie est composée de trois tables. La première table est celle contenant toutes les salles `yuukou_rooms`, la deuxième contenant tous les événements `yuukou_timetables`. Pour la dernière table `yuukou_mapping_room`, son rôle est de faire le lien entre les noms des salles dans Nagios et les noms des salles qui sont utilisées pour l'emploi du temps. En effet, la génération des emplois du temps est effectuée par un service de l'Université. De ce fait, ce service utilise un système pour nommer les salles qui lui est propre. Il est donc nécessaire de pouvoir relier un événement à une salle à l'aide de cette dernière table.

La partie *mapping* avec les salles a pour but de faire le lien entre une description complète d'un lieu et la description courte qui est donnée dans la description d'une salle dans la table `yuukou_rooms`. Les informations concernant les différents lieux sont contenues dans la table `yuukou_mapping_location`.

La dernière partie, configuration, a pour but de gérer les différents paramètres permettant la gestion de cycle lors du fonctionnement du service Web. Les explications concernant ce qu'est un cycle se trouvent au § 4.3.1.

4.1.8 Le format de retour

Le choix du type de retour est un choix important. En effet, il faut que les données puissent être rapidement transmises, récupérées, traitées et affichées. L'objectif est un affichage instantané des pages sur le navigateur Web ou l'application mobile. De ce fait, plusieurs possibilités ont été envisagées.

« *Parsing* » d'un texte

La première est de transmettre un document texte basique, celui-ci formaté d'une certaine façon afin d'être *parsé* (c'est-à-dire parcourir un flux structuré d'informations pour en extraire les données). Ce qui aurait comme avantage de pouvoir être

récupéré par un client utilisant n'importe quel langage. La figure 4.10 représente un exemple de fichier qu'il est facile de *parser*, il suffit de récupérer les informations se trouvant entre les arobases.

```
1 info1 @ info2 @ info3 @ info4
```

FIGURE 4.10 – Exemple de ligne pouvant être facilement *parsée*

Cependant cette méthode est assez contraignante et oblige à parcourir l'ensemble du fichier alors qu'un seul type d'information peut être utile. Ce qui signifie une perte de temps plus ou moins significative en fonction de la taille du fichier reçu.

Objet Java sérialisable

Une autre méthode pourrait être de transmettre un objet Java dit sérialisable. La sérialisation est un mécanisme fourni par Java permettant de considérer un objet comme une séquence d'octets qui inclut les données de l'objet, le type de l'objet et le type des données. De cette manière, l'objet en question peut être transmis à travers le réseau vers une autre application Java pour le récupérer et le déserialiser, ce qui représente la procédure inverse de la sérialisation. La figure 4.11 donne un exemple de classe Java sérialisable qui pourrait être transmise par le réseau comme un flux de bytes.

```
1 public class Exemple implements Serializable {
2     public String chaine = "Hello";
3     public int i = 0;
4 }
```

FIGURE 4.11 – Exemple de classe Java sérialisable

L'objet contiendrait toutes les données utiles et celles-ci seraient facilement accessibles. Le problème venant avec cette solution est que les objets doivent être récupérés par une JVM*. Ce qui peut s'avérer plus ou moins compliqué voire impossible dans d'autres langages de programmation que Java.

XML

eXtensible Markup Language ou XML est un métalangage pour réaliser du balisage générique permettant de mettre en forme des documents. XML a connu un grand succès depuis sa création. Il est utilisé en particulier pour gérer la configuration, le stockage des données, l'échange d'informations et bien d'autres fonctions encore. La

figure 4.12 donne un exemple de document XML pouvant être facilement converti en une arborescence DOM*.

```
1 <Menu id="file" value="File">
2   <Commands>
3     <MenuItem value="New" action="CreateDoc"/>
4     <MenuItem value="Open" action="OpenDoc"/>
5     <MenuItem value="Close" action="CloseDoc"/>
6   </Commands>
7 </Menu>
```

FIGURE 4.12 – Exemple de document XML

XML offre une solution structurée et très simple à comprendre pour l'envoi d'information par le réseau, cependant, dans les échanges d'informations entre client et serveur, il montre ses limites :

- le chargement et la manipulation deviennent vite compliqués, la plupart du temps, il est nécessaire de *parser* le XML sous forme de DOM*, puis de le parcourir, ce qui requiert l'appel de nombreuses fonctions, sans mentionner le fait que *parser* un document XML est parfois long ;
- la taille des fichiers échangés peut parfois être conséquente du fait de la duplication des données : par nature, le XML ne permet pas de gérer une énorme masse d'informations.

JSON

JavaScript Object Notation ou JSON est un format léger d'échange de données textes. Il utilise la notation des objets JavaScript pour transmettre de l'information structurée. Il est aussi souvent utilisé pour simplifier et alléger les accès à des services Web depuis les navigateurs. La figure 4.13 donne un exemple de fichier JSON reprenant le même arbre que le document XML de la figure 4.12.

```

1 { "Menu" :
2     {
3         "id": "file",
4         "value": "File",
5         "Commands": {
6             "MenuItem": [
7                 {
8                     "value": "New", "action": "CreateDoc"
9                 },
10                {
11                    "value": "Open", "action": "OpenDoc"
12                },
13                {
14                    "value": "Close", "action": "CloseDoc"
15                }
16            ]
17        }
18    }
19 }

```

FIGURE 4.13 – Exemple de fichier JSON

JSON offre l'avantage d'être indépendant du langage utilisé, il est léger et simple à utiliser. C'est un langage d'échange idéal pour beaucoup d'applications.

Choix du format

Le choix du format de retour des données s'est porté sur l'utilisation de JSON. Le but du client, qui demande des informations au service Web, est de recevoir les données le plus rapidement possible afin de pouvoir les afficher dans la foulée. De ce fait, le *parsing* d'un texte contenant des balises à certains endroits s'avère inadapté, notamment pour récupérer un seul type d'information. L'envoi d'un objet sérialisable Java est, quant à lui, trop restrictif. Le choix a donc été entre XML ou JSON.

XML offre un format de données verbeux et prend beaucoup d'espace. De plus, un document XML peut être validé *via* l'utilisation de DTD⁶, document permettant de décrire un modèle de document XML, ou de XML Schema, langage de description de format de document XML permettant de définir la structure et le type de contenu d'un document XML.

JSON offre un format de fichier plus simple que XML, facilement compréhensible. Pour compléter, les fichiers JSON, à arbre égal, seront toujours plus petits que l'équivalent XML et présenteront l'avantage d'être plus rapidement *parsés*.

Au final JSON permet une plus grande rapidité dans les échanges avec un serveur ainsi que sur le temps de *parsing* des fichiers, le tout avec une économie des ressources du fait de sa petite taille. Cependant, les données ne peuvent pas être vérifiées dans un fichier JSON, il demande donc une certaine rigueur dans son écriture du côté serveur ainsi qu'une connaissance de sa structure du côté client.

6. Document Type Definition

4.1.9 Complément d'information sur JSON

Présentation

Une courte description de ce qu'est JSON a déjà été faite au § 4.1.8. Pour reprendre ce qui a déjà été dit, JSON est un format léger d'échange de données qui est apparu en 2002. Il est facile à écrire et comprendre pour des humains. Il est fondé sur un sous-ensemble du langage de programmation JavaScript et est complètement indépendant de tout langage, tout en possédant des conventions familières aux langages descendant du C (C++, C#, Java, JavaScript, ...). Ces propriétés font de JSON un langage d'échange de données idéal.

Cependant, il ne supporte pas les espaces de noms au contraire d'XML. Pour la vérification des données, les schémas JSON ne sont pas très utilisés du fait qu'ils doivent être écrits à la main comme il n'existe aucun outil pour générer un schéma à partir de données JSON. Et concernant la sécurité, il existe la possibilité où des *scripts* malveillants pourraient être dissimulés et exécutés. Il existe des fonctions pour tester les fichiers JSON mais celles-ci ne sont pas vraiment concluantes. La meilleure défense est de connaître à l'avance les points sensibles et de prendre les précautions nécessaires.

Structure

JSON se base sur deux structures de données universelles dans pratiquement tous les langages de programmation modernes :

- une collection de couples nom/valeur ;
- une liste de valeurs ordonnées.

Ces éléments représentent trois types de données :

- un *objet* :
Ensemble de couples nom/valeur non ordonnés. Un objet commence par "{" (accolade gauche) et se termine par "}" (accolade droite). Chaque nom est suivi de ":" (deux-points) et les couples nom/valeur sont séparés par "," (virgule) ;
- un *tableau* :
Collection de valeurs ordonnées. Un tableau commence par "[" (crochet gauche) et se termine par "]" (crochet droit). Les valeurs sont séparées par "," (virgule) ;
- une *valeur* :
Soit une chaîne de caractères entre guillemets, soit un nombre, soit true ou false ou null, soit un objet, soit un tableau. Ces structures peuvent être imbriquées ;
- une *chaîne de caractères* :
Suite de caractères Unicode (zéro ou plus), entre guillemets, et utilisant les échappements avec antislash. Un caractère est représenté par une chaîne d'un seul caractère.

4.2 Communication avec Nagios

Nagios offre la surveillance d'un grand ensemble de machines dans l'Université. Cependant s'il n'y a aucune manière de récupérer ces informations, il n'aurait aucune utilité de son utilisation dans le projet. Ainsi un module existe et son but est de pouvoir interroger Nagios et obtenir ses réponses le plus rapidement et simplement possible.

4.2.1 Le module MKLlivestatus

Le module MKLlivestatus permet un accès immédiat au statut de Nagios ainsi qu'à ses données de *logs*. Avant le module, une façon classique de récupérer les informations importantes était de *parser* le fichier `status.dat`, qui est créé automatiquement par Nagios à chacun de ses cycles. Ce fichier contient toutes les informations concernant les services et les machines que Nagios surveille. *Parser* ce fichier s'avère délicat, surtout pour extraire quelques informations. Il existe bien une manière alternative de récupérer ces informations avec le module NDO. Ce module est directement chargé dans le processus de Nagios et à chaque nouveau cycle, il envoie les mises à jour *via* une *socket* UNIX* à un processus auxiliaire qui se charge de mettre à jour une base de données.

Les avantages étant une mise à jour immédiate de l'information et une simplicité et rapidité d'accès aux bases de données pour les applications. Mais d'un autre coté, la mise en place de ce système est complexe, elle nécessite une administration humaine de la base de données, la consommation CPU est significative juste pour garder la base à jour et son entretien régulier peut bloquer Nagios pendant un moment (cela dépend de la taille de l'infrastructure).

Le module MKLlivestatus reprend le principe de NDO dans l'intégration directe dans le processus Nagios, cependant, la différence de taille est qu'il ouvre juste une *socket* de communication par laquelle les données peuvent être retournées à la demande. La *socket* ouverte par défaut par MKLlivestatus est une *socket* UNIX*, mais il est possible de configurer une *socket* Internet classique sur un port choisi et accessible seulement par certaines machines. Par cette *socket*, il est possible d'envoyer une requête vers une cible spécifique comme un service, une machine, un ensemble de machines, ... La réponse est immédiate et les données renvoyées ont été directement lues dans les structures de données internes de Nagios.

MKLlivestatus offre donc de nombreux avantages :

- consommation CPU non mesurable, juste une petite consommation lors de l'exécution d'une requête ;
- pas d'écritures sur le disque ;
- accès aux données beaucoup plus rapide qu'avec un *parsing* du fichier `status.dat` ou l'exécution de requêtes SQL*.
- pas de configuration, d'administration ou autre, tout est installé automatiquement.

MKLivestatus est disponible sur le site Internet[6] de Mathias Kettner en version stable 1.1.12p7.

4.2.2 Requêtes pour Nagios

Écriture des requêtes avec LQL

LQL, *Livestatus Query Language*, prononcé « LiqueL », est un langage spécialement conçu pour communiquer avec le module MKLivestatus *via une socket*. Il ressemble un peu au langage SQL*.

Chaque requête consiste en :

- une commande commençant par le mot clé GET suivit du nom de la *table* ;
- un nombre arbitraire de lignes d'en-tête consistant en un mot-clé, un double-point et une liste d'arguments ;
- une ligne vide ou une fin de transmission (pour fermer la procédure d'envoi sur la *socket*).

