UNIVERSITÉ DE MONTPELLIER

I.U.T MONTPELLIER-SETE

RAPPORT FINAL DE PROJET

Éditeur de Configuration de Réseaux

RÉALISÉ PAR

DERBALI Karim
FARAH-SHAKIR Ismael
Lloret Adrien
NICOLAS Thibault

SOUS LA DIRECTION DE

M. Garcia Francis

POUR L'OBTENTION DU DUT INFORMATIQUE
ANNÉE UNIVERSITAIRE 2020-2021

Remerciements

Nous tenons à remercier dans un premier temps toute l'équipe pédagogique de l'IUT ainsi que les intervenants professionnels.

Avant d'entamer ce rapport, nous souhaitons également en profiter pour remercier notre tuteur, M. Garcia Francis, ainsi que Mme. Messaoui Anita, pour leur générosité en matière de formation et d'encadrement et qui n'ont pas cessé de nous encourager pendant notre projet. Nous les remercions également pour l'aide et les conseils concernant les missions qui seront évoquées durant ce rapport, ainsi que la confiance que notre tuteur de projet nous a témoignée. Nous tenons aussi à remercier nos professeurs de nous avoir incités à travailler pragmatiquement sur ce projet en mettant à notre disposition leur expérience et leurs compétences.

Glossaire

- *Framework :* En programmation informatique, un framework désigne un ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel (Ex: JAVA FX).

- *Instance :* En programmation orientée objet, on appelle instance d'une classe un objet avec un comportement et un état, tous deux définis par la classe.

- *FXML :* Format de données textuelles, dérivé du format XML, qui permet de décrire une interface utilisateur pour des applications conçues avec JAVA FX.

- *Drag-And-Drop (Glisser et Déposer) :* Sert à déplacer un élément dans une application en restant appuyé sur ce dernier avec le clic de la souris.

- *IDE (Integrated Development Environment) :* Regroupe un ensemble d'outils spécifiques. Ceux-ci sont dédiés aux programmeurs afin qu'ils puissent optimiser leur temps de travail et améliorer leur productivité.

- *Adresse IP (Internet Protocol) :* est le numéro qui identifie chaque ordinateur connecté à Internet, plus précisément, l'interface avec le réseau de tout matériel informatique (routeur, imprimante) connecté à un réseau informatique utilisant l'Internet Protocol

- *Internet Protocol :* Ensemble de protocoles de communication utilisés sur l'Internet.

- *Pattern:* Un patron de conception est un arrangement caractéristique de modules, reconnu comme bonne pratique en réponse à un problème de conception d'un logiciel. Il décrit une solution standard, utilisable dans la conception de différents logiciels.

- *Pattern Composite :* C'est un pattern de conception qui permet de composer des objets en arborescence avec une hiérarchie composant/composé.

Sommaire

Introduction

Le département Informatique de l'IUT de Montpellier possède un réseau interne (Intranet) permettant une communication entre toutes les machines, que ce soient les ordinateurs ou bien les imprimantes. Au fil des années, cet intranet s'est développé et a accueilli de nouvelles machines et de nouveaux appareils de communication, comme des switchs, des répéteurs ou des routeurs. Il permet aujourd'hui la synchronisation et le stockage, dans un serveur central, de l'ensemble des données concernant les utilisateurs et de gérer les divers échanges faits entre eux et sur Internet.

Malheureusement, dans le cas où nous voudrions modifier notre réseau en ajoutant ou supprimant des composants, il sera difficile d'avoir une vision d'ensemble des éléments qui le constitue.

C'est donc dans cette optique qu'il nous a été demandé de réaliser un logiciel permettant de construire une modélisation de l'intranet du département informatique de l'IUT de Montpellier-Sète afin de pouvoir avoir une meilleure visualisation de l'ensemble du réseau. Bien évidemment, l'utilisation de ce programme sera extensible à tout type de réseau local.

Premièrement, nous établirons l'analyse de l'existant ainsi que l'analyse des besoins fonctionnels et non-fonctionnels du logiciel que nous avons créé. Deuxièmement, nous proposerons un rapport technique incluant l'évolution de l'intégralité du processus de développement. Seront joints à ce rapport les résultats obtenus par le logiciel, comprenant notamment les tests unitaires, ainsi que les manuels d'installation et d'utilisation. Finalement, nous verrons comment nous nous sommes organisés lors de ce projet au travers de notre rapport d'activité.

Analyse

Analyse du sujet et de son contexte

Analyse de l'existant

Il existe un grand nombre de logiciels payants ou gratuits pour créer des diagrammes de réseau interactifs (EdrawMax, VisualParadigm, LucidChart, Network Notepad,...). Cependant, nous nous sommes inspirés du logiciel de conceptualisation StarUML pour la réalisation de notre éditeur de réseau car c'est un logiciel que nous avons l'habitude d'utiliser en cours et qui répond à la majorité des besoins pour notre projet, comme par exemple au niveau de la sauvegarde, de l'importation, de l'exportation dans d'autres formats, ou plus simplement son système de Drag-And-Drop et de liaisons entre les composants. Notre logiciel sera disponible dans un format exécutable, donc utilisable directement en cliquant sur un icône et non pas en étant obligé de devoir utiliser un IDE pour pouvoir le lancer. Ainsi, toutes les sauvegardes et importations se feront uniquement sur la machine de l'utilisateur, et il n'y aura donc pas besoin de base de données pour son bon fonctionnement.

Analyse de l'environnement

Dans le cadre de la demande faite par notre client de concevoir un logiciel de conceptualisation du réseau interne du département informatique de l'IUT de Montpellier-Sète, nous devrons réaliser un programme capable de représenter tous les types de composants d'un réseau et ses différentes liaisons afin d'avoir une vision globale du réseau actuel.
Il ne nous a pas été explicitement demandé de réaliser ce logiciel dans le but qu'il soit adaptable à n'importe quel réseau, comme par exemple celui d'une entreprise. Cependant, il n'en reste pas moins que les différents composants au sein d'un réseau informatique restent fondamentalement les mêmes et qu'il serait possible d'utiliser notre programme dans d'autres contextes que celui décrit par notre client.
Ce logiciel sera donc, dans un premier temps, utilisé exclusivement par l'équipe technique de l'IUT, mais aura aussi pour vocation d'être utilisable dans quelques années par les étudiants de première année de l'IUT afin de les familiariser aux principes de base du réseau.

Analyse des besoins fonctionnels

Spécifications fonctionnelles

L'éditeur de configurations réseau permettra à son utilisateur la conceptualisation de son réseau pour avoir une vue globale de ce dernier.

