2015

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

Table des illustrations

Introduction

Au cours de notre Licence 3 STRI (*Systèmes de Télécommunications et Réseaux Informatiques*), nous avons eu à 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.

Afin de gérer le travail collaboratif, l'utilisation de la plateforme qui s'est avéré une nécessité. Le dépôt ce trouve à l'adresse suivante : https://github.com/emeric254/Java-STRI-S2/tree/master/projet.

Le sujet modélise le problème d'une entreprise : la société STRI.

Elle veut mettre en place un service de gestion d'appareils connectés à au 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 alors qu'il y aura des données à manipuler.

Une contrainte est posée par le sujet : les données doivent être sauvegardées dans une base de données et également importées de celle-ci.

Comme énoncé dans l'introduction, le problème demandé est de stocker dans le logiciel des locaux avec leurs propriétés, les salles et les appareils que possède une société. Pour cela 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

Un appareil d'interconnexion, possédant lui-même des propriétés qui lui sont propres, nous avons choisi de faire une nouvelle classe Switch héritant d'Appareil. Mais cette classe contient, en plus des informations communes avec les autres appareils, la liste des équipements qu'il interconnecte.

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

On aura donc, en plus de toutes celles précédemment indiquées, 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 données et d'un jeu de données afin de pouvoir faire des tests unitaires.

La troisième parti 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 (en particulier le modèle) ont subi une batterie de tests unitaires afin de fiabiliser et optimiser le développement. Cela a pour conséquence de minimiser les phases de débogage.

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 Contrôleur.

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 gérer toutes les interactions entre l'utilisateur et le programme, en lançant dans le Modèle le traitement associé.

Maven avec Eclipse

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

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

Maven est, quand à lui, un module que l'on peut utiliser sous Eclipse pour gérer les dépendances entrainées par le travail collaboratif.

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 pour sauvegarder les données du programme. Nous avions le choix entre deux SGBD (*Système de gestion de base de données*) Open source : MySQL et PostgreSQL. Nous avons choisi PostgreSQL qui se montre plus compétent d'un point de vue de la gestion des contraintes de type clé étrangères (FOREIGN_KEY).

Conception du produit

Répartition des tâches

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 déjà établies :

- **Rémi Barbaste :** Jeu de données, Javadoc.
- **Guillaume Boulic :** Jeu de données.
- **Robin Degironde :** Conception UML & BDD, Codage Junit, Modèle, Interaction BDD.
- **Emeric Tosi :** Conception et Codage Vues et Contrôleurs.

Java

En prenant compte des contraintes et attentes du client / sujet, nous en sommes arrivé au schéma UML présenté ci-après.

