|  |  |
| --- | --- |
|  | **2015** |
| logo_Paul_Sabatier.jpg | Université Paul Sabatier Toulouse 3 |

BARBASTE Rémi

BOULIC Guillaume

DEGIRONDE Robin

TOSI Emeric

|  |
| --- |
| **[Rapport de projet java]** |
| Rapport sur le déroulement du projet logiciel Java de l'année 2014-2015 de la L3 STRI |



Sommaire

[Introduction 3](#_Toc420342428)

[Présentation 4](#_Toc420342429)

[Cahier des charges 4](#_Toc420342430)

[Elaboration de la conduire du projet 5](#_Toc420342431)

[Outils choisis et méthodes de conduites du projet 5](#_Toc420342432)

[MVC 5](#_Toc420342433)

[Maven avec Eclipse 5](#_Toc420342434)

[PostgreSQL 6](#_Toc420342435)

# Introduction

Pour le déroulement de notre Licence 3 en Systèmes de Télécommunications et Réseaux Informatiques, nous avons eu a effectuer un projet logiciel Java afin de développer les concepts de Conception Orientée Objet, Base De Données et schématisation UML.

L'équipe fut composée de quatre membres: Rémi Barbaste, Guillaume Boulic, Robin Degironde et Emeric Tosi.

Le sujet modélise le problème d'une entreprise:

La société STRI veut mettre en place un service de gestion d'appareils connecté à un réseau .

La société possède des locaux et chaque local possède plusieurs salles.

La société veut pouvoir décrire les appareils selon plusieurs critères (nom, adresseMAC, emplacement géographique, système d'exploitation,, connexion entre appareils).

# Présentation

## Cahier des charges

La mission spécifiée sur la fiche de projet est d'organiser la répartition de matériels pour les besoin d'une entreprise, nous savons déjà qu'il y aura des données à manipuler.

Une contrainte est posée par le sujet: les données doivent être sauvegardées et importées vers une base de données.

Comme énoncé dans l'introduction le problème demandé est de stocker dans le logiciel des locaux avec leur propriété, Salles, Appareils pour une société nous avons donc comme entités

Local (nom et lieu)

Salle (nom, local parent et liste des appareils contenus)

Appareil (nom, modèle, système d''exploitation)

Chaque appareil doit être connecté sur un réseau il doit donc posséder une carte réseau avec les propriétés standards (adresseMac unique, numéro de Firmware).

Un firmware possède un nom et un numéro de version

Chaque appareil d'interconnexion étant lui même un appareil avec des propriétés qui lui sont propres, nous avons choisi de faire une nouvelle classe Switch héritant d'Appareil. mais avec la liste des appareil qu'il interconnecte en plus.

De même pour un Terminal (Ordinateur, tablette ...) il possède une propriété qui lui est propre: son type (ordinateur tablette) c'est donc une future nouvelle entité semblable à Appareil qui sera crée.

On aura donc en plus de tout cela les entités:

Interface Réseau (adresseMac, nom)

Système d'exploitation (nom, version)

Firmware(nom, version)

## Elaboration de la conduite du projet

Une fois les besoins analysés et le cahier des charges établi, nous avons pu faire un plan de développement nous guidant dans la marche à suivre afin de mener à bien notre projet.

La première partie du travail a consisté à faire une étude de la conception du logiciel, notamment l'utilisation de concepts de la Conception Orientée Objet (Modèle).

La deuxième partie fut la conception de la base de donnéés et d'un jeu de données afin de pouvoir faire des tests unitaires.

La troisième partie, la plus conséquente fut la conception des fenêtres voulues sur papier.

La dernière partie, la plus conséquente fut le codage du logiciel.

Parallèlement à cela, chaque fonction de chaque module étant impliqué dans le fonctionnement du logiciel (donc en particulier le modèle) ont subit une batterie de tests unitaires afin de fiabiliser et optimiser le développement. Cela a pour conséquence de minimiser les phases de débugage.

## Outils choisis et méthodes de conduites du projet

Maitre mot: **Travail Collaboratif**

### MVC

Nous avons choisi de construire l'architecture du logiciel selon le modèle MVC: Modèle Vue Controleur

Le modèle: Module qui s'occupe de faire le traitement des données qui transitent dans le logiciel.

La vue: Module qui s'occupe de mettre en forme toutes les fenêtres et la partie IHM (Interface Homme Machine) du logiciel

Le contrôleur: Module qui s'occupe de détecter toutes les interractions avec l'utilisateur et de lancer dans le Modèle le traitement associé.

### Maven avec Eclipse

Le choix d'une organisation Maven sous Eclipse s'est montré évident puisque demandé dans le sujet.

Eclipse est un SDK (Software Developpment Kit) spécialisé dans le développement Java.

Maven est un module que l'on peut utiliser sous Eclipse pour gérer les dépendances entrainées par le travail collaborratif.

En effet, quand plusieurs développeurs travaillent sur des plateformes différentes, il se peut qu'il y ai des conflits de compilation selon les architectures utilisées. Maven s'occupe de gérer ces dépendances, rendant le projet adaptables sur un grand nombre d'architectures.

### PostgreSQL

Puisque demandé dans le sujet, nous avons choisi d'utiliser une base de données poru sauvegarder les données du programme. Nous avions le choix entre deux SGBD Open source: MySQL et PostgreSQL. Nous avons choisi PostgreSQL qui se montre plus compétent quant aux gestions des contraintes type clé étrangères (FOREIGN\_KEY)

# Conception du produit

## Répartition des taches

Le projet a été uniformément Réparti de façon à profiter au mieux des acquis de chacun afin d'avancer vite et d'approfondir les connaissances déja établies:

Remi Barbaste: Jeu de données, Javadoc

Guillaume Boulic : Jeu de données

Degironde Robin: Conception UML& BDD, Codage Junit , Modèle , Interaction BDD.