L'environnement de travail sera divisé en 3 parties que l'on va détailler par la suite et qui est représenté ci-dessous (Figure 3). Ces choix se sont principalement axés sur les diagrammes de cas d'utilisations qui permettent de répondre aux besoins fonctionnels de l'utilisateur (Figure 1 & 2).

Premièrement, dans la partie haute de l'application, nous aurons les différentes fonctionnalités concernant les outils classiques d'un logiciel quelconque, tels que l'exportation de fichiers en format PDF, la sauvegarde du projet en cours ou le chargement d'un projet réalisé auparavant.

Deuxièmement, nous aurons une partie "sélection" qui aura pour but de choisir les composants que l'on souhaite ajouter à notre réseau actuel. Elle sera constituée d'une liste de tous les composants que l'on peut retrouver dans un réseau, comme des routeurs, des switchs, ou des répéteurs par exemple. Cette partie utilisera un système de "glisser/déposer", que nous appellerons "Drag-And-Drop" tout au long de ce rapport, qui sera utilisé entre la partie "sélection" et le "plan de travail" (que l'on verra dans la troisième partie) pour ajouter les différents éléments.

Et finalement, nous aurons la partie "plan de travail" qui permettra de visualiser l'architecture du réseau en cours de construction par l'utilisateur. Le technicien pourra supprimer et modifier les composants qu'il souhaite.