Schéma uml

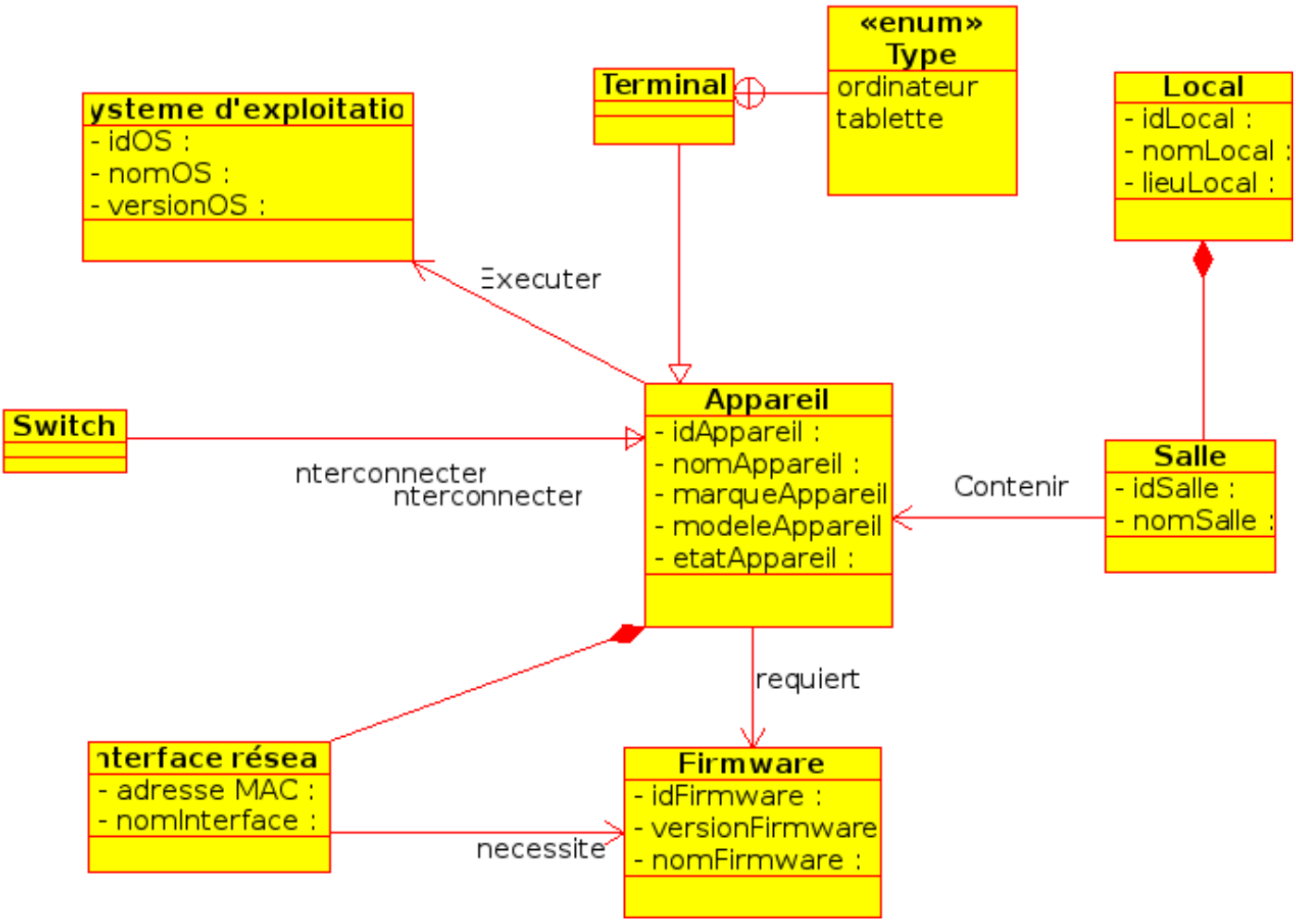

Figure 1 : Schéma UML

Comme on peut le constater sur ce 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 pouvons le voir dans ces classes, il y a une réutilisation des classes créées (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 les 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 les contraintes suivantes :

- 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'un 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 base de données comme suit. Ainsi, nous pouvons sauvegarder la totalité des objets et leurs attributs à l'identique.