Tous les mots-clés sont sensibles à la casse. La version actuelle de Livestatus donne accès à seize tables dont voici les principales :

- **hosts** : les hôtes surveillés par Nagios ;
- **services** : les services de Nagios, reliés avec toutes les données de la table *hosts* ;
- **hostgroups** : les groupes d'hôtes créés dans Nagios ;
- **servicegroups** : les groupes de services créés dans Nagios.

Les requêtes peuvent aussi être affinées avec la sélection de certaines colonnes, la création de filtres sur les résultats (tous les résultats dont la valeur est égale à deux), compter des résultats, faire des comparaisons avec des expressions régulières, ... Au final, LQL s'avère très flexible d'utilisation. La figure 4.14 donne un exemple de requête permettant de récupérer tous les services avec leur état actuel à 2.

```

1 GET services
2 Columns: host_name description state
3 Filter: state = 2

```

FIGURE 4.14 – Exemple de requête LQL

Les *scripts* Nagios utilisés tout au long du stage sont disponibles en annexe A de ce rapport.

Exécution des requêtes

Comme pour une base de données SQL*, toutes les tables possèdent un nombre de colonnes. Si une requête est passée sans paramètres, toutes les colonnes seront rentrées dans l'ordre alphabétique.

Comme vu au § 4.2.1, il existe deux manières d'accéder à la *socket* de MKLlivestatus. La première méthode est avec une *socket* UNIX*. Pour ce faire, MKLlivestatus fournit pendant son installation un petit utilitaire appelé **unixcat** permettant de communiquer avec une *socket* UNIX*. Sous forme d'une commande shell, cet utilitaire envoie toutes les données lues par l'entrée standard (**stdin**) à la *socket* et écrit sur la sortie standard (**stdout**) tout ce qu'il reçoit de la *socket*. Voici un exemple d'exécution en utilisant la commande **unixcat** :

```
echo 'GET hosts' | unixcat /var/lib/nagios/rw/live.
```

La seconde manière consiste en l'ouverture d'une *socket* Internet sur un port prédéfini. Avec cela, il est possible d'accéder depuis le réseau de l'Université au serveur hébergeant Nagios. Il suffit ensuite d'envoyer *via* la commande UNIX **netcat**, qui permet d'ouvrir des *sockets* serveurs et clientes, un fichier contenant la requête LQL :

```
cat requete | netcat yuukou-ws.wmin.ac.uk 6557.
```

Dans le cadre du projet, l'utilisation d'une *socket* UNIX* en Java nécessite l'installation de bibliothèques de type JNI⁷. Ce sont en fait des bibliothèques qui permettent l'intégration du code écrit en Java avec du code écrit dans un autre langage tel le C ou le C++. Cette installation est plus ou moins complexe et nécessite une certaine configuration du serveur, ainsi, il est plus simple d'ouvrir une *socket* Internet plutôt que d'essayer la configuration d'une *socket* UNIX* avec JNI. De plus, au point de vue rapidité, l'avantage de MKLlivestatus est d'être directement intégré dans Nagios, de ce fait, les requêtes sont traitées immédiatement.

Le tableau 4.1 montre un comparatif des temps d'exécution mesurés avec la commande UNIX **time** pour 100 requêtes. La requête de la figure A.2 se trouvant en annexe A de ce rapport a été utilisée pour effectuer ces tests. Chaque réponse contient 1983 lignes.

7. Java Native Interface

time	unixcat	netcat
Réel	1m44.462s	1m42.129s
Utilisateur	0m0.372s	0m0.432s
Système	0m0.268s	0m0.404s

TABLE 4.1 – Comparatif des temps d'exécution entre la commande `netcat` et la commande `unixcat`

Après 100 requêtes, il est visible qu'utiliser une *socket* UNIX*, pour une exécution système, est presque deux fois plus rapide que l'utilisation d'une *socket* Internet. Cependant, l'application ne nécessite pas l'exécution de nombreuses requêtes simultanées. Une seule sera effectuée à chaque cycle comme décrit au § 4.3.1. De ce fait, l'utilisation d'une *socket* Internet convient dans la récupération des informations de Nagios et évite l'installation de bibliothèques JNI.

Récupération de l'information

Suite à l'exécution de la requête, l'information est retournée par Nagios, elle est ensuite *parsée* en fonction du type de la requête et les informations sont traitées. Nagios retourne les informations sous une forme bien spécifique :

- une ligne correspond à une ligne de la table demandée ;
- la ligne contient toutes les colonnes de la table ou seulement celles demandées dans la requête. Chaque colonne est séparée d'une autre par un ";" (point-virgule).

Des exemples de réponses correspondant aux requêtes de l'annexe A sont disponibles dans l'annexe B.

Nagios possède sa propre façon de nommer une salle ou encore un ordinateur. Les salles dans Nagios sont nommées comme suit :

abrégation du lieu—nom de la salle

Les différentes abréviations sont définies dans le tableau 4.2.

Abréviation	Description
e	<i>Electronics and Computer Science</i>
h	<i>Harrow</i>
l	<i>Little Tichfield Street</i>
m	<i>Marylebone</i>
n	<i>New Cavendish Street</i>
r	<i>Regent Street</i>
w	<i>Wells Street</i>

TABLE 4.2 – Abréviations et lieux utilisés par Nagios pour nommer les salles et ordinateurs

Ainsi, la salle 4111 se trouvant sur le campus de New Cavendish sera identifiée de cette manière : **n-4111**.

Les ordinateurs sont nommée en suivant la même logique :

abréviation du lieu—nom de la salle—type ordinateur—numéro ordinateur

Le type d'ordinateur peut être de deux type : **pc** si c'est un PC ou **mc** si c'est un Macintosh. Ainsi, un ordinateur se trouvant dans la salle **n-4111**, étant un Macintosh et portant le numéro 1 sera identifié comme suit : **n-4111-mc-01**.

4.3 Fonctionnement du service Web

Le service Web peut se décomposer en deux grandes parties. La première consiste en un cycle principal qui a pour but de récupérer les informations de Nagios et de les traiter. La deuxième consiste en de multiples fonctions pouvant être accessibles soit par un administrateur, soit par un utilisateur normal et permettant le retour d'une partie des informations accumulées avec le cycle principal. Le principe étant qu'un client lance de multiples requêtes et puisse récupérer toutes les informations dont il a besoin et ensuite directement les afficher, sans traitements supplémentaires. Suite à ces deux grandes parties, il sera possible d'observer la façon dont les informations sont transmises à un client en utilisant JSON avec des structures par défaut.

4.3.1 Le cycle principal

C'est en quelque sorte le moteur du projet. Il collecte des informations de Nagios, et effectue diverses opérations sur la base de données pour la maintenir à jour. Un cycle est lancé toutes les minutes et dure entre 5 et 7 secondes pour ajouter, retirer ou mettre à jour les informations sur les 1983 ordinateurs surveillés par Nagios ainsi que le nombre variable d'utilisateurs qui sont connectés sur ces dits ordinateurs. La figure 4.15 permet de voir les différentes étapes qui composent le cycle principal du service Web.

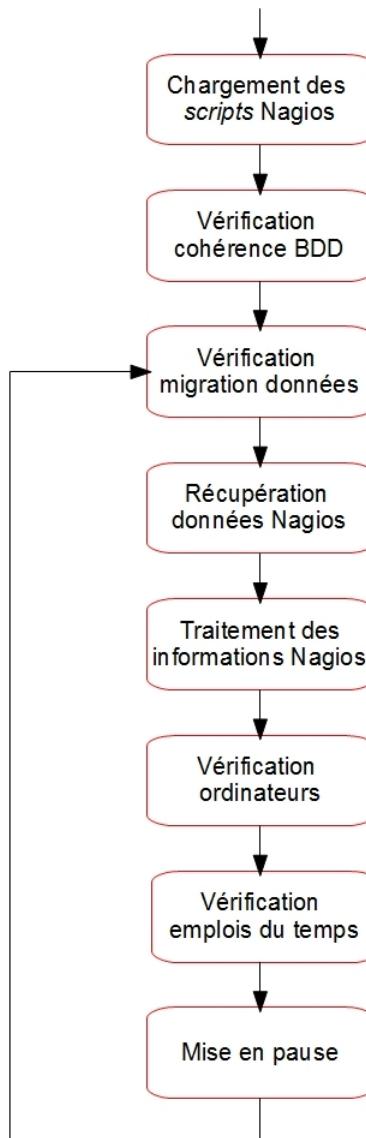


FIGURE 4.15 – Schéma du déroulement du cycle principal du service Web

Dans un premier temps, à la première exécution du cycle, les différents *scripts* écrits en LQL sont chargés en mémoire. Ensuite, une vérification de la cohérence des données contenues dans les tables permettant de gérer l'archivage des données est effectuée. Cette vérification a pour but de s'assurer qu'il n'y a pas des données absurdes de connexion : les données de connexion datées de plus de 6 heures pour la table *yuukou_who* et les données de connexion dont la différence entre la date de début de connexion et la

date de fin de connexion est de plus de 6 heures pour la table `yuukou_last`. Si de telles données sont trouvées, elles sont automatiquement effacées de la table auxquelles elles appartiennent.

L'étape suivante entre dans le début du cycle à proprement parler. Elle a pour objectif de vérifier si des données de la table `yuukou_last` ont besoin d'être déplacées. Des explications sur la migration des données se trouvent au § 4.4.5.

Suite à cette vérification, les *scripts* LQL sont envoyés et les informations récupérées sont transmises à l'étape de traitements des données Nagios.

Cette étape commence par ajouter les machines qui n'existent pas dans la base de données. Ensuite, ce sont les utilisateurs qui sont traités. S'ils n'existent pas dans la base de données, une recherche LDAP* est effectuée pour obtenir les informations les concernant (nom, prénom, photo, rôle). Ils sont ensuite, à leur tour, ajoutés dans la table correspondante. Le couple ordinateur-utilisateur est ensuite ajouté, mis à jour ou retiré des tables d'archivage. Ici aussi, si une donnée de connexion dépasse les 6 heures, elle sera ignorée.

Suite à ce traitement, une vérification est effectuée sur tous les ordinateurs connus. En effet, un ordinateur possède un statut particulier : fonctionnel, éteint ou en attente d'effacement. Ce statut est fixé d'une part avec l'étape précédente des résultats, d'autre part avec l'étape actuelle. Si le statut d'un ordinateur reste éteint pendant une semaine, son statut change et passe à : en attente d'effacement.

Une vérification des emplois du temps est ensuite effectuée. Des explications sur la récupération des emplois du temps se trouvent au § 4.4.3. Cette vérification a pour rôle de mettre à jour les données sur les emplois du temps de façon journalière. De ce fait, tous les jours, les différents emplois du temps sont récupérés automatiquement. À la suite de cette dernière étape, le cycle se met en pause pendant une minute.

Différentes sécurités ont été mises en place pour éviter les appels successifs au cycle principal. En effet, quand un administrateur lance le cycle principal, il lance un processus annexe qui se charge d'effectuer le cycle indéfiniment. Le seul moyen d'arrêter ce processus ou d'éviter les conflits est l'utilisation de la table `yuukou_settings`. Cette table permet de savoir quand un cycle a été effectué pour la dernière fois et s'il est ou non en cours. Agir sur cette table permet donc de stopper le processus mais aussi d'avoir une assurance qualité sur les informations qui seront retournées par les méthodes accessibles pour le client. Si un cycle date de 5 heures, alors il ne faut pas tenir compte des informations retournées.

4.3.2 Les fonctions accessibles

Les fonctions publiques

Ces fonctions sont accessibles par tous les utilisateurs appartenant à l'Université. Elles ne retournent aucune information pouvant porter atteinte aux utilisateurs ou

pouvant donner accès à l'administration du service Web.

healthForRoom (String idRoom) retourne toutes les informations concernant une salle : emploi du temps, nombre d'utilisateurs, description de la salle, ... Cette fonction donne des informations détaillées sur une salle informatique ;

healthForAllRooms () retourne des informations simplifiées concernant l'ensemble des salles surveillées, cette fonction donne une vue d'ensemble de l'occupation des salles ;

getListRooms () retourne la liste de toutes les salles ;

getSitesInformation () retourne les informations concernant les différents campus de l'Université ;

getRoomsType (String typeRoom) retourne la liste des salles en fonction d'un type de machine (PC ou MAC).