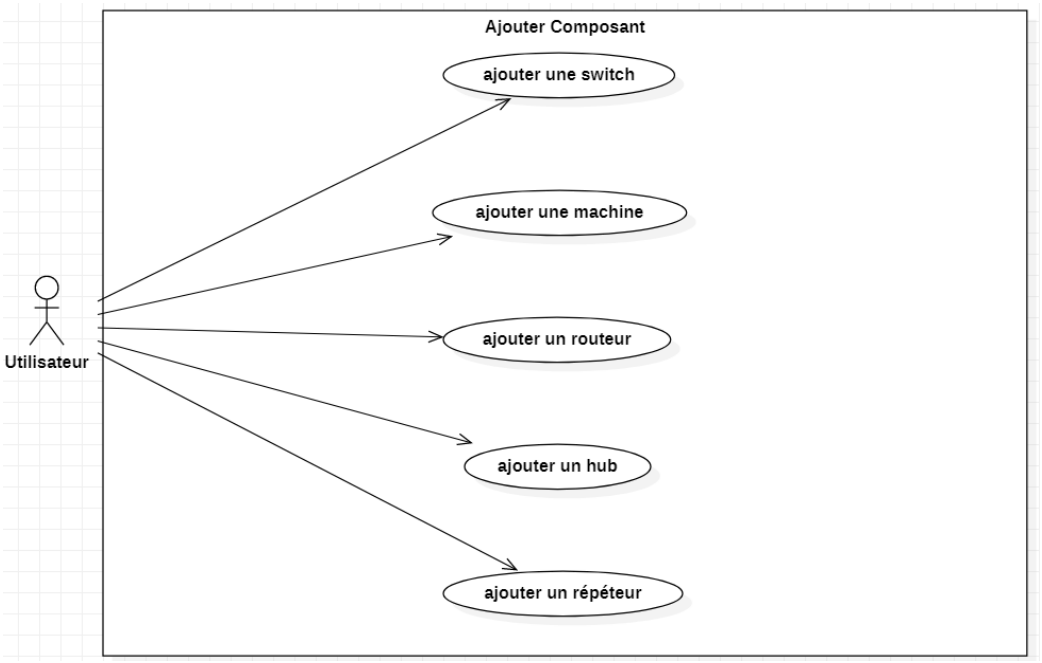

Figure 1 : Diagramme général des cas d'utilisation

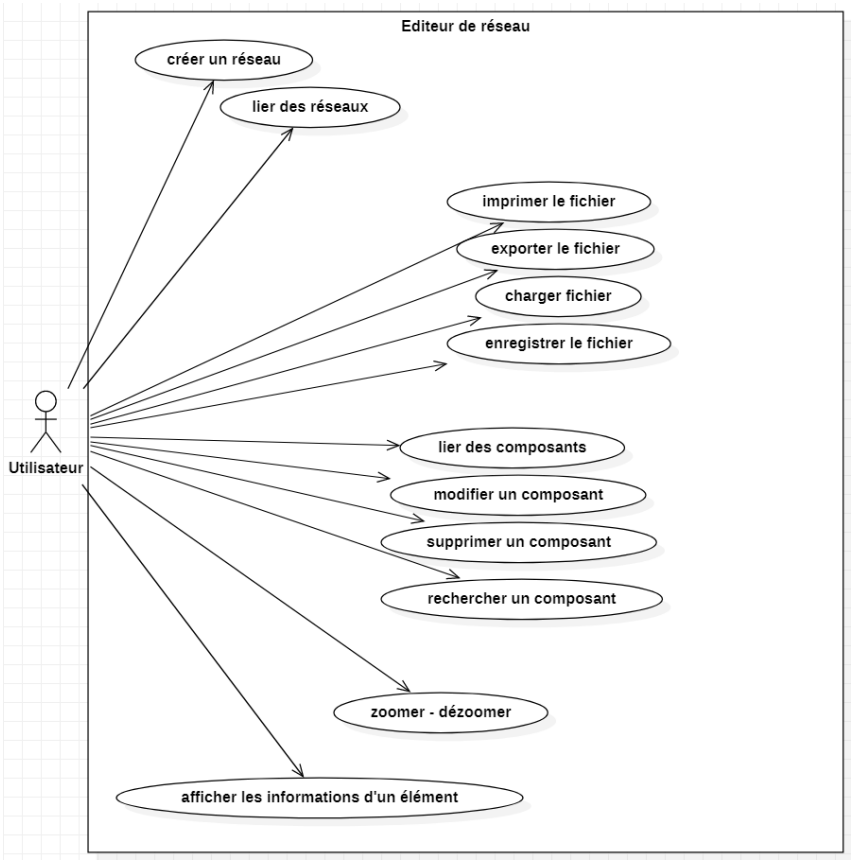

Figure 2 : Diagramme détaillé des cas d'utilisation

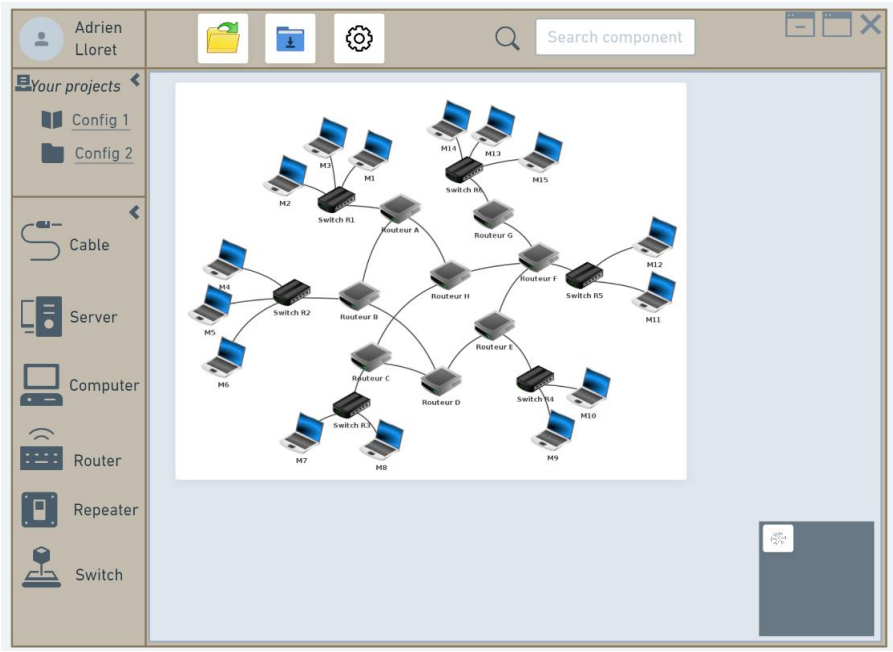

Figure 3 : Maquette lors de la conception du logiciel

Analyse des besoins non-fonctionnels

Spécifications techniques

Aucune contrainte spécifique ne nous a été imposée, il nous a simplement été demandé de choisir nous-mêmes une solution adaptée au problème. Nous souhaitons pouvoir modéliser des réseaux et sauvegarder la modélisation afin de pouvoir y accéder ou de la modifier. Il a donc fallu choisir comment nous allions stocker les données relatives à la sauvegarde d'un projet. Une base de données n'a aucune utilité dans notre projet car ces données peuvent être directement sauvegardées sur la machine personnelle de l'utilisateur sous forme de fichier local. Ainsi, l'utilisateur n'aura plus qu'à renseigner l'emplacement de la sauvegarde pour pouvoir rouvrir son projet.

Nous avons choisi Java comme langage de programmation car c'est le langage de programmation orienté objet le plus utilisé et parce que nous l'avons déjà appris durant nos années à l'IUT. L'interface graphique sera créée grâce à JAVA FX, un module JAVA dont nous avions des connaissances. Par ailleurs, nous utiliserons l'outil SceneBuilder de JAVA FX pour sa simplicité.

La modélisation de réseau est un domaine qui n'exige pas de norme obligatoire. Cependant, nous avons suivi les normes communes en matière d'ergonomie logicielle afin de faciliter la prise en main et l'utilisabilité, c'est-à-dire, un menu en haut contenant les fonctionnalités classiques d'une application, des options qui apparaissent lors d'un clic droit, le déplacement d'objets avec le Drag-And-Drop, une page d'aide,... Également, l'interface graphique sera épurée dans un souci d'ergonomie et non-contraignante (Figure 3). En effet, les utilisateurs doivent pouvoir placer les éléments comme ils l'entendent sans aucune contrainte, excepté celles relatives à la réalisation d'un réseau. De plus, les quelques fonctionnalités que nous souhaitons proposer doivent être 100% fonctionnelles car l'utilisateur pourrait par exemple perdre complètement tout le travail qu'il avait réalisé. Enfin, il est important pour nous d'assurer la fluidité d'utilisation car si ce logiciel est utilisé pour modéliser de gros réseaux composés de plusieurs réseaux d'entreprises, alors il pourrait y avoir de longs temps de chargement, ce qui rendrait pénible l'interface utilisateur.

Lors de la conception de la structure de nos différentes classes, le diagramme de classes a été réalisé (Figure 4) utilisant un patron de composition (Composite Pattern). En effet, ce modèle nous permettra d'éviter de nombreuses redondances inutiles dans le code car, par exemple, la classe "Réseau composé" possédera les propriétés de la classe "Réseau", comme une adresse IP, car elle héritera de cette dernière. De plus, un réseau devra pouvoir contenir d'autres réseaux, d'où le choix de ce Pattern qui apporte une solution adaptée à notre problème. Tous les composants hériteront de la classe "Composant" pour qu'ils aient tous les mêmes attributs, comme un identifiant, un nom, le réseau auquel il appartient.

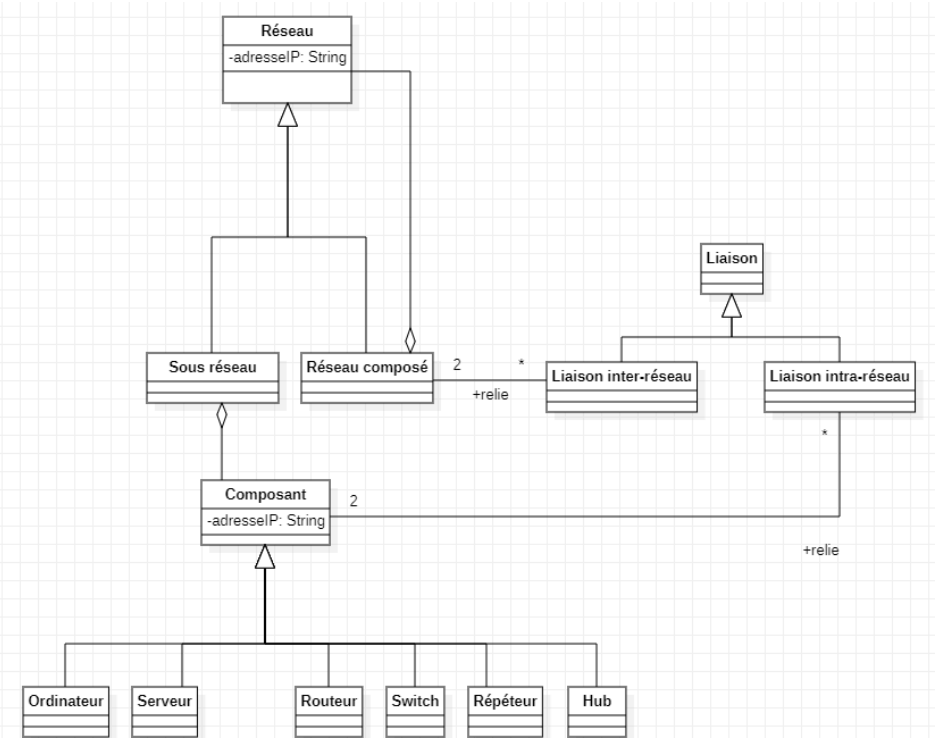

Figure 4 : Diagramme de classe de l'application

Autres contraintes

Le logiciel n'a pas vocation à être commercialisé. Bien que les quelques images utilisées pour le logiciel (routeur, switch..) soient issues d'internet, nous les avons modifiées afin de n'avoir aucun problème de droit d'auteur. Cela nous a permis de pouvoir personnaliser l'intégralité de nos composants.