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

Modèle Conceptuel de Données

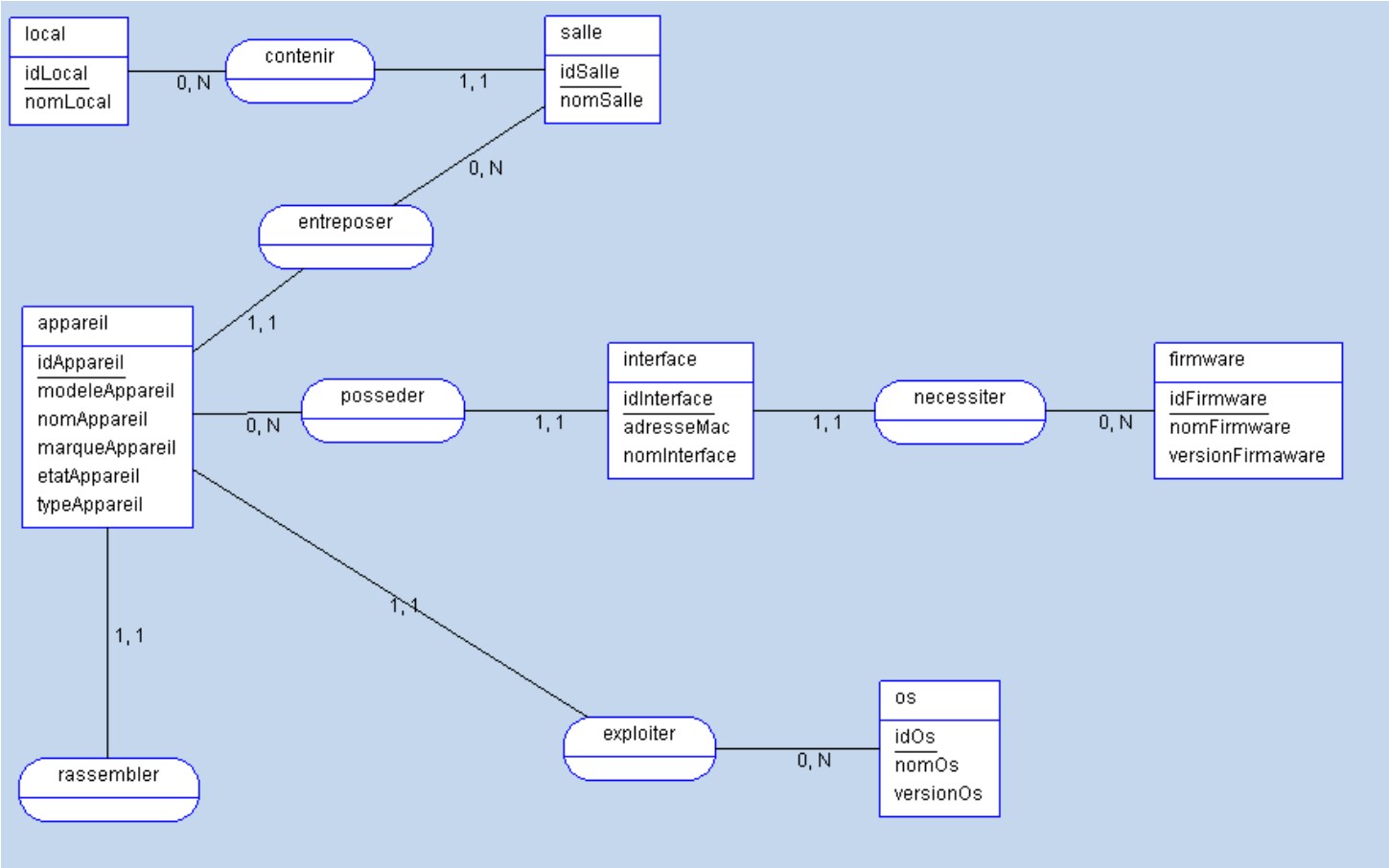

Figure 2 : Modèle Conceptuel de Données

Les relations ci-dessus entrainent la création des tables brutes suivantes, issues du MLDR (visible dans la partie qui suit).

Modèle Logique de Données Relationnel

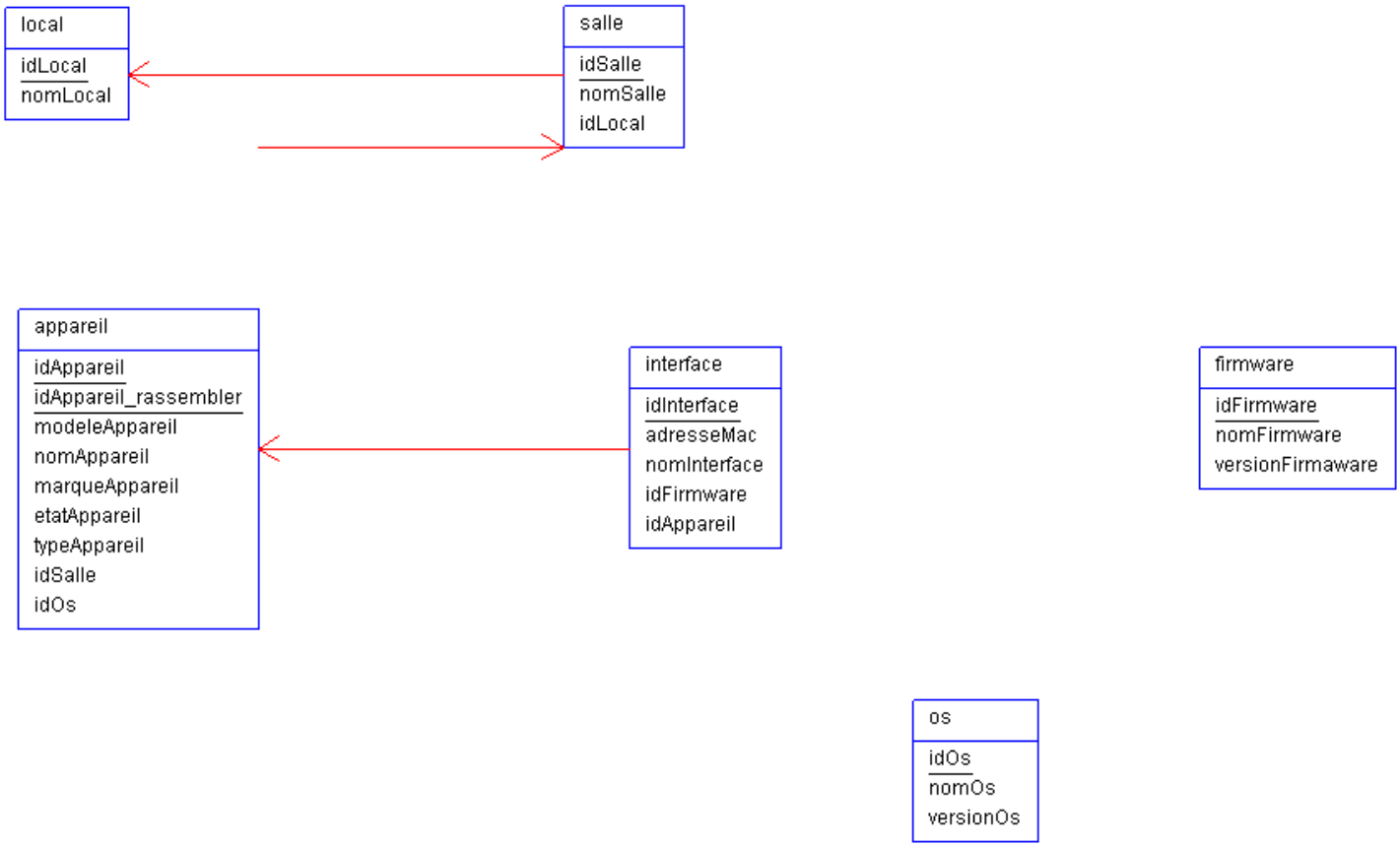

Figure 3 : Modèle Logique de Données Relatonnel

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 schéma MVC Java).