Les fonctions privées

Ces fonctions sont un complément aux fonctions publiques. Elles permettent un contrôle sur le cycle principal du service Web et des informations retournées plus complètes.

launchCycle () permet de lancer le cycle du service Web ;

checkConfigHealth () compare les différences entre la table des salles informatiques (*yuukou_rooms*) et la configuration dans Nagios, et retourne les différences ;

who () retourne la liste de tous les utilisateurs connectés, sur quel ordinateur et depuis quand (*yuukou_who*) ;

lastDefault () retourne la table complète *yuukou_last* ;

last (int numberDays) retourne la table *yuukou_last* mais pour un nombre de jours définis ;

searchHistoryUser (String idUser, boolean who, boolean last, int numberLast) retourne l'historique de connexion actuel (*who*) ou passé (*last*) d'un utilisateur, possibilité de fixer le nombre d'informations retournées ;

searchHistoryResource (String idResource, boolean who, boolean last, int numberLast) retourne l'historique d'utilisation en cours (*who*) et passé (*last*) d'un ordinateur, possibilité de fixer le nombre d'informations retournées ;

getGraphWithRequestUsingJson (String rqtLabel, String label, String startTime, String endTime, String addToRqt, int factor) retourne un flux d'octets contenant une image générée à partir de la fréquence d'utilisation des salles informatiques entre un temps de début et de fin (*startTime* et *endTime*), les données servant à construire le graphe sont aussi retournées. L'échelle du graphe peut être fixée avec *factor* : heures, semaines, mois, années. Il est possible de personnaliser la requête (*addToRqt*), de personnaliser la légende (*label*) et de choisir l'élément recherché (*rqtLabel*) ;

healthResourcesReportForAllRooms () retourne l'état des ordinateurs de toutes les salles informatiques surveillées ;
healthResourcesReportForRoom (String idRoom) retourne l'état des ordinateurs d'une salle spécifique ;
actualiseLDAPInfo () permet d'actualiser les données LDAP* des utilisateurs qui sont inconnus dans la base de données ;
isCycleRunning () donne l'état du cycle, s'il est en cours ou non ;
askMaintenance () si le cycle n'est pas en cours, modifie la table de configuration pour stopper le cycle à sa prochaine itération. Tout lancement de cycle est impossible ensuite ;
endMaintenance () met fin à la maintenance, le cycle peut ensuite être relancé ;
isMaintenanceScheduled () permet de savoir si une maintenance est en cours ou non.

4.3.3 Retour des fonctions

Étant dans un cadre de travail collaboratif comme expliqué au § 4.5.2, il est nécessaire d'adopter de la rigueur dans le retour d'information. Encore plus du fait qu'en JSON, le fichier ne peut pas être vérifié comme un fichier XML et son schéma. C'est pourquoi, deux manières spécifiques ont été choisies pour retourner les informations : le cas où les informations ont été correctement transmises et le cas où elles ne l'ont pas été. Des exemples de retours concrets sont disponibles en annexe D.

Retour de fonction pour une exécution correcte

La structure d'un retour d'une fonction suit toujours le même schéma. Le bout de code JSON 4.16 montre un exemple de structure de retour lorsqu'aucune erreur n'est détectée pendant l'exécution d'une méthode du service Web.

```

1 { "JSONState" : "OK" ,
2   "JSONMaintenance" : "NO" ,
3   "JSONLastCycle" : "2012-05-28 17:59:36" ,
4   "JSONContents" : ...
5 }
```

FIGURE 4.16 – Structure générale de retour de fonction JSON sans erreur

Retour de fonction pour une exécution incorrecte

Le bout de code JSON 4.17 montre un exemple de structure de retour lorsqu'une erreur est détectée pendant l'exécution d'une méthode du service Web.

```

1 { "JSONState" : "KO" ,
2   "JSONReason" : "Problem detected"
3 }
```

FIGURE 4.17 – Structure générale de retour de fonction JSON avec erreur

4.4 Fonctionnalités en place

4.4.1 Sécurisation du service Web

Explications

La sécurité est un point important dans l'accès à un service Web. Les données transmises par *Yuukou II* contiennent des informations sur la structure des différents campus mais aussi des informations sur les étudiants (nom, prénom, photo par exemple). Il est donc important de limiter l'accès aux données mais aussi de sécuriser toutes les communications. C'est dans cette optique, qu'il a été décidé de mettre en place le protocole SSL entre le service Web et toute application voulant communiquer avec.

Le protocole *Secure Sockets Layer*, ou SSL, est conçu pour assurer la confidentialité, l'authentification et l'intégrité des données lors des échanges entre un client et un serveur Web, sur différents protocoles de transport (en général HTTP). SSL met en place un chiffrement à clés publiques, c'est-à-dire, une combinaison de deux clés : une clé publique servant au chiffrement des données et une clé privée servant au déchiffrement. Les sites Internet bénéficiant de ce protocole sont reconnaissables du fait de leurs URL⁸ commençant par `https://` où le 's' signifie *secured*.

Fonctionnement

La mise en place d'un tunnel SSL entre client et serveur se fait en 4 étapes :

1. le client fait une demande de connexion sécurisée, il demande donc le certificat garantissant la clé publique du serveur ;
2. le serveur lui envoie son certificat d'authentification délivré par l'autorité de certification, ce certificat contient la clé publique. Une autorité de certification est un organisme chargé de délivrer des certificats et de leur attribuer une date de validité, ainsi que de les révoquer avant la fin de la date en cas de compromission ;
3. si l'autorité de certification et le certificat sont valides, alors le client envoie une clé secrète, créée à partir de la clé publique ;
4. le client et le serveur possède maintenant une clé secrète qui sera encryptée à l'aide d'un algorithme de hachage pour plus de sécurité. Si la connexion est perdue, une

^{8.} Uniform Resource Locator

nouvelle clé secrète sera négociée.

Mise en place

La mise en place d'un protocole SSL sur un service Web *via* NetBeans et GlassFish est très simple une fois que la démarche suivante est connue. Il faut cependant disposer de plusieurs fichiers : une autorité de certification et un couple clé privé-clé publique, tous deux fournis par l'Université. À partir de ces deux fichiers, un certificat de sécurité est généré. Ensuite, l'autorité de certification et le nouveau certificat sont ajoutés à la configuration de GlassFish.

La configuration avec NetBeans consiste à préciser que le service Web utilisera une connexion sécurisée de type SSL sur toutes les méthodes HTTP (Get, Post, ...). Après déploiement, le service Web ne sera accessible qu'avec HTTPS et le certificat sera diffusé.

4.4.2 Génération de graphes d'utilisation

Afin de rajouter des fonctionnalités au service Web, il a été mis en place une méthode permettant de générer des graphes d'utilisation sous forme d'une image. Basiquement, une requête est exécutée sur la table `yuukou_last`, les données sont comptées et la génération d'un graphe est effectuée.

La génération de graphes utilise l'API* RRD4J. Cette API* est en fait l'implémentation Java de RRDtool. RRDtool est un outil de gestion de base de données RRD (*Round-Robin Database*). C'est un outil *open source* permettant le stockage et la génération de graphes à partir de données chronologiques. RRDtool est utilisé avec notamment Nagios et Cacti par exemple.

Dans le cadre du projet, l'utilisation réelle de ce logiciel a été détournée du fait que normalement, les données sont constamment archivées. En fait, à chaque demande de graphes, une nouvelle table RRD est créée, remplie, exploitée pour obtenir une image de graphe puis effacée. Le but n'était pas une intégration parfaite au service Web, mais un aperçu des possibilités qu'offre RRDtool.

La figure 4.18 présente un exemple de graphe d'utilisation des salles informatiques de l'Université à l'aide de l'API* RRD4J pendant une journée.

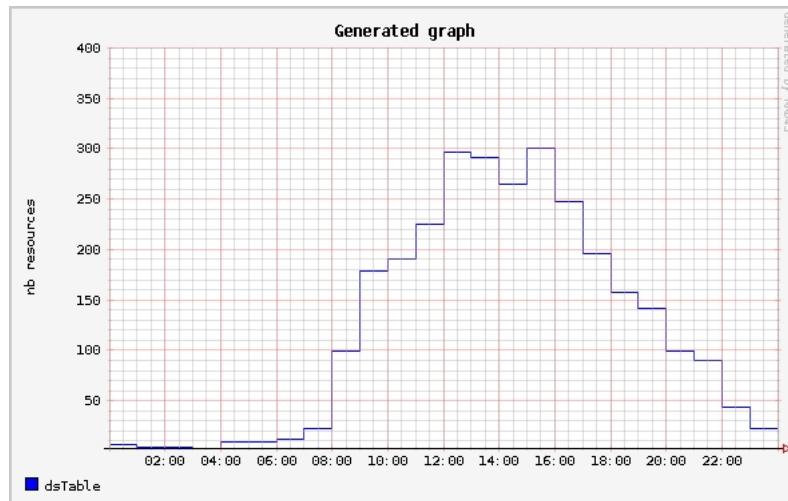


FIGURE 4.18 – Exemple de graphe d'utilisation des salles informatiques de l'Université générée avec RRD4J

4.4.3 Gestion des emplois du temps

Les flux RSS de l'Université

Pouvoir donner le statut d'une salle informatique à un étudiant est un point clé du service Web. En effet, donner la disponibilité d'une salle sans prendre en compte le fait qu'un cours puisse s'y dérouler n'est pas suffisant. Il est ainsi nécessaire de connaître l'occupation d'une salle : le nombre d'ordinateurs libres, mais aussi si l'emploi du temps de cette salle permet de venir y travailler.

L'Université met à disposition 4 flux RSS* contenant toutes les informations sur l'occupation des salles. Ces 4 flux correspondent aux 4 grands campus de l'Université : Harrow, New Canvendish, Marylebone et Regent. Ils présentent l'avantage d'être écrits en XML, de ce fait, le contenu des flux peut être récupéré. Les informations permettant d'identifier un événement sont extraites à l'aide d'un *parseur* de type DOM*. Chaque événement est ensuite ajouté dans la base de données.

Correspondance des salles

Les noms des salles informatiques contenus dans les flux RSS* et ceux présents dans *Yuukou II* ne sont pas les mêmes. Cela vient du fait que les flux existent depuis longtemps, de ce fait, les équipes chargées de leurs maintenance ont adopté leur propre système de nommage des salles. Une table intermédiaire a donc été créée pour faire la relation entre une salle de *Yuukou II* et une salle de l'emploi du temps. Cependant, le nommage ambigu des salles dans les différents flux d'emploi du temps fait que seules

quelques salles ont une correspondance dans le projet. Une discussion a eu lieu à ce sujet dans le but d'adopter un système logique permettant de faire une correspondance plus facile avec toutes les salles surveillées. Mais la mise en place d'un tel système n'est pas prévue dans l'immédiat.

4.4.4 Catalogue logiciels des salles

Un des objectifs de *Yuukou II* est de fournir à un étudiant une liste des logiciels pouvant être utilisés dans chaque salle. Le but étant, qu'un étudiant, voulant travailler dans une salle libre sur un logiciel spécifique comme Visual Studio, puisse trouver rapidement les salles qui correspondent à ses attentes.

Il existe un *MediaWiki* mettant à disposition des membres de l'Université, des informations sur les salles telles qu'une photo de la salle, une description, les responsables et la configuration logicielle de la salle. Un *MediaWiki* est un logiciel permettant de réaliser des sites Internet de type wiki*. Il s'agit d'un système de gestion de contenu de sites Web qui rend les pages Web librement modifiables par tous les visiteurs autorisés. Le site Web *Wikipédia* (http://fr.wikipedia.org/wiki/Wikip%C3%A9dia:Accueil_principal) a été développé avec *MediaWiki*.

Afin de faciliter la gestion de sites Web de ce genre, de nombreuses API* existent. C'est donc en utilisant une de ces API* que la configuration logicielle de chaque salle a été récupérée. Les différents logiciels et groupes de logiciels sont extraits et les différents liens sont construits puis le tout est ajouté à la base de données. Cependant, il est à noter que le *MediaWiki* ne concerne que quelques salles de la *School of Electronics and Computer Science*.

4.4.5 Migration des données

Pendant le développement du projet, vu la quantité de données contenues dans la table *yuukou_last* (environ 50 000 à 60 000 entrées par mois), la question de l'archivage de ses données s'est posée. Une méthode a donc été recherchée en considérant le fait que les données archivées peuvent être réutilisées lors de l'exécution de quelques méthodes par le client.