Rapport technique

Conception

Présentation et justification des choix technologiques

Dès le début du projet, il a été assez facile de remarquer que le programme que nous allions devoir programmer se voulait essentiellement graphique. Lors de nos discussions avec le client, il ne nous a pas été imposé d'utiliser un langage de programmation spécifique, nous avons donc choisi JAVA et sa bibliothèque d'interface utilisateur : JAVA FX, parce que nous avions déjà utilisé ce framework lors de notre première année à l'IUT pour la réalisation d'un projet graphique dont le but était de réaliser un Escape Game. De plus, lors de ce projet, il nous a été nécessaire, pour une ou deux énigmes, d'utiliser le principe de Drag-And-Drop, qui est au cœur de notre projet actuel. Bien que cette fonctionnalité n'ait pas été mise en avant dans le jeu que nous avions réalisé, nous avons pu partir sur des bases solides et ainsi améliorer le Drag-And-Drop de notre ancien jeu qui avait été assez mal conçu.

Conception de la structure des classes

Finalement, lors de la conception, nous n'avons pas utilisé le Pattern Composite avec les réseaux composés et les sous-réseaux. Cela s'explique par un souci de temps et car cela n'apporterait pas vraiment de valeur ajoutée, notamment au niveau graphique où l'on peut toujours mettre un réseau dans un autre. Nous en avons profité pour avancer sur des fonctionnalités qui apportent une plus grande valeur ajoutée à la place. Nous avons donc utilisé une classe pour chaque type d'élément et il est possible de faire appartenir n'importe quel élément à un réseau. En revanche, l'exception est qu'un réseau ne peut pas appartenir à un autre réseau. Cependant cela reste possible graphiquement c'est-à-dire avoir un rectangle dans un autre rectangle (Figure 5).
Cependant, nous avons utilisé un autre Pattern pour notre projet, le "Singleton Pattern". Ce pattern de conception a pour but de restreindre l'instanciation d'une classe à un seul objet et permet ainsi l'unicité d'un objet. Nous l'avons utilisé pour la classe "ID" qui permet d'attribuer un numéro unique à chacun des composants présents dans le plan de travail pour ainsi pouvoir les retrouver et les sauvegarder.

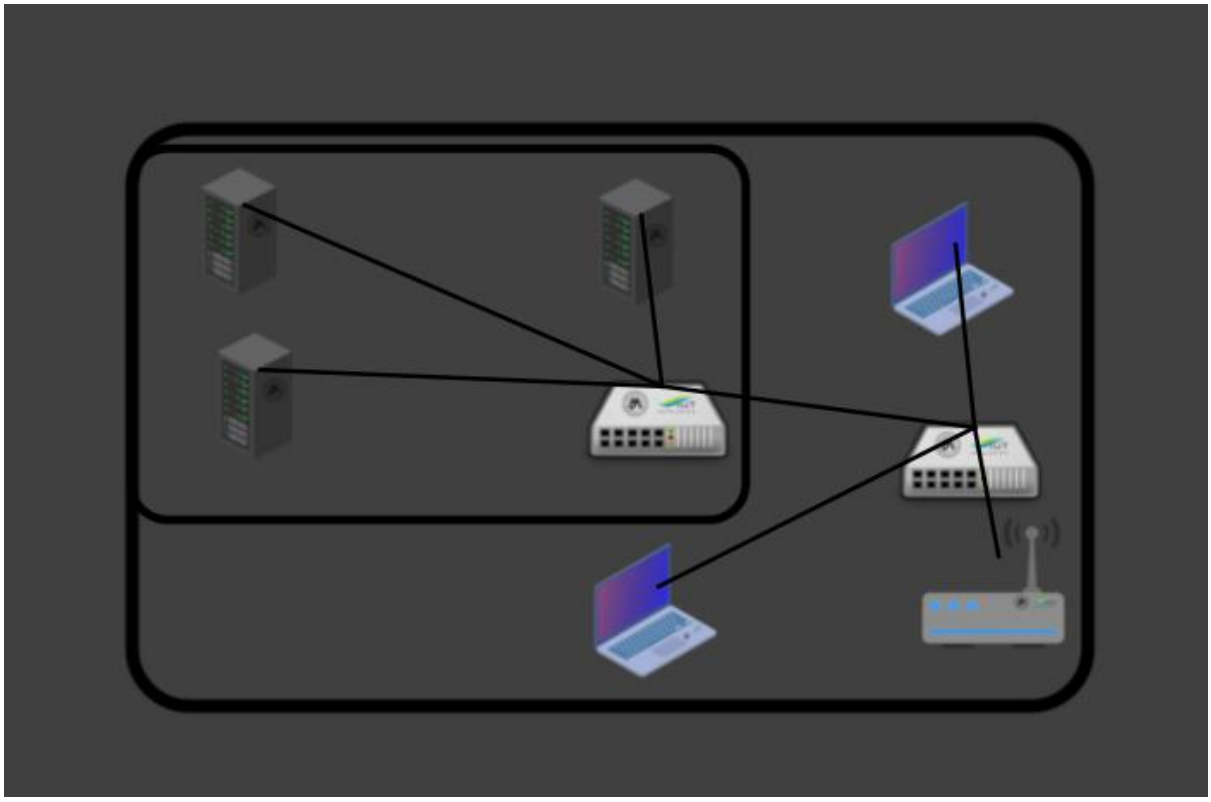

Figure 5 : Exemple d’un schéma réseau

Description des algorithmes

En JAVA FX, la classe "Contrôleur" désigne la classe qui permet de relier la partie graphique de l’application au reste du code JAVA, ainsi, les actions qu’entreprend l’utilisateur passent par cette classe. Donc si par exemple l’utilisateur place un routeur dans un réseau qu’il vient de créer avec pour adresse IP “192.168.1.1”, le “Contrôleur” créera une instance “Sous-Réseau” contenant l’adresse IP en attribut et y ajoutera une instance “Routeur” qu’il viendra de créer. Il attribuera également aux instances les positions de chaque élément. Ce programme, en plus de joindre l’interface graphique aux données conformément au modèle MVC (présenté dans la partie : Description de l’architecture du programme), rend possibles les actions utilisateur telles que le Drag-And-Drop.