Ces tables sont les suivantes :

- Local
- Salle
- Appareil
- Interface
- Firmware
- Os

Manuel d'utilisation

Lancement du programme

Lors du lancement de l'application de gestion du parc informatique, la page suivante apparaît :

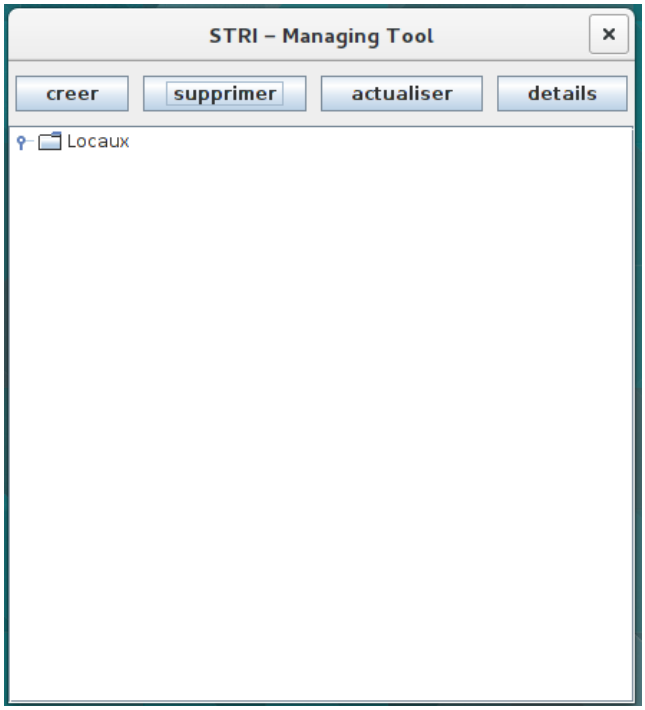

Figure 4 : Lancement du programme

Cette page permet de gérer les équipements, on peut grâce à elle ajouter ou supprimer des locaux, des salles et des équipements. De plus, on peut également actualiser l'affichage ou afficher des détails sur l'élément sélectionné.

Création d'un local

Pour créer un local, il suffit de cliquer sur « creer » puis de saisir le nom et le lieu de ce nouveau local. Après validation des informations, la page d'accueil s'affiche avec le local ajouté.

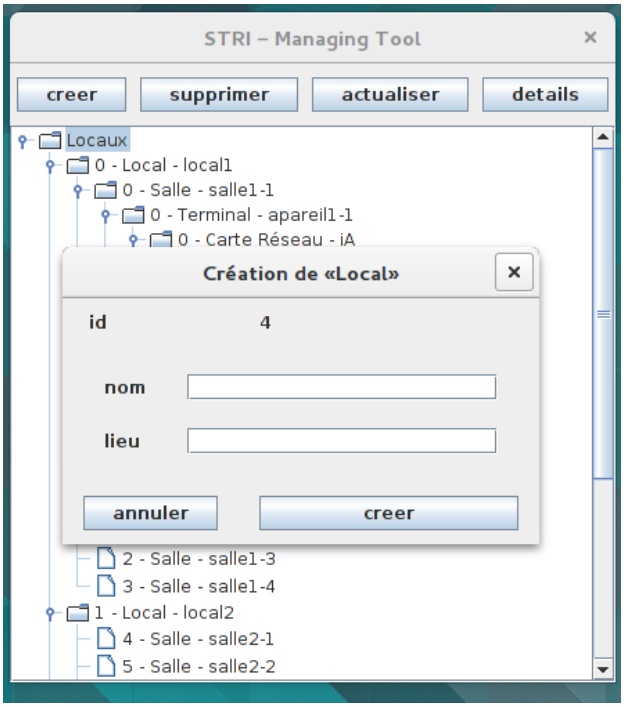

Figure 5 : Création d'un local

Il est possible d'en créer le nombre souhaité en réitérant cette manipulation autant de fois que nécessaire.

Création d'une salle

Dans un local, il est possible de créer des salles. Pour cela, il faut sélectionner le local dans lequel on souhaite la rajouter puis cliquer sur « creer ». Une nouvelle fenêtre apparaît demandant de saisir le nom de la salle.