Une solution a été trouvée : le déplacement à chaque début de mois des données dans une nouvelle table. Chaque nouvelle table est identifiée de la sorte :

yuukou_last_année mois

Avec un exemple pour illustrer, étant en juin 2012, la table archivée correspondant au mois de mai portera le nom suivant :

yuukou_last_201205

De cette manière, les données contenues dans la table `yuukou_last` correspondent aux mois en cours.

4.5 Organisation du travail

Une rigueur dans le déroulement d'un projet permet d'éviter de nombreuses difficultés. C'est d'autant plus vrai quand le projet comprend plusieurs personnes comme c'est le cas pour *Yuukou II*. De ce fait, cette partie permet de comprendre comment s'organisait le travail au sein de l'Université. Dans un premier temps sera abordée la gestion du poste de travail, c'est-à-dire, la stratégie de développement. Ensuite, le déroulement du travail en équipe sera expliqué, le projet étant un travail collaboratif entre une partie service Web et une partie affichage. Et dans la dernière partie sera vue la façon de tester le service Web avant de le considérer comme stable.

4.5.1 Gestion du poste de travail

Afin de développer efficacement, une organisation du poste de travail mais plus généralement du projet en lui-même a été choisie. Cette organisation comporte une machine de développement et deux serveurs comme le montre la figure 4.19.

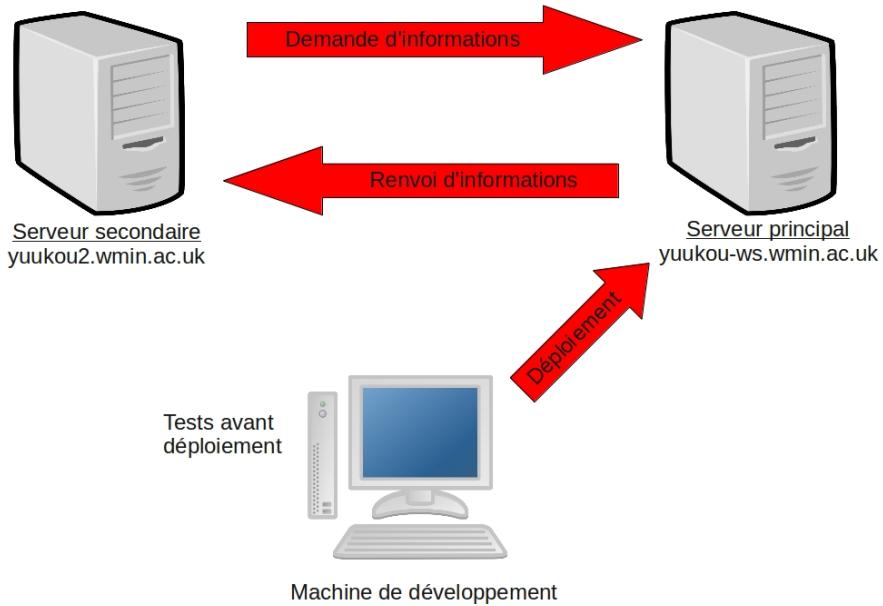


FIGURE 4.19 – Schéma de la gestion du projet

Le serveur principal possède un système d'exploitation Debian version 6.0.5 (*Squeeze*) et héberge le service Web dans une version stable ainsi que tous les outils nécessaire à son fonctionnement comme le serveur de gestion de base de données (SGBD) MySQL, une version de Java à jour, le serveur d'application GlassFish mais aussi Nagios. La faible consommation CPU de Nagios dans la surveillance des machines de l'Université fait qu'actuellement il peut être hébergé sur le même serveur que le service Web. Le serveur principal héberge aussi le gestionnaire de versions subversion (SVN) sur lequel est constamment maintenu le projet. Il est accessible sur le réseau sous le nom de domaine `yuukou-ws.wmin.ac.uk`.

Le serveur dit « secondaire » possède lui aussi le même système d'exploitation Debian et héberge l'application Web permettant de communiquer avec le service Web. Cette application fait régulièrement appel au service Web pour obtenir les informations dont elle a besoin pour afficher correctement ses pages. Il est accessible sur le réseau sous le nom de domaine `yuukou2.wmin.ac.uk`.

La machine de développement possède un système d'exploitation Linux Mint version 11 (*Katya*), tous les outils nécessaires au développement (comme Java, NetBeans, ...), mais aussi un serveur de gestion de base de données (SGBD) MySQL et un serveur GlassFish, tous deux servant à effectuer des tests avant déploiement. À chaque déploiement sur le serveur principal, la version du projet est publiée sur le SVN afin de garantir une sécurité dans le cas d'une régression dans le développement.

4.5.2 Travail en équipe

Le projet *Yuukou II*, comme énoncé dans le § 4.1.1, a été divisé en deux parties. La partie service Web, permettant de remplir et maintenir la base de données ainsi que de mettre à disposition d'un client des méthodes retournant une partie des informations récupérées, et la partie affichage dont le rôle est de présenter les informations à un utilisateur lambda. La partie affichage fut confiée à Yacine MAGHEZZI, étudiant en troisième année de Licence Informatique à l'Université de Franche-Comté de Besançon, ainsi qu'à M. Thierry DELAITRE et Anne-Gaelle COLOM, venus apporter leurs contributions par la suite. Le but de ce projet était, à l'aide du service Web, de permettre un affichage résumant la disponibilité des laboratoires informatiques dans l'Université.

Ainsi, à chaque redéploiement du service Web lors d'une mise à jour importante ou d'un ajout de fonctionnalités, il était important de les tenir informé des différents changements qui ont été effectués. C'est dans le cadre de ce travail en équipe que le principe des structures de retour standard a été adopté (voir § 4.3.3). En effet, un minimum de rigueur dans le développement permettait de ne pas avoir de surprises à chaque étape du projet.

Cependant, la majeure partie de la communication se faisait selon les besoins des différents développeurs. Avant l'arrivée de M. MAGHEZZI, les fonctionnalités et les informations qu'elles retournaient n'étaient que limitées. Cela étant dû au fait qu'elles

n'étaient exploitées par personne. Après son arrivée et sa première connexion au service Web, M. MAGHEZZI a commencé à exprimer des demandes concernant les éléments qu'il aimeraient voir apparaître. C'est à ce moment que le dialogue s'est vraiment mis en place.

Étant dans le même bureau que M. MAGHEZZI, la communication se déroulait principalement oralement, il était informé directement et même en avance par rapport au déploiement d'une nouvelle version du service Web. Pour les autres acteurs du projet, un *mail* était envoyé, celui-ci contenant très souvent des structures de retour JSON contenant le schéma d'une structure de retour de tous les cas possibles pour une méthode spécifique du service Web.

4.5.3 Tests du service Web

Le déploiement du service Web sur le serveur principal impose d'avoir une version stable et sans bogue. De ce fait, comme expliqué au § 4.5.1, le service Web est une première fois déployé sur la machine de développement. Afin de vérifier son bon fonctionnement, un service Web client a été mis en place. Ce client ouvre une connexion sécurisée, puis donne accès aux méthodes qu'il est possible de tester sur le service Web. Il agit comme l'application permettant l'affichage des données du service Web que M. MAGHEZZI a créé, à la différence que les résultats sont affichés en texte brut.

Le client est donc développé en tant que service Web client. Les pages Web sont générées grâce à des *Servlet** Java. Il est à noter la présence d'un *Servlet* permettant la gestion du cycle principal. Il offre donc un contrôle sur le cycle : démarrage, mise en maintenance et fin de la maintenance.

4.6 Exploitation du service Web

Le service Web est directement interrogé par la partie affichage du projet. Cette partie affichage se présente sous la forme d'une interface homme-machine (IHM) développée grâce aux technologies JSP, Servlet et jQuery Mobile. Cette application présente, en fonction de différents niveaux de sécurité, différentes vues.

Un utilisateur normal, comme un étudiant, aura une vue sur la disponibilité des salles informatiques. Il pourra voir toutes les salles, leur disponibilité (ordinateurs disponibles, occupés, éteints, salle disponible ou occupée par un événement de l'emploi du temps), les filtrer par lieu, ou encore par type d'ordinateur (PC ou MAC).

Un membre des équipes techniques de l'Université (*staff*) aura une vue sur la disponibilité des salles informatiques en plus d'une vue sur le matériel. Il aura accès à des graphiques, des résumés sur la disponibilité des salles mais aussi à des informations supplémentaires sur les salles (informations sur les utilisateurs connectés et historiques des ordinateurs).

La figure 4.20 donne trois vues différentes sur ce que l'application d'affichage des données du service Web propose. Les figures sont des captures d'écrans prises à l'aide d'un émulateur d'iPhone. Il est à noter que les figures suivantes ont été prises en s'authentifiant avec un compte utilisateur. Comme précédemment annoncé, les *staffs* bénéficieront, dans certaines pages, de plus d'options.



FIGURE 4.20 – Exemples de captures d'écrans sur un iPhone de l'application affichant les informations du service Web

La figure (a) montre une vue généraliste de la liste des salles informatiques dans l'Université. Sur la gauche, le statut et le lieu de chaque salle sont affichés. Le statut permet de savoir si un cours se déroule dans la salle ou non comme le montre la figure (c). Des couleurs permettent de se rendre rapidement compte de la disponibilité des salles. Information supplémentaire en l'image qui est affichée : le symbole de Windows est présent afin de spécifier que la salle est une salle contenant des PC, le symbole de Macintosh est présent pour les salles contenant exclusivement des MAC. Un récapitulatif sur la droite permet de voir le nombre total d'ordinateurs, ceux qui sont occupés, libres ou éteints.

En sélectionnant une salle, une page contenant la description de cette salle s'affiche comme le montre la figure (b). Cette description contient l'image de la salle, un commentaire sur la salle, le récapitulatif du statut et de la disponibilité de la salle comme pour la figure (a), la configuration logicielle de la salle, un lien vers Google Map pour situer l'emplacement dans Londres et des informations complémentaires (responsables de la salle, adresse complète, ...).

La figure (c) montre une vue avec une recherche des salles occupées dans l'Uni-

versité. Des informations apparaissent sur l'évènement qui se déroule actuellement dans les salles et la période d'occupation des salles. Sont toujours disponibles le type d'ordinateurs dans la salle et leur disponibilité.

4.7 Problèmes rencontrés

Durant toute la durée du projet, de nombreux problèmes ont été rencontrés. Ils étaient principalement dus au manque de connaissances dans le domaine visé.

Le SSL a été difficilement mis en place dans les premiers temps. En effet, le manque de connaissances dans les services Web et dans les outils NetBeans et GlassFish ont été un frein dans sa mise en place. De plus, la documentation sur le Web est plus ou moins évasive et les méthodes sont diverses pour sécuriser un service Web. Cependant, avec l'aide de l'Université, notamment en fournissant les différents certificats et les instructions, le protocole a pu être mis en fonction.

Il est apparu pendant le stage que les différents services de l'Université n'appliquaient que rarement les mêmes normes. Cela s'est fait ressentir dans la mise en place de la gestion des emplois du temps. Effectivement, les noms des salles entre *Yuukou II* et ceux dans les emplois du temps n'avaient rien à voir. Certes, certaines pouvaient être devinées, elles ont été de ce fait traitées. Mais pour d'autres cela s'avérait impossible, surtout si l'on considère que plusieurs personnes s'occupent des emplois du temps et que chacune de ces personnes applique chacun une méthode de nommage plus ou moins personnelle. Une réunion a eu lieu pendant le stage avec l'équipe en charge afin d'aborder ce point. Au final, un nouveau système d'identification des salles devrait être mis en place, mais le service étant occupé, il n'y a pas de date dans la mise en fonction d'une telle mesure. Lorsque ce système sera appliqué, la liste des nouveaux noms de salles sera communiquée, ce qui rendra cette partie pleinement fonctionnelle.

5 Bilan

5.1 Bilan du travail réalisé

Ce bilan résumera le travail réalisé durant le stage. Dans une première partie, un récapitulatif du projet sera donné, avec notamment un graphique afin de montrer les fonctionnalités reprises de *Yuukou* ainsi que les différentes fonctionnalités qu'apporte *Yuukou II*. Ensuite le calendrier des 16 semaines de stage, avec les différentes tâches effectuées semaine après semaine. Il sera suivi par une partie donnant ce qu'apporte le projet à l'Université, les avantages qu'elle peut en tirer. Enfin une dernière partie traitera des améliorations qu'il est possible d'apporter au service Web afin d'accroître ses performances et ses fonctionnalités.