Réalisation

Description de l’architecture du programme

Après avoir débattu sur le langage qui serait utilisé pour ce projet, nous nous sommes posé la question sur le type d’architecture que nous allions utiliser pour notre programme. Nous avons opté pour le modèle MVC (Modèle-Vue-Contrôleur) qui nous permet d’avoir une séparation des données (modèle), de l’affichage (vue) et des actions (contrôleur). Celui-ci offre une conception claire et efficace grâce à une bonne séparation des différentes parties du code, un gain de temps de maintenance et

d’évolution du logiciel et enfin il propose une plus grande souplesse pour organiser le développement entre les développeurs (indépendance des données, de l’affichage et des actions).

Découpage en fonctions

- **Le Drag-And-Drop**

Le but de notre projet est de fournir un logiciel d’édition facile à prendre en main pour n’importe quel type d’utilisateur, et c’est dans cette optique que nous avons mis en place un système de Drag-And-Drop. Il est extrêmement intuitif car très répandu pour les logiciels de ce type en ce qui concerne le déplacement d’éléments. C’est donc l’une de nos fonctions majeures de notre logiciel.

Pour mettre en place ce système, nous avons réalisé 3 fonctions distinctes dont chacune gère un événement spécifique relatif au Drag-And-Drop, l’une se déclenchant lors de la sélection d’un élément dans la barre de sélection des composants, l’autre lorsqu’on est en train de déplacer l’élément, et enfin une lorsqu'on le lâche dans le plan de travail.

Tout d’abord, dans la classe correspondant à la partie graphique de l’application, c’est-à-dire la classe “SceneBuilder.fxml”, nous avons ajouté à toutes les images correspondantes aux composants, l’attribut onDragDetected=”#handlerDragDetect” pour que les fonctions du Contrôleur sachent sur quels composants il faut faire un Drag-And-Drop. (Figure 6)

```
<ImageView fx:id="ordinateur"
onDragDetected="#handlerDragDetect">
                <Image url="@../resources/pc_editeur.png" />
```

Figure 6 : onDragDetect côté XML

Pour ce qui est de la zone sur laquelle l’élément à le droit d’être déposé, on attribue onDragDropped="#handlerDragDrop" et onDragOver="#handlerDragOver" ainsi que l’ID fx:id="destination" sur le plan de travail pour savoir où seront déposables les images. (Figure 7)

```
<AnchorPane fx:id="destination"
```

Figure 7 : onDragDropped et onDragOver côté XML

Pour ce qui concerne toutes les fonctions JAVA qui vont être décrites, nous rappelons que la classe permettant la liaison entre la partie graphique et le code JAVA est la classe Controller.

Tout d’abord, nous allons commencer par expliquer comment fonctionne la détection de la sélection des éléments, c'est-à-dire lorsque l’image d’un composant est choisie dans la barre de sélection. Cette fonction s’appelle “handlerDragDetect”, d’où le nom de l’attribut dans le SceneBuilder.fxml concernant les éléments pouvant être déplacés. Pour que le code de la fonction s'exécute, on vérifie si l’élément a bien été sélectionné

avec le clic gauche de la souris par “event.isPrimaryButtonDown()”, puis le code relatif au Drag-And-Drop s'exécute. (Figure 8)

```
@FXML
    public void handlerDragDetect(MouseEvent event) {
        source = ((ImageView) event.getSource());
        if (event.isPrimaryButtonDown()) {
            Dragboard db =
source.startDragAndDrop(TransferMode.ANY);
            ClipboardContent cb = new ClipboardContent();
            cb.putImage(source.getImage());
            db.setContent(cb);
            event.consume();
        }
    }
```

Figure 8 : Fonction handlerDragDetect()

Ensuite, lorsque l’on garde le clic gauche de la souris enfoncé, on peut déplacer l’image. Le curseur de notre souris est alors modifié pour faire comprendre à l’utilisateur que l’on est en train de déplacer le composant. (Figure 9)

```
@FXML
public void handlerDragOver(DragEvent event) {
        event.acceptTransferModes(TransferMode.COPY_OR_MOVE);
}
```

Figure 9 : Fonction handlerDragOver()

Lorsque l’on lâche le clic gauche de la souris, la fonction “handlerDragDrop” est appelée. L’une des principales difficultés que l’on a pu rencontrer avec cette fonction venait du fait que l’image que l’on déplaçait ne se plaçait pas à l’endroit où notre curseur se situait au moment où l’on relâchait le clic gauche de la souris. Il fallait donc centrer l’image pour qu’elle soit placée au centre du curseur de la souris. Pour ce faire, il a fallu récupérer les Coordonnées X et Y du curseur par rapport à la fenêtre actuelle et la largeur et longueur de l’image de base ainsi que la largeur et la hauteur de l’image une fois redimensionnée. Pour avoir le centre d’une image, il suffit de diviser la largeur et sa hauteur par 2. Ce processus est décrit ci-dessous (Figure 10).

```
@FXML
  public void handlerDragDrop(DragEvent event) {
      Image img = event.getDragboard().getImage();
      source2 = new ImageView();
      source2.setImage(img);

      source2.setFitHeight(75.0);
      source2.setFitWidth(75.0);
      source2.setPreserveRatio(true);

      //event.getX() == Coordonnée X du curseur dans la fenêtre
      //img.getWidth() == Largeur de l'image d'origine avant
modification
      //source.getFitWidth() == Largeur image redimensionnée
```

Figure 10 : Fonction handlerDragDrop() (Partie 1)

Une fois l'aspect graphique fait, il faut ensuite créer un nouvel élément et l'ajouter dans le réseau du côté JAVA. On crée donc un composant en fonction du nom de l'image que l'on déplace avec le Drag-And-Drop. (Figure 11)

```
        Composant cpt = null;
        if (source.getId().equalsIgnoreCase("routeur")) {
            source2.setId("routeur");
            cpt = new Routeur(source2, "168.199.1.0");
            cpt.setNom("routeur");//il me faut le type pour la sauvegarde
        } else if (source.getId().equalsIgnoreCase("ordinateur")) {
            source2.setId("ordinateur");
            cpt = new Ordinateur(source2, "168.199.1.0");
            cpt.setNom("ordinateur");
        } else if (source.getId().equalsIgnoreCase("switch")) {
            source2.setId("switch");
            cpt = new Switch(source2, "168.199.1.0");
            cpt.setNom("switch");
        } else if (source.getId().equalsIgnoreCase("serveur")) {
            source2.setId("server");
            cpt = new Serveur(source2, "168.199.1.0");
            cpt.setNom("serveur");
        }

       if (cpt != null) {
            makeDraggable(cpt);
```

Figure 11 : Fonction handlerDragDrop() (Partie 2)

- **Le système des liaisons entre composants**

```
public void lier(MouseEvent
event) {
    if (lien == 0) {
        lien = 1; //on peut lier
    } else if (lien == 1) {
        lien = 0; //on ne peut
plus
    }
}
```