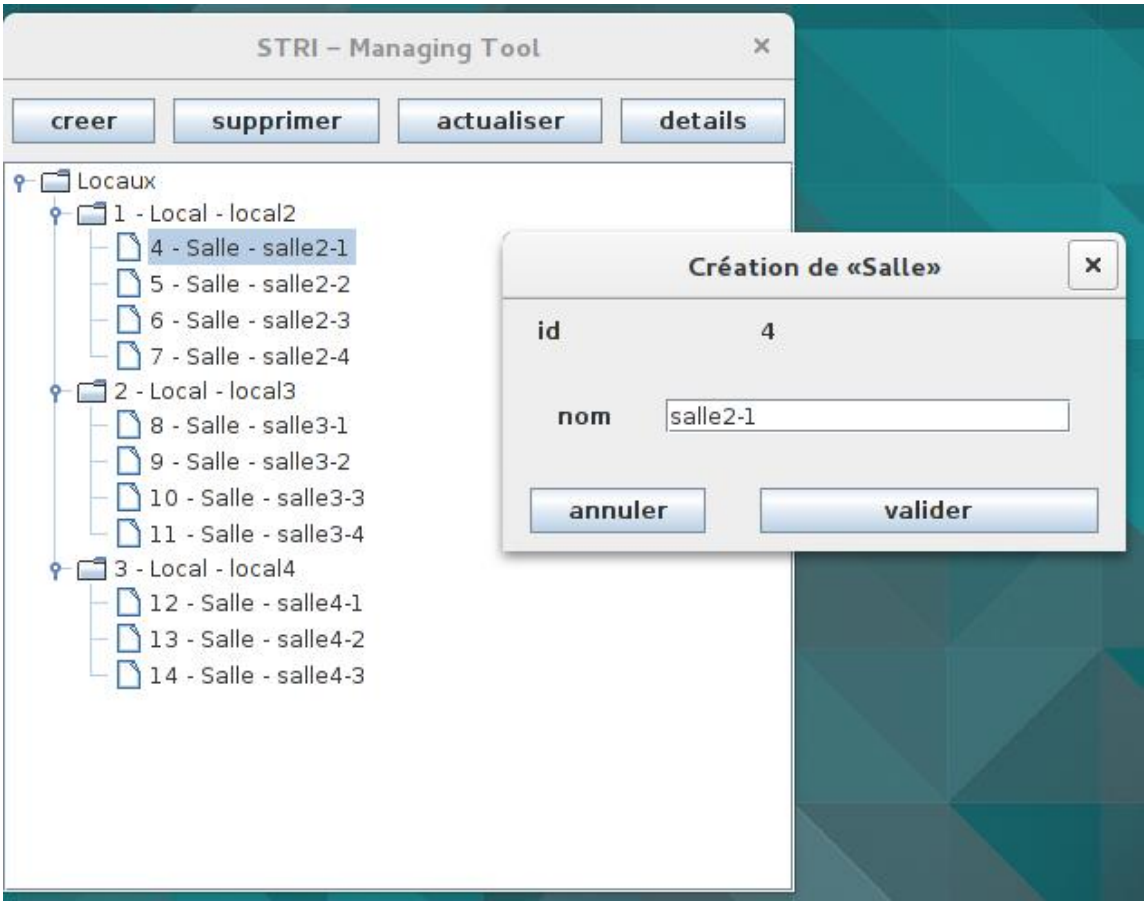

Figure 6 : Création d'une salle

Création d'un équipement

Maintenant qu'il y a au moins un local et une salle, il possible des créer des appareils. En effet un appareil devant être placé dans une salle, il faut cliquer sur celle où on souhaite le placer puis sur « créer ». Il suffit alors de remplir les champs demandés, à savoir le nom, la marque, le modèle. Ensuite on choisit si cet équipement est ou non actif en cochant la case associée et on sélectionne dans le menu déroulant le type d'équipement et l'OS. Tant que tous les champs ne sont pas remplis, l'appuie sur le bouton « creer » ne fonctionne pas.