5.1.1 Récapitulatif du projet

La figure 5.1 permet de donner une vue d'ensemble sur les principales fonctionnalités dont le projet dispose. Les cadres pleins représentent des fonctionnalités dont les principes ont été repris de *Yuukou*, les cadres en pointillés, quant à eux, représentent les nouvelles fonctionnalités qu'apporte *Yuukou II*.

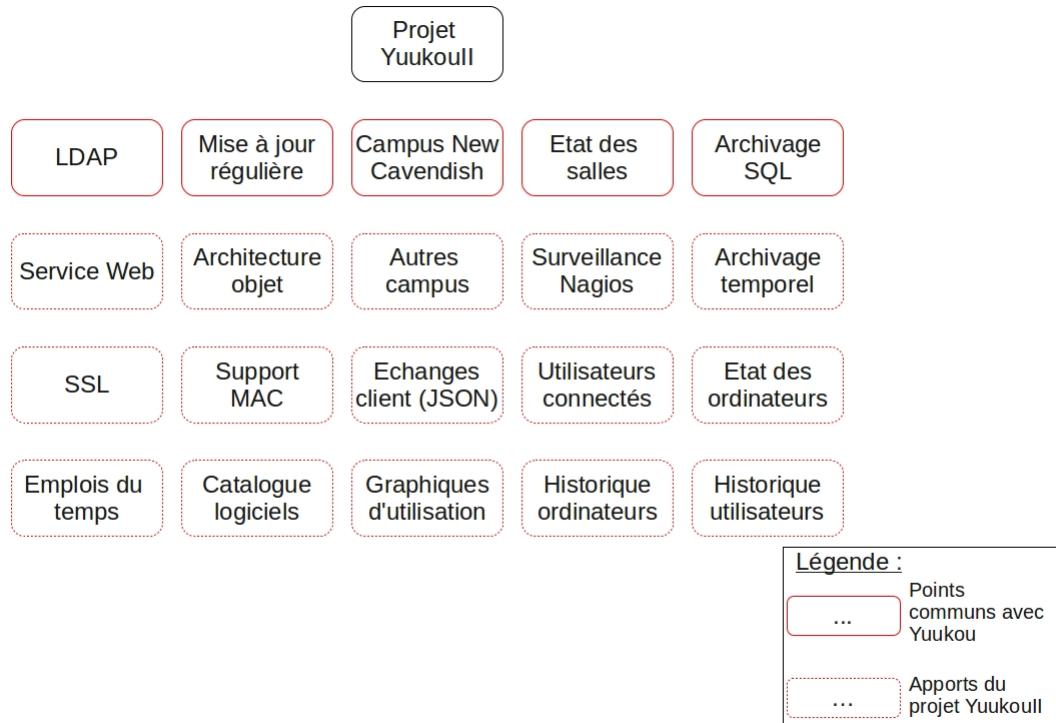


FIGURE 5.1 – Récapitulatif des fonctionnalités reprises de *Yuukou* et les nouvelles de *Yuukou II*

5.1.2 Calendrier du stage

Voici le calendrier du stage d'une durée de 16 semaines commençant le 13 février 2012 et terminant le 4 juin 2012. Certains points ont demandé plus de travail que d'autres, comme la conception du cycle principal du service Web ou encore la mise en place du SSL qui a posé quelques difficultés.

- semaine 1** : arrivée et découverte du projet, premier entretien avec l'équipe technique et leurs attentes ;
- semaine 2** : découverte des services Web et choix des outils utilisés ;
- semaines 3 et 4** : Conception de la base de données et implémentation du cycle principal de l'application ;
- semaine 5** : mise en place de l'emploi du temps et début mise en place de SSL ;
- semaine 6** : mise en place du système JSON et fin de mise en place de SSL ;
- semaines 7 et 8** : développement des méthodes Web ;
- semaine 9** : mise en place de l'architecture finale et développement des méthodes Web ;

semaine 10 : début de rédaction du rapport (présentation et sujet) ;
semaine 11 : développement des méthodes Web ;
semaines 12 et 13 : récupération du catalogue de logiciels de MédiaWiki ;
semaine 14 : ajout de méthodes Web et amélioration du système de chargement de la base de données ;
semaine 15 : rédaction du rapport de stage et maintenance des fonctionnalités du service Web ;
semaine 16 : fin d'écriture du rapport de stage.

5.1.3 Bilan pour l'Université

Le service Web est fonctionnel à l'Université. Il dispose d'un cycle principal permettant la récupération des données de Nagios et le stockage dans la base de données. De nombreuses fonctionnalités ont été développées pour étendre les possibilités du service Web. Une sécurisation des communications avec la mise en place du protocole SSL. Une gestion des emplois du temps en récupérant quotidiennement les différents emplois du temps des campus de l'Université. Une gestion de la configuration logicielle des salles informatiques avec l'extraction des données présentes sur le *MediaWiki* de la *School of Electronics and Computer Science*. Une génération d'un graphe d'utilisation des salles informatiques dans une période donnée.

Le service Web dispose de nombreuses fonctions permettant à un client de récupérer juste les informations dont il a besoin concernant la disponibilité des salles en temps réel. Mais aussi les historiques dans le temps des connexions. Il propose aussi une gestion du cycle principal permettant de l'arrêter et de le remettre en marche.

L'utilisation de JSON permet la création de clients indépendamment du langage employé. Une fois la structure du fichier retourné comprise, il est facile d'extraire les données voulues et de les afficher.

Ajouté à cela la documentation Java (*javadoc*) du service Web afin que dans le futur différentes améliorations, parmi celles exprimées au § 5.1.4, soient mises en place.

5.1.4 Améliorations possibles

Le projet *Yuukou II* est loin d'être fini même si fonctionnel à l'heure actuelle. Il reste diverses fonctionnalités à développer. Une de ces fonctionnalités serait de rajouter une sécurité lors de l'accès aux méthodes du service Web par le client. En effet, il serait intéressant que le client s'authentifie. De ce fait, deux rôles pourraient être définis : administrateur et utilisateur. Avec cette méthode, un utilisateur ne pourrait jamais se servir d'une méthode dite privée car actuellement, si un client développe son propre programme pour accéder au service Web *Yuukou II*, une fois le protocole SSL en place, rien ne l'empêche d'utiliser les fonctions qu'il veut.

Des améliorations peuvent aussi être portées sur la façon dont RRDtool a été mis en place. Il ne s'agit que d'un essai pour tester les possibilités, mais il serait intéressant de l'intégrer pleinement au service Web et surtout au cycle principal d'exécution. De ce fait, les données seraient enregistrées en temps réel et les requêtes de génération de graphe pourraient gagner en rapidité d'exécution. De plus, cette méthode rendrait une utilisation normale de RRDtool au service Web.

La suite concernerait les attentes pour les emplois du temps et le catalogue logiciels. En effet, la gestion des emplois du temps est susceptible d'évoluer vers une notation plus lisible des salles par le service responsable de la création des emplois du temps. Cela permettrait un suivi de toutes les salles informatiques au lieu de seulement certaines actuellement situées à la *School of Electronics and Computer Science*. Pour le catalogue de logiciels, le principe est un peu le même. Ne concernant que la *School of Electronics and Computer Science*, il serait intéressant de trouver une façon simple de récupérer les informations sur la configuration logicielle des autres salles. Par exemple, si une partie de la base de données permettant de construire cette configuration, en amont, pouvait être disponible, l'extraction des données s'en verrait simplifiée et plus complète du fait qu'elle serait effective pour toutes les salles surveillées par Nagios.

Il pourrait être aussi implanté une série de petites fonctions permettant d'effectuer des modifications d'informations, quand le service Web est en maintenance, sur les différentes tables. Par exemple l'ajout d'une salle avec l'aide d'une interface Web.

Nagios surveille actuellement les salles informatiques de l'Université, cependant, il est possible de lui faire surveiller les imprimantes et d'ajouter les services adaptés comme la gestion du niveau d'encre et du papier. Un étudiant recherchant une imprimante de libre obtiendrait les salles correspondantes. Avec cela, il serait possible d'avoir un statut complet d'une salle informatique.

Il serait aussi intéressant d'ajouter des fonctions au service Web pour retourner les informations sur la surveillance des salles sous forme d'un flux RSS* pouvant être exploité de différentes façons par un client.

5.2 Bilan personnel

D'un point de vue humain, ce stage m'a beaucoup apporté, spécialement au niveau de la langue. Je me suis rendu compte combien il est difficile au début de communiquer convenablement avec quelqu'un. Le même cycle se répète : on essaye de traduire ce qu'on entend, ensuite on réfléchit à la réponse en français, on la traduit rapidement et on la dit plus ou moins bien. C'est compliqué de se défaire de ce cycle mais après quelques mois, j'ai pu ressentir beaucoup plus de facilité à communiquer avec les membres du CPC.

De plus, je suis vraiment content d'avoir pu effectuer mon stage dans une ville si cosmopolite que Londres. J'ai pu y découvrir une vie un peu différente de celle que

je connaissais. Durant les quatre mois, j'ai résidé dans une résidence étudiante où j'ai partagé une chambre avec un Croate, de ce fait, je n'avais pas d'autre choix que de parler anglais avec lui. C'était d'autant plus le cas que la majorité des résidents n'étaient pas anglais, mais venaient d'un peu partout dans le monde. Chose qui, je pense, m'a beaucoup fait progresser notamment dans la conversation, tout en gardant quand même un fort accent français.

Concernant le cadre du stage, j'ai beaucoup apprécié la prise en charge de M. DELAITRE qui s'est beaucoup impliqué dans ce projet, tant dans la partie service Web que dans la partie affichage. J'ai réellement aimé la liberté d'action qu'il m'a laissé dans le projet tout en fixant les objectifs qu'il voulait voir accomplis.

La collaboration avec les différents acteurs du projet d'affichage s'est très bien passée, notamment avec M. Yacine MAGHEZZI comme nous étions dans le même bureau. Nous avons dû beaucoup discuter l'un avec l'autre concernant le retour d'informations du service Web. Mon objectif était vraiment de faciliter un maximum le travail des autres développeurs pour qu'ils puissent se concentrer essentiellement sur la partie affichage.

D'un point de vue pédagogique, ce stage fut vraiment très riche en nouvelles connaissances. J'ai pu y découvrir les services Web que je ne connaissais pas auparavant ainsi que des outils qui m'étaient totalement inconnus comme NetBeans et GlassFish. J'avoue avoir des appréhensions quand on me parle de technologies du Web mais le projet que m'a confié M. Thierry DELAITRE m'a vraiment passionné du début à la fin du stage. J'ai pu réutiliser beaucoup de mes acquis avec le développement Java, les interactions avec les bases de données ou encore la manipulation de fichiers XML.

6 Conclusion

Ce stage de deuxième année de Master Informatique est une chance de se projeter dans le contexte d'un travail réel en entreprise en travaillant avec des technologies du marché. La liberté dont j'ai bénéficié durant toute la durée du stage m'a permis de découvrir de nouvelles technologies et le concept des services Web notamment. J'ai pu remettre en question certains de mes choix et exercer un avis critique sur mon travail pour au final fournir un service Web fonctionnel. Celui-ci comprend un cycle principal permettant la récupération d'informations issues de Nagios pour les stocker dans une base de données, ainsi que de multiples fonctions permettant à un client d'extraire une partie des données qui peuvent lui être utiles afin de pouvoir vérifier la disponibilité des salles informatiques dans l'Université de Westminster.

Ce travail a été aussi une chance de travailler en équipe avec différentes personnes. La communication fut très importante tout au long du développement afin de pouvoir se mettre d'accord et arriver à un résultat tant pour la partie service Web que pour la partie affichage. Au final, l'application cliente est capable d'afficher toutes les informations concernant la disponibilité des salles en prenant en compte l'emploi du temps. Elle fournit aussi aux membres des équipes techniques des informations supplémentaires comme des graphiques ou des résumés sur la situation globale des salles informatiques de l'Université.

Le stage fut aussi une bonne opportunité d'améliorer mon anglais. Étant dans un environnement en grande partie anglophone, ma compréhension et ma pratique de la langue n'en ont été que meilleures. Ce fut aussi la première fois que je venais à Londres, j'ai donc pu découvrir la ville ainsi que les différents modes de vie qui la composent.