Lorsqu'on clique sur le câble cette fonction se déclenche ainsi, le "mode liaison" s'active ou se désactive (variable *lien* globale de type enum)

Figure 12: Fonction lier()

Nous avons également une fonction *OnClickLiaison* permettant d'instancier les deux composants du lien (arrivée et départ) en vérifiant si les composants sont différents et que la liaison n'est pas déjà existante. (Figure 13)

```
    public void OnClickLiaison(Composant cp) {//se déclenche
lorsqu'on clique sur un composant
        cp.getImg().setOnMouseClicked(me -> {
            if (lien == 1) {
                if (liaisonsCompTab[0] == null && liaisonsCompTab[1] ==
null) {
                    liaisonsCompTab[0] = cp;
                    System.out.println("depart instancie");
                } else if (liaisonsCompTab[0] != null && liaisonsCompTab[1]
== null) {
                    if (liaisonsCompTab[0] != cp) {
                        liaisonsCompTab[1] = cp;
                        System.out.println("arrivée instancie");
                        boolean b = false;
                        //on vérifie si une liaison entre ces deux composants
n'existe pas déjà
                        for (LiaisonComposant liaisonComposant :
liaisonComposants) {
                            Composant depart = liaisonComposant.getDepart();
                            Composant arrivee = liaisonComposant.getArrivee();
                            if ((arrivee == liaisonsCompTab[0] && depart ==
liaisonsCompTab[1])
                                                    || (arrivee ==
```

Figure 13 : Fonction OnClickLiaison()

Voici donc la fonction “liaison” (Figure 14) qui permet grâce à la librairie “javafx.scene.shape.Line” de :

- tracer une ligne dans le schéma en lui donnant un point de départ et d’arrivée (avec les coordonnées x et y du plan),
- mettre à jour la liaison entre 2 composants lorsque l’on déplace l’un d’eux.

```
public void liaison(LiaisonComposant l) {
        if (l.getLine()==null) {
        //ligne qui n’existe pas encore donc on la crée
                Line line = new Line();
                //initialisation du point de départ
                line.setStartX(l.getDepart().getImg().getX() + 40);
                line.setStartY(l.getDepart().getImg().getY() + 25);
                        //initialisation du point d’arrivée
                line.setEndX(l.getArrivee().getImg().getX() + 40);
                line.setEndY(l.getArrivee().getImg().getY() + 25);
                //la largeur du trait
                line.setStrokeWidth(2);
                        //la couleur du trait
                line.setStroke(Color.BLACK);
                //le curseur au survol du trait
                line.setCursor(Cursor.HAND);
                line.setId("ligne");
                line.setOnMouseClicked(me -> OnClickLigne(line));
                l.setLine(line);
                //ajout de la ligne dans le plan
                destination.getChildren().addAll(line);
        } else {
                //lien déjà existant on le met juste à jour
                l.getLine().setStartX(l.getDepart().getImg().getX() + 40);
                l.getLine().setStartY(l.getDepart().getImg().getY() + 25);
                l.getLine().setEndX(l.getArrivee().getImg().getX() + 40);
                l.getLine().setEndY(l.getArrivee().getImg().getY() + 25);
        }
}
```

Figure 14 : Fonction liaison()

Arborescence des fichiers du programme

L’ensemble de notre projet repose sur une arborescence avec un dossier “Main” divisé en 3 sous-parties (plus une partie : Ressources), conformément au

modèle MVC : le Modèle (contient toutes les classes Java), la Vue (qui possède le fichier fxml pour l'interface graphique), et le Contrôleur (qui s'occupe de lier les classes du modèle aux éléments graphiques qui sont dans la vue). Le dossier "Ressources" contient principalement les images des icônes utilisées dans l'interface graphique. (Figure 15)