Emeric Tosi: Conception et Codage Vues et Controleurs

## Java

En prenant compte des contraintes et attentes du client / sujet: nous en sommes arrivé au schéma comme suit

### class_diagram.bmpSchéma uml

Comme on peut le constater sur le schéma, le modèle se composera de sept classes:

* Local
* Salle
* Appareil
* Terminal
* Switch
* Os
* InterfaceRéseau
* Firmware

Plus une classe enumType qui se charge de mettre en constante le type d'un appareil.

### Classe Local

|  |  |  |
| --- | --- | --- |
| idLocal | Entier Unique | Unique |
| nomLocal | String |  |
| lieuLocal | String |  |
| salles | ArrayList<Salle> | Représente les salles dépendant d'un local. |

### Classe Salle

|  |  |  |
| --- | --- | --- |
| idSalle | Entier Unique | Unique |
| nomSalle | idUnique |  |
| listeAppareils | ArrayList<Appareil> | Représente les appareils appartenants à une salle |

### Classe Appareil

|  |  |  |
| --- | --- | --- |
| idAppareil | Entier | Unique |
| nomAppareil | String | Nom d'un appareil |
| marqueAppareil | String |  |
| modèleAppareil | String |  |
| etatAppareil | Boolean | Appareil activé ou non |
| carteReseau | InterfaceReseau | Carte Réseau d'un appareil |
| Os | SystemeExploitation | OS d'un Appareil |

### Classe SystemeExploitation

|  |  |  |
| --- | --- | --- |
| idOS | Entier | Unique |
| nomOs | String |  |
| versionOs | String | Version de l'os |

### Classe Firmware

|  |  |  |
| --- | --- | --- |
| idFirmware | Entier | Unique |
| nomFirmware | String |  |
| versionFirmware | String | Version du firmware |

### Classe InterfaceReseau

|  |  |  |
| --- | --- | --- |
| adresseMAC | Entier | Unique, adresse MAC unique de l'interface reseau |
| nomInterface | String | Nom de l'interface |
| firmwareInterface | Firmware | Firmware de l'interface réseau |

Comme nous nous pouvons le voir dans les classes, il y a réutilisation des classes crées (Firmware, InterfaceReseau, SystèmeExploitation).

Le schéma UML montre également que les Switch et Terminal héritent de la classe appareil, ils sont donc des appareils, héritant de toutes es méthodes et attributs mais ayant leurs spécificités:

On aborde ici les concepts d'Héritage et de Polymorphisme.

### Classe Terminal

|  |  |  |
| --- | --- | --- |
| typeTerminal | Enum | ORDINATEUR ou TABLETTE |

### Classe Switch

|  |  |  |
| --- | --- | --- |
| listeAppareils | ArrayList<Appareil> | Liste des appareils interconnectés par le switch en question. |

## Base de données

Étant donné que:

Les salles n'ont qu'un seul et unique local parent

Les appareils n'ont qu'une seule et unique salle parente

Les appareils n'ont qu'une seul est unique os

Les appareils n'ont qu'une seule et unique carte réseau

Une carte réseau ne peut être connectée qu'à un seul et unique Switch

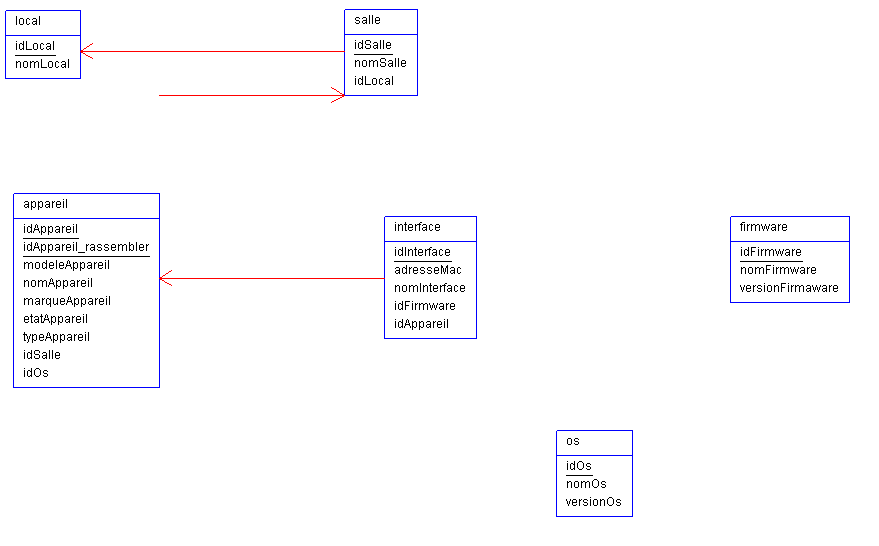
Nous avons choisi de modéliser la basede données comme suit, ainsi, nous pouvons sauvegarder la totalité des objets et leus attributs à l'identique.

Seuls les Appareils et Classes filles sont fondues dans une seule table Appareil.

### MCD.pngModèle Conceptuel de Données

Les relations ci-dessus entrainent la création des tables brutes suivantes, issues du MLDR:

### Modèle Logique de DonnéesRelationnel



On peut Voir ici que nous avons six tables finales dans lesquelles nous allons pouvoir sauvegarder le Modèle (Nous parlons ici du module Modèle utilisé par le scéma MVC Java);

* Local
* Salle
* Appareil
* Interface
* Firmware
* Os