Je suis au final très satisfait de ce stage qui m'a apporté beaucoup de connaissances ainsi qu'une meilleure pratique de l'anglais. L'application est disponible actuellement pour les membres de l'Université. Dans le cas où des améliorations, des adaptations ou des extensions devaient être effectivement implantées dans le service Web, j'ai tout lieu de penser que la prise en main du programme serait facile, du fait, entre autres, de la présence de la documentation Java.

Bibliographie

- [1] Site de documentation sur JAX-WS. <http://jax-ws.java.net/>.
- [2] Site de Nagios. <http://www.nagios.org/>.
- [3] Site de NetBeans. <http://netbeans.org/>.
- [4] Site de GlassFish. <http://glassfish.java.net/>.
- [5] Site de MySQL. <http://www.mysql.com/>.
- [6] Site de Mathias Kettner mettant à disposition le module MKLivesstatus.
<http://mathias-kettner.de/>.
- [7] Site de RRD4J. <http://code.google.com/p/rrd4j/>.
- [8] Site de RRDtool. <http://oss.oetiker.ch/rrdtool/>.

Glossaire

API (*Application Programming Interface*)

Interface qui a pour objet de faciliter le travail d'un programmeur en lui fournissant les outils de base nécessaires à tout travail à l'aide d'un langage donné. Elle constitue une interface servant de fondement à un travail de programmation plus poussé.

Design pattern (*Patron/modèle/motif de conception*)

Dans un contexte de programmation objet, un *design pattern* décrit une organisation pratique de classes objets. Le but étant la réutilisation et la maintenance du code.

DOM (*Document Object Model*)

API pour les documents HTML ou XML qui fournit une structure arborescente de représentation du document permettant la modification de son contenu.

Framework

Ensemble de fonctions facilitant la création de tout ou partie d'un système logiciel, ainsi qu'un guide architectural en partitionnant le domaine visé en modules. Un *framework* est habituellement implanté à l'aide d'un langage à objets, bien que cela ne soit pas strictement nécessaire : un *framework* objet fournit ainsi un guide architectural en partitionnant le domaine visé en classes et en définissant les responsabilités de chacune ainsi que les collaborations entre classes.

IDE (*Integrated Development Environment*)

Programme regroupant un ensemble d'outils pour le développement de logiciels. En règle générale, un IDE regroupe un éditeur de texte, un compilateur, des outils automatiques de fabrication et souvent un débogueur.

JVM (*Java Virtual Machine*)

Environnement d'exécution indépendant de la plate-forme permettant la conversion d'un *bytecode* Java (résultat de la compilation d'une classe Java) en langage machine puis son exécution.

LDAP (*Lightweight Directory Access Protocol*)

Protocole standard permettant de gérer des annuaires. Il permet l'accès à des bases d'informations sur les utilisateurs, les périphériques et autres composants réseau par l'intermédiaire de protocoles TCP/IP.

Microsoft Active Directory

Service d'annuaires LDAP, mis au point par Microsoft, pour les systèmes d'exploitation Windows.

Novell eDirectory

Service d'annuaires LDAP, mis au point par l'entreprise Novell, permettant de gérer de façon centralisée l'accès aux ressources des serveurs et ordinateurs au sein d'un même réseau.

Perl

Langage de programmation créé en 1987 reprenant des fonctionnalités du langage C et des langages de *scripts* sed, awk, shell. C'est un langage interprété adapté dans le traitement et la manipulation de fichiers textes.

PHP (*Personal Home Page* ou *PHP : Hypertext Preprocessor*)

Langage de *scripts* principalement utilisé pour produire des pages Web dynamiques.

RSS (*Rich Site Summary*)

Un flux RSS est un fichier texte particulier dont le contenu est produit automatiquement en fonction des mises à jour d'un site Web. Les flux RSS sont souvent utilisés pour présenter les titres des dernières informations consultables en ligne dans le cas des sites d'actualité par exemple. Les flux RSS s'appuient sur le langage XML pour afficher leurs données.

Servlet

Programme Java qui s'exécute dynamiquement sur un serveur Web et permet l'extension des fonctions de ce dernier (communication avec un serveur LDAP par exemple). Les *Servlets* permettent la gestion de requêtes HTTP et de fournir au client une réponse HTTP et ainsi de créer des pages Web dynamiques.

Socket UNIX

Interface de communication de données permettant l'échange d'informations entre des processus s'exécutant sur un même système d'exploitation. Ces *sockets* ont l'avantage d'être plus rapide que les *sockets* Internet classiques utilisant un numéro de port et accessible depuis le réseau.

SQL (*Structured Query Language*)

Langage informatique normalisé permettant d'effectuer des opérations sur des bases de données.

TCP/IP (*Transmission Control Protocol / Internet Protocol*)

Ensemble de protocoles utilisés pour le transfert de données sur Internet.

Wiki

Site Web dont les pages sont modifiables par les visiteurs afin de permettre l'écriture et l'illustration collaboratives des documents numériques qu'il contient.

WOL (*Wake On Lan*)

Technique permettant de démarrer un ordinateur éteint à partir d'un réseau. Pour un *Wake On Lan*, on parle de réseau local, pour un *Wake On Wan*, on parle d'Internet.

Workflow

Traduction littérale « flux de travail », c'est la modélisation et la gestion informatique de l'ensemble des tâches à accomplir et des différents acteurs impliqués dans la réalisation d'un processus métier (ou opérationnel).

Annexes

A Fichiers LQL Nagios

Cette annexe décrit sommairement les fichiers LQL permettant la récupération des informations de Nagios. Ces informations sont ensuite traitées par le service Web qui se charge de remplir ou actualiser la base de données en conséquence.

Récupération des salles

La requête A.1 permet d'interroger Nagios afin de récupérer les salles qu'il surveille. Ces salles sont appelées *hostgroups* et contiennent des machines appelées *hosts*. Parmi ces *hostgroups*, les imprimantes, serveurs et autres ressources sont exclues pour ne retenir que les salles. La *socket* Nagios retournera alors le nom de la salle, le nombre de machines qu'elle contient, et la liste des machines qui font partie du groupe.

```
1 GET hostgroups
2 Columns: name num_hosts members
3 Filter: name !~~ printer
4 Filter: name !~~ server
5 Filter: name !~~ all
6 Filter: name !~~ ecs_sunrays
7 And: 4
```

FIGURE A.1 – Code LQL de récupération des salles que surveille Nagios

Récupération des machines

La requête A.2 permet d'interroger Nagios afin de récupérer toutes les machines qui peuvent posséder un utilisateur de connecté. En fait, il est demandé la récupération de tous les services ayant accès à l'information *check_whoisloggedin*. Cette information permet de récupérer l'identifiant de la personne connectée à une machine, si tant est qu'une personne y est connectée. Il y a pour chaque machine, un service portant ce nom, cela revient donc à demander tous les ordinateurs. La *socket* Nagios retournera alors le

nom de la machine, son adresse IP, le `host_groups` donc le nom de la salle à laquelle elle appartient, son état et enfin l'utilisateur connecté s'il y en a un.

```
1 GET services
2 Columns: host_name host_address host_groups state plugin_output
3 Filter: description = check_whoisloggedin
```

FIGURE A.2 – Code LQL de récupération des machines que surveille Nagios

Récupération de tous les utilisateurs connectés

La requête A.3 permet d'interroger Nagios afin de récupérer seulement la liste des utilisateurs connectés sur les machines sous surveillance. Il est demandé la récupération de tous les services parmi lesquels il n'est gardé que ceux sur lesquels un utilisateur peut se connecter. De plus, les messages de Nagios concernant un éventuel problème dans la récupération de l'information sont écartés. Seuls les identifiants utilisateurs "corrects" sont gardés, c'est-à-dire ceux n'étant pas des messages d'erreurs de Nagios comme "(null)" par exemple qui signifie qu'un utilisateur est connecté sur une session UNIX et que son identifiant n'est pas connu.

```
1 GET services
2 Columns: plugin_output
3 Filter: description = check_whoisloggedin
4 Filter: plugin_output !~ ^No user
5 Filter: plugin_output !~ ^Cannot
6 Filter: plugin_output !~ ^\(`
```

FIGURE A.3 – Code LQL de récupération des utilisateurs connectés sur les machines que surveille Nagios

B Retours de requête LQL

Cette annexe donne une vue des informations qui sont retournées par une requête LQL. Les informations se présentent toujours sous la même forme : une ligne qui est composée des différentes colonnes demandées dans la requête. Les colonnes sont séparées par un ";" (point-virgule).