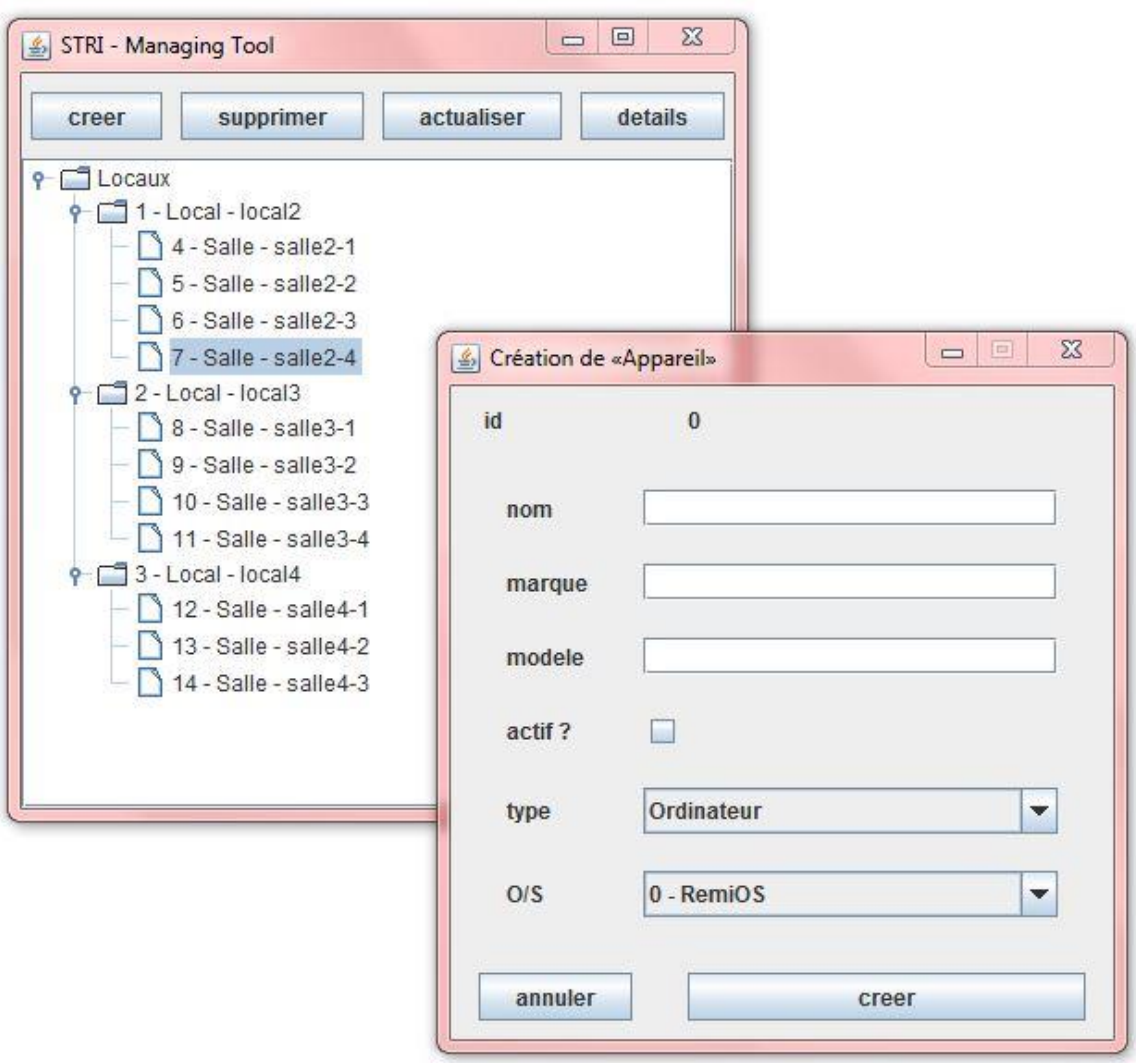

Figure 7 : Création d'un équipement

Annulation d'une création

En cas d'erreur ou de retrait d'un élément, il suffit de le sélectionner et de cliquer sur le bouton « supprimer ». Dans le cas d'une annulation au moment de la création, le fait de cliquer sur « annuler » permet de fermer la fenêtre sans rajouter l'élément. Un message de confirmation apparait alors pour vérifier que l'utilisateur souhaite réellement effectuer l'annulation.