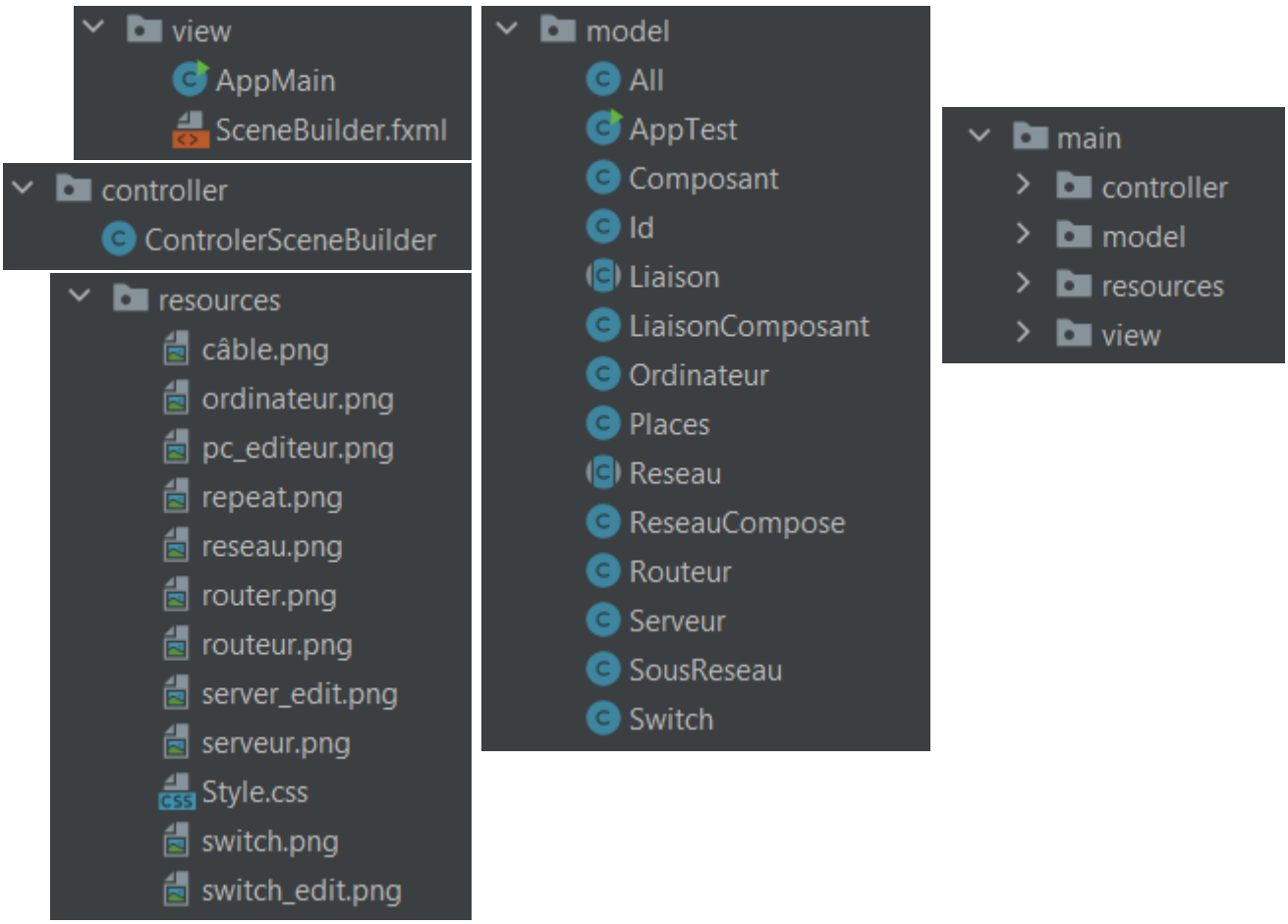

Figure 15 : Arborescence du projet

Validation, résultats et perspectives

Examen des résultats obtenus par rapport aux objectifs initiaux

Les objectifs initiaux étaient généraux, il s'agissait principalement de pouvoir ajouter des composants dans des réseaux, les lier et éditer et afficher leurs caractéristiques. Il fallait également pouvoir sauvegarder et charger depuis un fichier une configuration réseau. Ainsi, les objectifs initiaux ont tous été accomplis. De plus, le logiciel est fonctionnel et rapide. Concernant le design et l'ergonomie, nous étions plutôt libres et le résultat est à la hauteur de nos attentes. De plus, nous sommes satisfaits d'avoir réussi à implémenter une fonctionnalité pour exporter la visualisation au format PDF et PNG au choix. En effet, nous avons eu un peu de mal à réaliser cela. Cependant, concernant nos propres objectifs et prévisions, nous n'avons pas mis en place le système de sous-réseau, c'est-à-dire, le fait de faire appartenir plusieurs sous-réseaux à un réseau principal. Actuellement, il est seulement possible de le faire graphiquement. Or si cette fonctionnalité venait à être implémentée, le logiciel pourrait par la suite proposer automatiquement à l'utilisateur un plan d'adressage des sous-réseaux en fonction du réseau principal par exemple, de sorte à ce qu'il évite de rentrer manuellement ces données.

Suites possibles de développement

Actuellement, il n'est pas possible d'ajouter un composant sans avoir déjà déposé un réseau au préalable. Egalement, il n'est pas possible d'attribuer une adresse IP au mauvais format (entrer des lettres par exemple). Nous pourrions aussi ajouter un remplissage automatique concernant les adresses IP du routeur ou des composants en fonction de l'adresse IP initiale du réseau. De plus, nous pourrions limiter les liaisons de composants entre plusieurs réseaux en fonction du type de composants.

Résultats

Le logiciel obtenu est fonctionnel, c'est le critère primordial pour que notre projet soit abouti et présentable. En effet, notre chargé de projet nous a recommandé de terminer en priorité les éléments simples et d'être capables de présenter un programme fonctionnel plutôt que de vouloir chercher à commencer plusieurs fonctionnalités sans les terminer et sans pouvoir les présenter.

Nous avions un objectif global avant de commencer mais pas d'objectifs précis dès le départ, c'est notre chargé de projet qui nous demandait de nouvelles fonctionnalités au fur et à mesure de l'avancée des travaux, conformément à la méthode de travail AGILE qui sera expliquée à la partie : Rapport d'activité.

Ainsi, nous avons l'impression d'avoir répondu à l'exercice qui nous était demandé, le projet étant fini, mais tout de même améliorable.

Test et validation

Étant donné le type de fonctions que nous avions à implémenter, nous avons mis en place quelques tests algorithmiques mais également avons fait tester l'application à plusieurs utilisateurs. En effet, dans le but de tester l'ergonomie de l'interface graphique, rien de mieux que de le faire essayer par d'autres personnes. Bien sûr, nous n'avons pas uniquement testé l'interface graphique. Nous avons également testé la sauvegarde de fichier et son chargement, ainsi que les attributions des adresses IP. Nous avons finalement vérifié que les appartenances entre réseaux et composants et uniquement entre réseaux étaient valides dans plusieurs situations. En effet, il doit être possible de changer le composant de réseau par exemple.

Manuel d'installation

Le logiciel est fourni sous forme d'archive .zip pour être léger. Il faut le décompresser puis lancer le fichier exécutable : EditeurDeConfigurationReseau.exe.

Manuel d'utilisation

Une fois le logiciel lancé, vous apercevez un panneau latéral à gauche comportant certains éléments visuels, il s'agit des outils permettant d'éditer votre schéma dans la scène se trouvant à droite. Le nom d'un outil est indiqué au survol de celui-ci.

La première chose à faire est de placer un réseau dans le plan de travail, pour cela il faut cliquer sur le rectangle dans la barre de sélection des composants, maintenir appuyé avec un clic gauche et déplacer le curseur où vous souhaitez placer votre réseau dans le plan de travail, c'est-à-dire la zone centrale de la fenêtre. Vous pouvez également agrandir ce réseau en faisant un clic gauche sur sa bordure noire tout en maintenant appuyé et en déplaçant la souris pour l'agrandir ou le diminuer,

relâchez pour stopper l'action. Vous pouvez faire de même pour déplacer le réseau mais en maintenant la touche CTRL et le clic gauche. Vous pouvez maintenant placer des composants de la même manière : il s'agit là du principe de Drag-And-Drop.

Pour ce qui est de la liaison, il faut appuyer sur le câble dans le volet latéral gauche afin d'activer ou désactiver le mode liaison. Si celui-ci est en surbrillance, alors le mode liaison est activé, dans le cas contraire, il ne l'est pas. Il suffit donc ensuite de cliquer sur un élément puis sur un deuxième et une droite apparaîtra entre les deux. Vous pouvez lier autant de composants que vous le souhaitez.

Pour attribuer ou changer l'adresse l'IP d'un élément, ou pour tout simplement le supprimer, il suffit de faire un clic droit dessus puis de cliquer sur l'option correspondante dans le menu qui apparaitra. Attention, lors de la suppression d'un réseau, tous ses composants et liaisons seront supprimés.

Pour sauvegarder ou charger un projet, il suffit de cliquer sur le menu d'en-tête et de choisir l'élément correspondant. Vous avez également la possibilité d'exporter votre schéma au format PNG ou PDF (si celui-ci n'est pas vide).

Rapport d’activité

Dans cette partie, nous présenterons la manière dont nous avons géré ce projet qui s’est étalé sur une période d’environ 3 mois.