Retour lors d'une récupération des salles

La réponse B.1 correspond à l'exécution de la requête A.1 sur le serveur contenant Nagios.

```
e-cg24;21;e-cg24-pc-01,e-cg24-pc-02,e-cg24-pc-03,e-cg24-pc-04,e-cg24-pc-05,e-cg24-pc-06,e-cg24-pc-07,e-cg24-pc-08,e-cg24-pc-09,e-cg24-e-clg41;1;e-clg41-pc-10
e-clg42;1;e-clg42-pc-zz
e-clg45;1;e-clg45-pc-zz
e-clg46;21;e-clg46-pc-01,e-clg46-pc-02,e-clg46-pc-03,e-clg46-pc-04,e-clg46-pc-05,e-clg46-pc-06,e-clg46-pc-07,e-clg46-pc-08,e-clg46-pc-e-clg49;11;e-clg49-pc-01,e-clg49-pc-02,e-clg49-pc-03,e-clg49-pc-04,e-clg49-pc-05,e-clg49-pc-06,e-clg49-pc-07,e-clg49-pc-08,e-clg49-pc
e-clg50;21;e-clg50-pc-01,e-clg50-pc-02,e-clg50-pc-03,e-clg50-pc-04,e-clg50-pc-05,e-clg50-pc-06,e-clg50-pc-07,e-clg50-pc-08,e-clg50-pc
e-clg51;1;e-clg51-pc-zz
e-clg52;21;e-clg52-pc-01,e-clg52-pc-02,e-clg52-pc-03,e-clg52-pc-04,e-clg52-pc-05,e-clg52-pc-06,e-clg52-pc-07,e-clg52-pc-08,e-clg52-pc
e-clg53;1;e-clg53-pc-zz
e-nl108;21;e-nl108-pc-01,e-nl108-pc-02,e-nl108-pc-03,e-nl108-pc-04,e-nl108-pc-05,e-nl108-pc-06,e-nl108-pc-07,e-nl108-pc-08,e-nl108-pc
e-nl108;21;e-nl108-pc-01,e-nl108-pc-02,e-nl108-pc-03,e-nl108-pc-04,e-nl108-pc-05,e-nl108-pc-06,e-nl108-pc-07,e-nl108-pc-08,e-nl108-pc
e-nl111;21;e-nl111-mc-01,e-nl111-mc-02,e-nl111-mc-03,e-nl111-mc-04,e-nl111-mc-05,e-nl111-mc-06,e-nl111-mc-07,e-nl111-mc-08,e-nl111-mc
e-nl112;21;e-nl112-mc-01,e-nl112-mc-02,e-nl112-mc-03,e-nl112-mc-04,e-nl112-mc-05,e-nl112-mc-06,e-nl112-mc-07,e-nl112-mc-08,e-nl112-mc
e-nl113;21;e-nl113-mc-01,e-nl113-mc-02,e-nl113-mc-03,e-nl113-mc-04,e-nl113-mc-05,e-nl113-mc-06,e-nl113-mc-07,e-nl113-mc-08,e-nl113-mc
e-nl108;21;e-nl108-pc-01,e-nl108-pc-02,e-nl108-pc-03,e-nl108-pc-04,e-nl108-pc-05,e-nl108-pc-06,e-nl108-pc-07,e-nl108-pc-08,e-nl108-pc
e-nl109;21;e-nl109-pc-01,e-nl109-pc-02,e-nl109-pc-03,e-nl109-pc-04,e-nl109-pc-05,e-nl109-pc-06,e-nl109-pc-07,e-nl109-pc-08,e-nl109-pc
e-nl110;21;e-nl110-pc-01,e-nl110-pc-02,e-nl110-pc-03,e-nl110-pc-04,e-nl110-pc-05,e-nl110-pc-06,e-nl110-pc-07,e-nl110-pc-08,e-nl110-pc
e-nl111;21;e-nl111-pc-01,e-nl111-pc-02,e-nl111-pc-03,e-nl111-pc-04,e-nl111-pc-05,e-nl111-pc-06,e-nl111-pc-07,e-nl111-pc-08,e-nl111-pc
e-nl112;21;e-nl112-pc-01,e-nl112-pc-02,e-nl112-pc-03,e-nl112-pc-04,e-nl112-pc-05,e-nl112-pc-06,e-nl112-pc-07,e-nl112-pc-08,e-nl112-pc
e-nl113;21;e-nl113-pc-01,e-nl113-pc-02,e-nl113-pc-03,e-nl113-pc-04,e-nl113-pc-05,e-nl113-pc-06,e-nl113-pc-07,e-nl113-pc-08,e-nl113-pc
e-nl114;41;e-nl114-pc-01,e-nl114-pc-02,e-nl114-pc-03,e-nl114-pc-04,e-nl114-pc-05,e-nl114-pc-06,e-nl114-pc-07,e-nl114-pc-08,e-nl114-pc
e-nl116;16;e-nl116-pc-01,e-nl116-pc-02,e-nl116-pc-03,e-nl116-pc-04,e-nl116-pc-05,e-nl116-pc-06,e-nl116-pc-07,e-nl116-pc-08,e-nl116-pc
e-nq100;21;e-nq100-pc-01,e-nq100-pc-02,e-nq100-pc-03,e-nq100-pc-04,e-nq100-pc-05,e-nq100-pc-06,e-nq100-pc-07,e-nq100-pc-08,e-nq100-pc
e-nq102;21;e-nq102-pc-01,e-nq102-pc-02,e-nq102-pc-03,e-nq102-pc-04,e-nq102-pc-05,e-nq102-pc-06,e-nq102-pc-07,e-nq102-pc-08,e-nq102-pc
e-nq103;21;e-nq103-pc-01,e-nq103-pc-02,e-nq103-pc-03,e-nq103-pc-04,e-nq103-pc-05,e-nq103-pc-06,e-nq103-pc-07,e-nq103-pc-08,e-nq103-pc
e-nq105;21;e-nq105-pc-01,e-nq105-pc-02,e-nq105-pc-03,e-nq105-pc-04,e-nq105-pc-05,e-nq105-pc-06,e-nq105-pc-07,e-nq105-pc-08,e-nq105-pc
e-nq106;21;e-nq106-pc-01,e-nq106-pc-02,e-nq106-pc-03,e-nq106-pc-04,e-nq106-pc-05,e-nq106-pc-06,e-nq106-pc-07,e-nq106-pc-08,e-nq106-pc
e-nlg102;21;e-nlg102-pc-01,e-nlg102-pc-02,e-nlg102-pc-03,e-nlg102-pc-04,e-nlg102-pc-05,e-nlg102-pc-06,e-nlg102-pc-07,e-nlg102-pc-08,e
e-nlg105;21;e-nlg105-pc-01,e-nlg105-pc-02,e-nlg105-pc-03,e-nlg105-pc-04,e-nlg105-pc-05,e-nlg105-pc-06,e-nlg105-pc-07,e-nlg105-pc-08,e
e-nlg106;21;e-nlg106-pc-01,e-nlg106-pc-02,e-nlg106-pc-03,e-nlg106-pc-04,e-nlg106-pc-05,e-nlg106-pc-06,e-nlg106-pc-07,e-nlg106-pc-08,e
e-nlg107;40;e-nlg107-pc-01,e-nlg107-pc-02,e-nlg107-pc-03,e-nlg107-pc-04,e-nlg107-pc-05,e-nlg107-pc-06,e-nlg107-pc-07,e-nlg107-pc-08,e-nlg107-pc-09,e
e-e201;21;h-e201-pc-01,h-e201-pc-02,h-e201-pc-03,h-e201-pc-04,h-e201-pc-05,h-e201-pc-06,h-e201-pc-07,h-e201-pc-08,h-e201-pc-09,h-e201
```

FIGURE B.1 – Réponse lors d'une requête de récupération de salle

Retour lors d'une récupération des machines

La réponse B.2 correspond à l'exécution de la requête A.2 sur le serveur contenant Nagios.

```
e-cg24-pc-01;10.7.65.10;e-cg24,all;0;No user logged in
e-cg24-pc-02;10.7.65.11;e-cg24,all;0;No user logged in
e-cg24-pc-03;10.7.65.12;e-cg24,all;0;No user logged in
e-cg24-pc-04;10.7.65.13;e-cg24,all;0;No user logged in
e-cg24-pc-05;10.7.65.14;e-cg24,all;0;No user logged in
e-cg24-pc-06;10.7.65.15;e-cg24,all;0;No user logged in
e-cg24-pc-07;10.7.65.16;e-cg24,all;0;No user logged in
e-cg24-pc-08;10.7.65.17;e-cg24,all;0;No user logged in
e-cg24-pc-09;10.7.65.18;e-cg24,all;0;No user logged in
e-cg24-pc-10;10.7.65.19;e-cg24,all;0;wl267337
e-cg24-pc-11;10.7.65.20;e-cg24,all;0;wl171410
e-cg24-pc-12;10.7.65.21;e-cg24,all;0;No user logged in
e-cg24-pc-13;10.7.65.22;e-cg24,all;0;No user logged in
e-cg24-pc-14;10.7.65.23;e-cg24,all;0;wl266525
e-cg24-pc-15;10.7.65.24;e-cg24,all;2;Cannot check: host is down
e-cg24-pc-16;10.7.65.25;e-cg24,all;0;No user logged in
e-cg24-pc-17;10.7.65.26;e-cg24,all;0;No user logged in
e-cg24-pc-18;10.7.65.27;e-cg24,all;0;No user logged in
e-cg24-pc-19;10.7.65.28;e-cg24,all;0;No user logged in
e-cg24-pc-20;10.7.65.29;e-cg24,all;0;No user logged in
e-cg24-pc-zz;10.7.65.30;e-cg24,all;0;No user logged in
e-clg41-pc-10;10.7.86.20;e-clg41,all;0;No user logged in
e-clg42-pc-zz;10.7.50.11;e-clg42,all;0;No user logged in
e-clg45-pc-zz;server;e-clg45,all;2;Cannot check: invalid hostname
e-clg46-pc-01;10.7.53.11;e-clg46,all;1;Cannot check: PC booted with linux
e-clg46-pc-02;10.7.53.12;e-clg46,all;1;Cannot check: PC booted with linux
e-clg46-pc-03;10.7.53.13;e-clg46,all;0;No user logged in
e-clg46-pc-04;10.7.53.14;e-clg46,all;1;Cannot check: PC booted with linux
e-clg46-pc-05;10.7.53.15;e-clg46,all;0;(null)
e-clg46-pc-06;10.7.53.16;e-clg46,all;0;No user logged in
e-clg46-pc-07;10.7.53.17;e-clg46,all;1;Cannot check: PC booted with linux
e-clg46-pc-08;10.7.53.18;e-clg46,all;1;Cannot check: PC booted with linux
e-clg46-pc-09;10.7.53.19;e-clg46,all;2;Cannot check: host is down
```

FIGURE B.2 – Réponse lors d'une requête de récupération des machines

Retour lors d'une récupération des utilisateurs

La réponse B.3 correspond à l'exécution de la requête A.3 sur le serveur contenant Nagios.

```
wl267337
wl171410
wl266525
wl274936
wl188929
wl217268
wl237318
wl238494
wl174255
wl173482
wl227070
wl238866
```

FIGURE B.3 – Réponse lors d'une requête de récupération des utilisateurs

C Base de données

Cette annexe donne une vue des relations entre tables ainsi que sur leur contenu. Dans un premier temps, une description rapide des tables sera faite. Ensuite, le diagramme de leur agencement les unes avec les autres sera donné en entier. Enfin, ce diagramme sera découpé en différentes parties pour plus de visibilité.

Description des tables

Yuukou II est composé de 14 tables :

- yuukou_rooms** contient les descriptions des salles informatiques ;
- yuukou_resources** contient les descriptions des ordinateurs appartenant à une salle particulière ;
- yuukou_users** contient les descriptions des utilisateurs ;
- yuukou_last** contient les historiques de toutes les connexions passées sur les ordinateurs ;
- yuukou_who** contient les historiques de toutes les connexions en cours sur les ordinateurs ;
- yuukou_mapping_location** contient les informations sur les différents campus de l'Université, le but étant de faire un lien avec la localisation de la table *yuukou_rooms* et la description complète de cette localisation contenue dans la table de *mapping* ;
- yuukou_mapping_room** contient les différentes correspondances entre le nom des salles telles qu'elles sont appelées dans *Yuukou II* et le nom des salles récupérées avec les emplois du temps ;
- yuukou_timetables** contient les différentes informations concernant un élément de l'emploi du temps ;
- yuukou_settings** contient les différentes informations de configuration de *Yuukou II* ;
- yuukou_groups** contient les descriptions des groupes de logiciels ;
- yuukou_software** contient les descriptions des logiciels ;
- yuukou_groups_software** contient les différents liens entre un groupe de logiciels et tous les logiciels les composant ;
- yuukou_rooms_groups** contient les différents liens entre les salles et tous les

groupes de logiciels les composant ;
yuukou_roms_software contient les différents liens entre les salles et tous les logiciels, n'appartenant pas à un groupe de logiciels, les composant.

Structure générale

La figure C.1 présente la structure générale du projet *Yuukou II*. Elle s'articule en différentes sous-parties : la partie archivage, logicielle, emploi du temps, mapping et configuration.

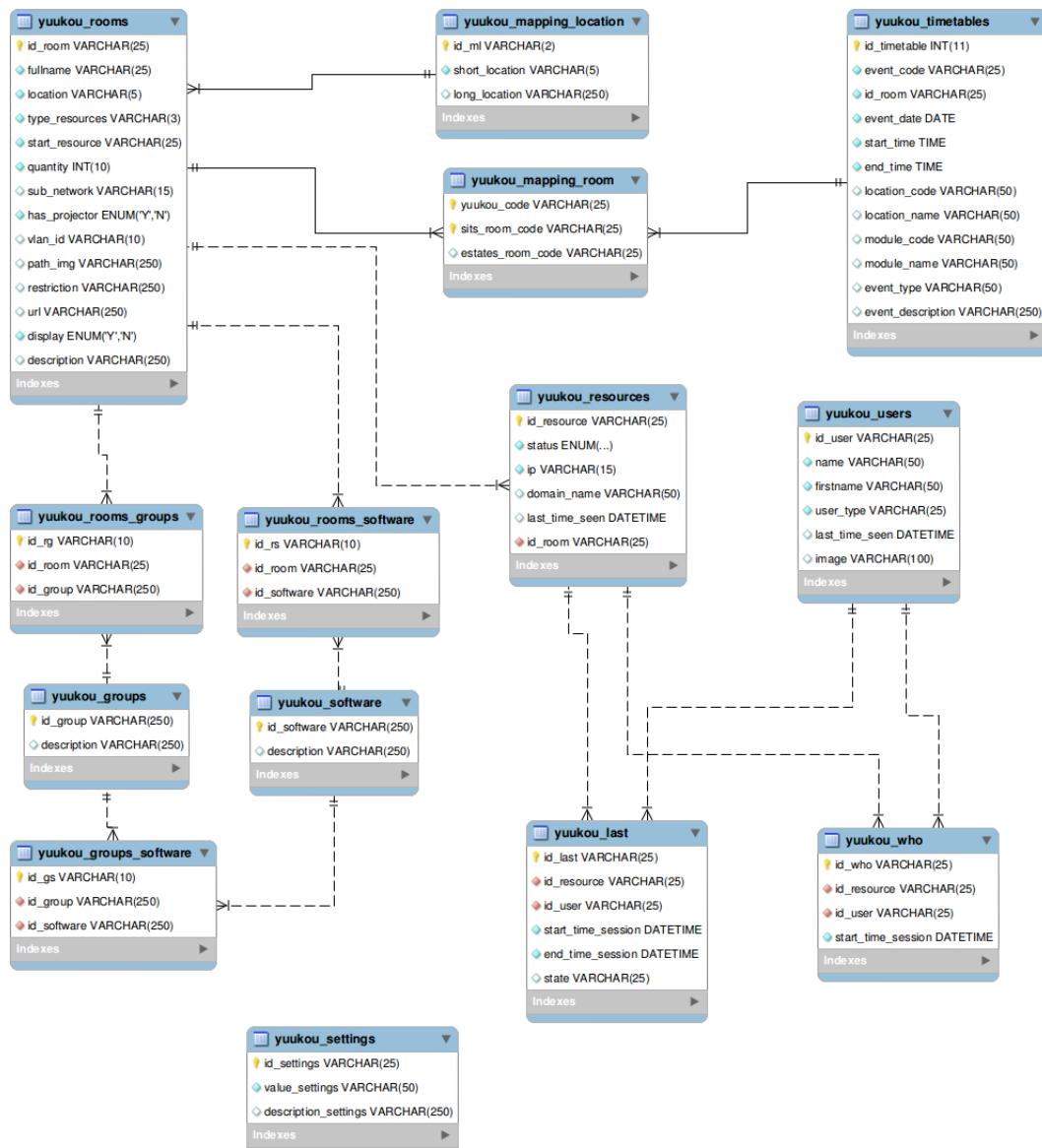


FIGURE C.1 – Structure générale de la base de données

Partie archivage

La partie archivage a pour rôle le stockage des données concernant d'une part l'historique des connexions actuelles et d'autre part, l'historique des connexions passées.