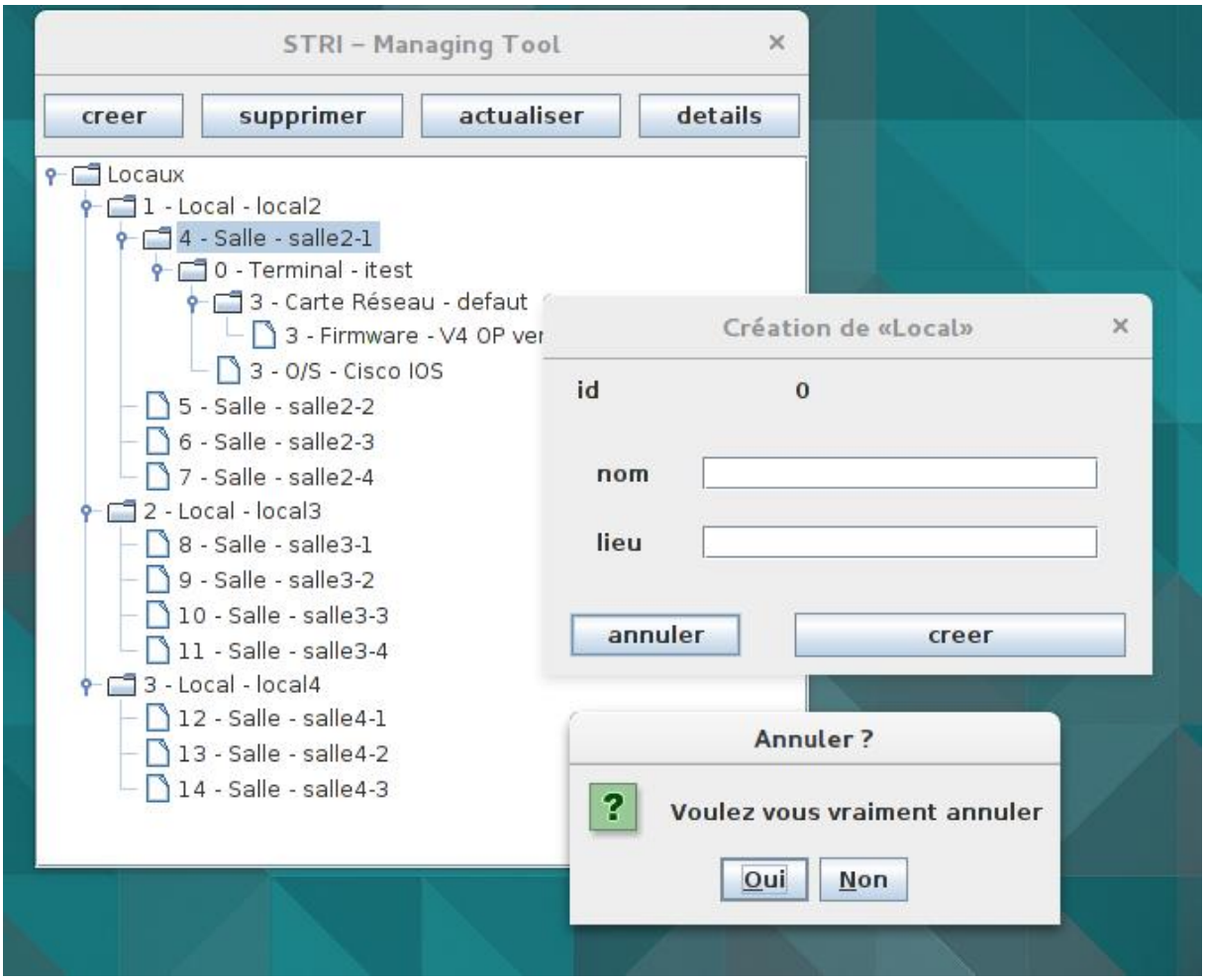

Figure 8 : Annulation de la création d'un local

Gestion de la interconnexion des appareils

Lorsqu'il y a dans une salle un switch, il est possible de définir quels sont les appareils qui lui sont connectés. Pour cela il suffit, à l'intérieur des informations du switch, de cliquer sur « connexions » puis « creer ». On peut alors sélectionner dans le choix déroulant l'appareil à interconnecter.