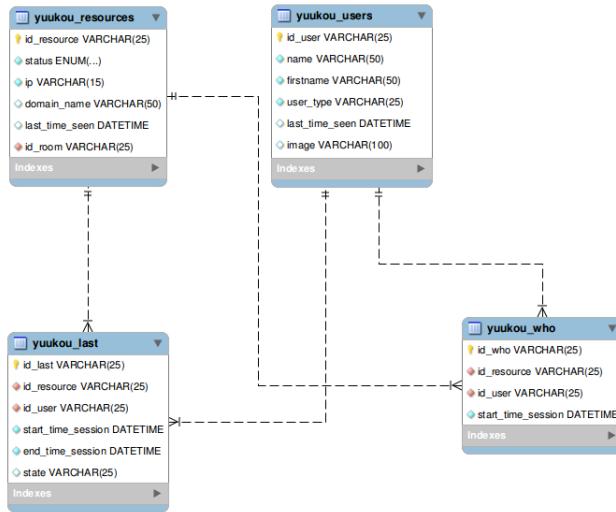


FIGURE C.2 – Structure de la partie archivage de la base de données

Partie logicielle

La partie logicielle a pour rôle de lier les informations de configuration logicielle avec les salles informatiques de l'Université.

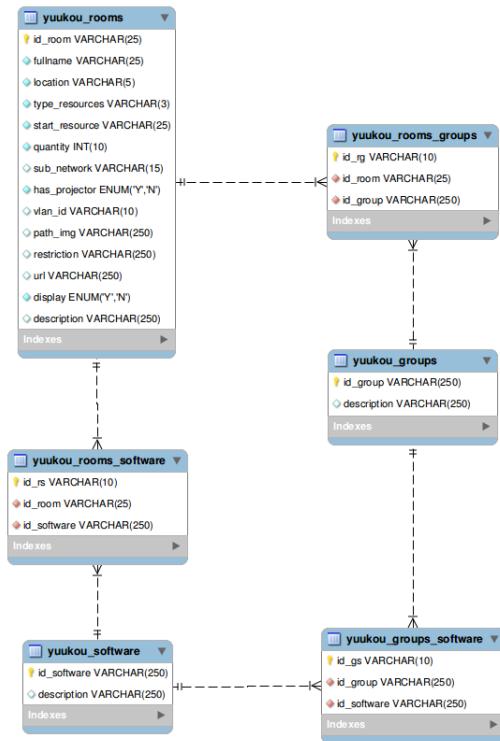


FIGURE C.3 – Structure de la partie logicielle de la base de données

Partie emploi du temps

La partie emploi du temps a pour rôle de lier les informations d'emploi du temps à une salle en passant par une table de mapping faisant le lien entre le nom d'une salle dans *Yuukou II* et le nom d'une salle utilisé par les services centraux informatiques lors de la génération des emplois du temps.

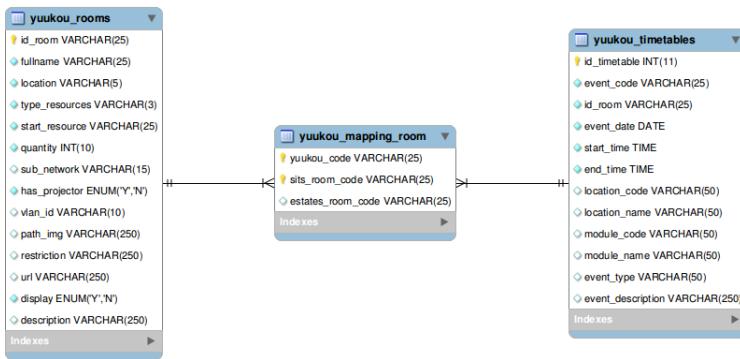


FIGURE C.4 – Structure de la partie emploi du temps de la base de données

Partie *mapping* avec les salles

La partie *mapping* avec les salles a pour rôle de faire le lien entre la localisation d'une salle et les informations concernant cette localisation.

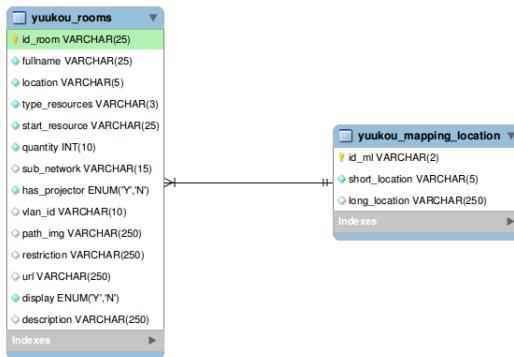


FIGURE C.5 – Structure de la partie *mapping* de la base de données

Partie configuration

La partie configuration a pour rôle de stocker tous les paramètres permettant de gérer un cycle du service Web pendant son exécution.

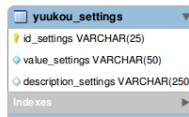


FIGURE C.6 – Structure de la partie configuration de la base de données

D Exemples de retours JSON

Cette annexe donne des exemples de retours corrects lors d'appels de fonctions du service Web. Deux retours de méthodes sont proposés ici.

Le code JSON D.1 correspond au retour de l'appel à la fonction **getSitesInformation ()** du service Web.

```
1 { "JSONMaintenance": "NO" ,  
2   "JSONLastCycle": "2012-05-28 17:59:36" ,  
3   "JSONContents": [ { "Location": "e" ,  
4     "LongLocation": "Electronics and  
5       Computer Science" ,  
6     "ShortLocation": "ECS"  
7   } ,  
8   { "Location": "h" ,  
9     "LongLocation": "Harrow" ,  
10    "ShortLocation": "HAR"  
11  } ,  
12  ...  
13  ] ,  
14  "JSONState": "OK"  
15 }
```

FIGURE D.1 – Exemple de retour de la fonction **getSitesInformation ()**

Le code JSON D.2 correspond au retour de l'appel à la fonction **getListRooms ()** du service Web.

```
1 { "JSONMaintenance" : "NO" ,  
2   "JSONLastCycle" : "2012-05-28 18:01:49" ,  
3   "JSONContents" : [ { "Room" : "e-cg24" } ,  
4                     { "Room" : "e-clg41" } ,  
5                     { "Room" : "e-clg42" } ,  
6                     { "Room" : "e-clg45" } ,  
7                     { "Room" : "e-clg46" } ,  
8                     { "Room" : "e-clg49" } ,  
9                     ...  
10                    ] ,  
11   "JSONState" : "OK"  
12 }
```

FIGURE D.2 – Exemple de retour de la fonction **getListRooms ()**

Table des figures

1.1	Blason de l'Université	3
1.2	Campus de Regent Street	4
1.3	Campus de New Cavendish Street	5
2.1	Architecture de <i>Yuukou</i>	9
2.2	Exemple de page publique de <i>Yuukou</i> représentant un campus et l'utilisation des salles informatiques d'un département	11
2.3	Exemple de page privée de <i>Yuukou</i> montrant en particulier les données d'un utilisateur	11
3.1	Fonctionnement des services Web	18
3.2	Exemple de classe Java annotée avec JAX-WS	20
4.1	Architecture du projet, partie service Web	22
4.2	Architecture du projet, partie affichage	22
4.3	Logo de Nagios	23
4.4	Exemple d'affichage de Nagios, vue sur tous les <i>hostgroups</i>	26
4.5	Exemple d'affichage de Nagios, vue sur tous les <i>hosts</i> d'un <i>hostgroup</i> spécifique	26
4.6	Logo de NetBeans	27
4.7	Logo de GlassFish	27
4.8	Logo de MySQL	28
4.9	Exemple de <i>mapping</i> d'une table avec sa classe DAO associée (<i>source : Cyrille Herby</i>)	29
4.10	Exemple de ligne pouvant être facilement <i>parsée</i>	31
4.11	Exemple de classe Java sérialisable	31

4.12	Exemple de document XML	32
4.13	Exemple de fichier JSON	33
4.14	Exemple de requête LQL	36
4.15	Schéma du déroulement du cycle principal du service Web	40
4.16	Structure générale de retour de fonction JSON sans erreur	43
4.17	Structure générale de retour de fonction JSON avec erreur	44
4.18	Exemple de graphe d'utilisation des salles informatiques de l'Université générée avec RRD4J	46
4.19	Schéma de la gestion du projet	48
4.20	Exemples de captures d'écrans sur un iPhone de l'application affichant les informations du service Web	51
5.1	Récapitulatif des fonctionnalités reprises de <i>Yuukou</i> et les nouvelles de <i>Yuukou II</i>	54
A.1	Code LQL de récupération des salles que surveille Nagios	64
A.2	Code LQL de récupération des machines que surveille Nagios	65
A.3	Code LQL de récupération des utilisateurs connectés sur les machines que surveille Nagios	65
B.1	Réponse lors d'une requête de récupération de salle	66
B.2	Réponse lors d'une requête de récupération des machines	67
B.3	Réponse lors d'une requête de récupération des utilisateurs	67
C.1	Structure générale de la base de données	70
C.2	Structure de la partie archivage de la base de données	71
C.3	Structure de la partie logicielle de la base de données	72
C.4	Structure de la partie emploi du temps de la base de données	73
C.5	Structure de la partie <i>mapping</i> de la base de données	73
C.6	Structure de la partie configuration de la base de données	73
D.1	Exemple de retour de la fonction getSitesInformation ()	74
D.2	Exemple de retour de la fonction getListRooms ()	75

Liste des tableaux

4.1 Comparatif des temps d'exécution entre la commande <code>netcat</code> et la com- mande <code>unixcat</code>	38
4.2 Abréviations et lieux utilisés par Nagios pour nommer les salles et ordi- nateurs	39

Résumé

Yuukou II est un projet ayant pour objectif d'afficher la disponibilité des salles informatiques de l'Université de Westminster à Londres. Il consiste en la création d'un service Web permettant de surveiller les ordinateurs de l'Université à l'aide de Nagios et d'archiver toutes les données utiles permettant de donner la disponibilité des salles. Le projet offre diverses fonctionnalités comme une connexion sécurisée, une récupération des emplois du temps, de la configuration logicielle des salles, des informations concernant les utilisateurs *via* LDAP ou encore la génération de graphes d'utilisation.

Le présent document rapporte le travail qui a été effectué dans le cadre du projet *Yuukou II* au sein de l'Université, projet réalisé pour le stage de fin de cursus de Master 2 à l'Université de Franche-Comté de Besançon.

Mots clés

Java, service Web, JAX-WS, Nagios, surveillance, LDAP, Yuukou, GlassFish, NetBeans, MySQL.

Abstract

Yuukou II is a project which enables to show the availability of computing laboratories within the University of Westminster in London. During the project, a Web service was implemented to monitor computers of the University with Nagios and which can archive all useful data for showing rooms' availability. The project provides various functionalities including secured connection, getting timetables of the University, software configuration of laboratories, information concerning users with LDAP or generation of graphical representation of laboratories utilization.

This document gives a view of the whole work done through the project *Yuukou II* within the University during the work placement of the second year of Master degree in the *Université de Franche-Comté* of Besançon.

Keywords

Java, Web service, JAX-WS, Nagios, monitoring, LDAP, Yuukou, GlassFish, NetBeans, MySQL.