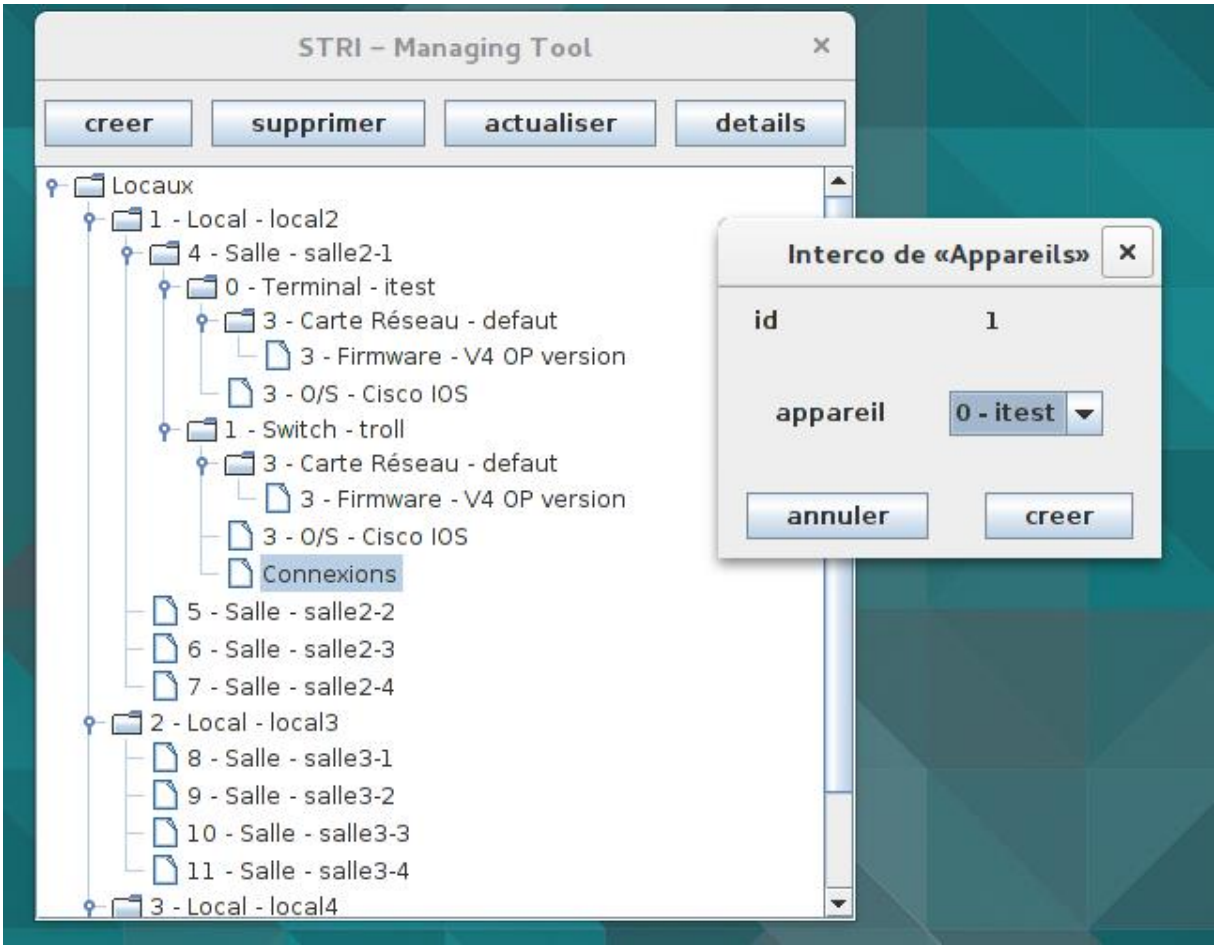

Figure 9 : Ajout d'un appareil connecté au switch

Modification des informations d'un élément

A tout moment, il est possible de modifier les informations d'un élément. Il suffit simplement de sélectionner l'élément et de cliquer sur « details ». Après modification des informations souhaitées, et confirmation de la modification via le bouton « valider », les informations sont mises à jour dans le programme ainsi que dans la base de données.

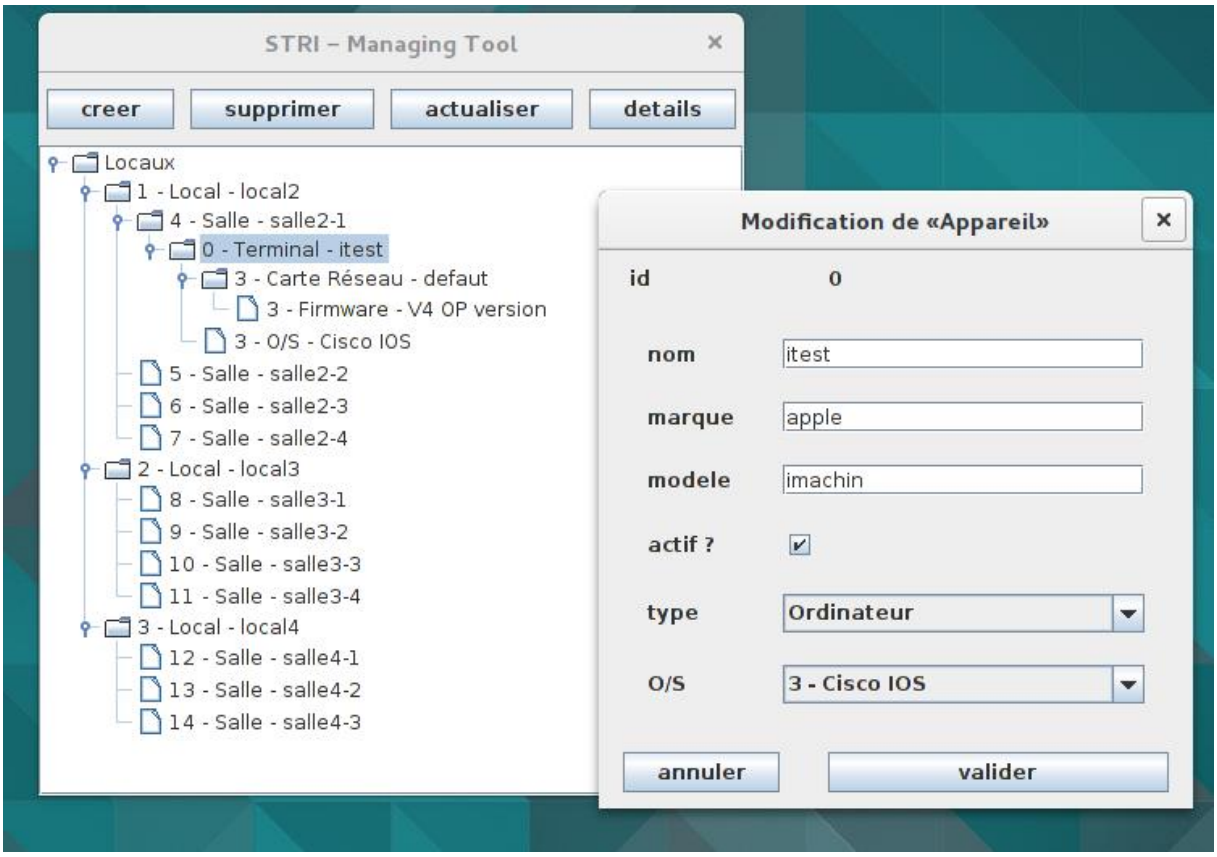

Figure 10 : Affichage du détail d'un appareil