Méthode de développement et outils

Pour le développement de ce projet, nous avons utilisé deux outils : IntelliJ IDEA, pour le développement de l’application, ainsi que GIT, pour l'hébergement du code de notre logiciel. IntelliJ est un IDE qui facilite le développement car il assure l'automatisation de certaines tâches et peut éventuellement réaliser des opérations de simplification du langage. GIT est un logiciel de gestion de versions du code d’un projet informatique. Pour faire simple, il s’agit d’un programme qui crée différentes versions d’un projet au fur et à mesure que les fichiers sont édités. Il suffit simplement de synchroniser, de manière manuelle ou automatique (à intervalle régulier par exemple), votre code sur votre machine locale, vers le serveur sur lequel il est hébergé. Pour la planification des rendez-vous avec notre tuteur, tout passait les mails pour fixer la date de nos rendez-vous. Avant le confinement, les rendez-vous se passaient dans son bureau, puis ils ont continué sur la plateforme Discord.

Planification des tâches

Pour la réalisation de notre projet, nous avons mis en place la méthode SCRUM car c’est une méthodologie AGILE qui nous permet de livrer un produit testable à la fin de chaque livrable, autrement à chaque rendez-vous avec le client. Pour l’organisation du projet, nous avons utilisé deux outils primordiaux pour utiliser la méthode SCRUM : la méthode GANTT, permettant de planifier à l'avance la durée des tâches (Figure 16), ainsi que GITLAB pour l’organisation des différentes tâches que nous devions accomplir pour la réalisation de notre projet (Figure 17). GITLAB nous permettait de créer un tableau de tâches regroupant 3 colonnes : TO DO pour les tâches à accomplir, DOING, pour celles en cours de réalisation, et CLOSED (ou DONE) pour les tâches terminées. Cette plateforme est très pratique car elle nous a permis de planifier chacune de nos tâches au début de chaque nouveau Sprint en estimant la valeur de chaque tâche et ainsi savoir lesquelles étaient primordiales. Certaines de ces tâches n’ont pas forcément été réalisées dans les délais d’un Sprint, mais ont donc été réalisées sur plusieurs.

	Nom	Durée	Début	Fin	Prédécesseur
1	Analyse : Diagrammes, choix du langage	7 jours	30/11/20 08:00	08/12/20 17:00	
2	Icones des éléments, drag and drop	7 jours	09/12/20 08:00	17/12/20 17:00	1
3	Liaisons d'éléments, réseaux, amélioration	7 jours	18/12/20 08:00	28/12/20 17:00	2
4	Rédaction du rapport	15 jours	30/11/20 08:00	18/12/20 17:00	
5	Tables de routage, sauvegarder, charger	9 jours	29/12/20 08:00	08/01/21 17:00	3
6	Fin de la rédaction du rapport	15 jours	21/12/20 08:00	08/01/21 17:00	4

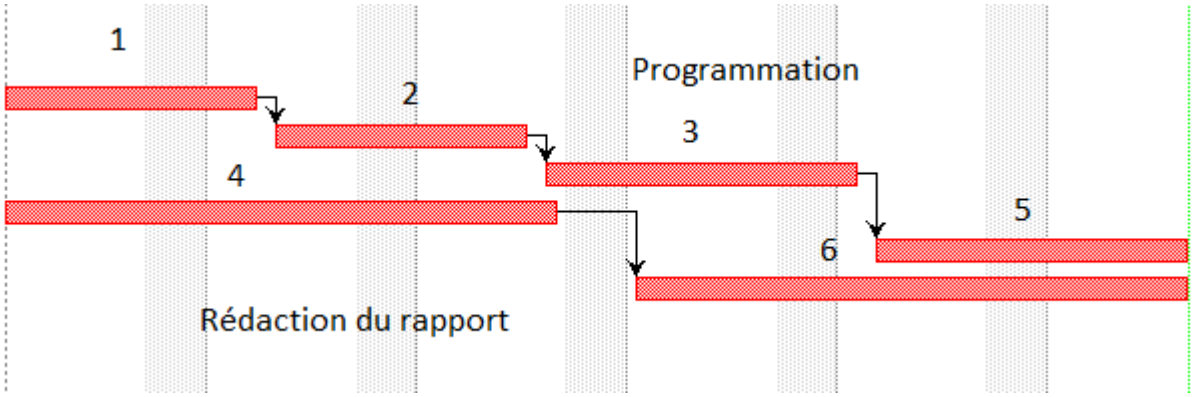

Figure 16 : GANTT

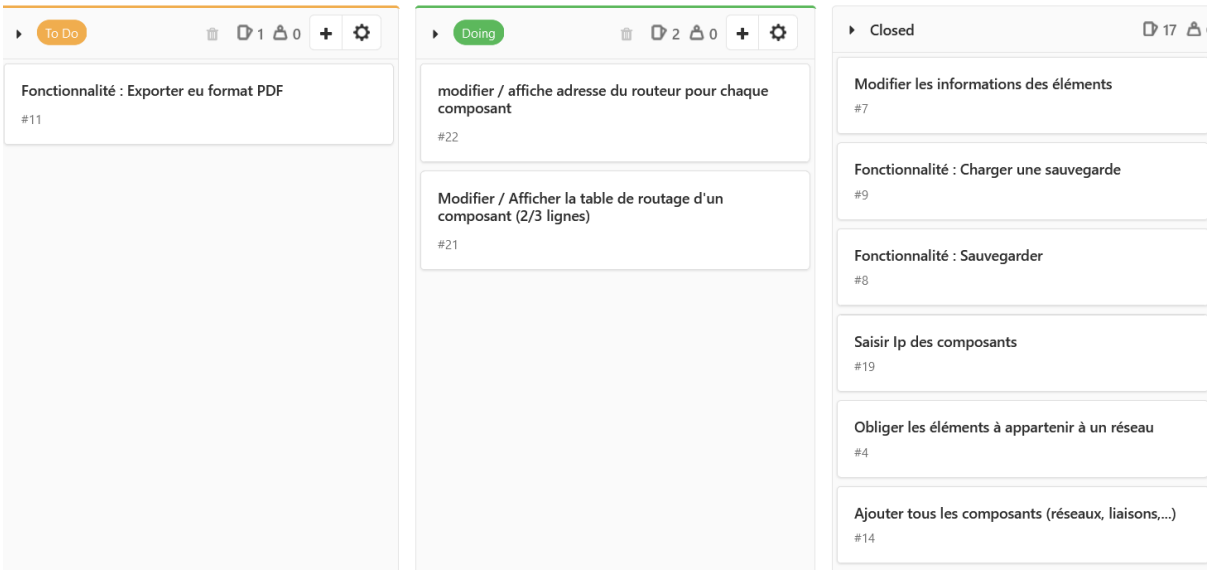

Figure 17 : GITLAB

Bilan critique par rapport au cahier des charges

Nous sommes parvenus à rester plutôt constants sur toute la durée du projet. Certaines semaines étaient plus productives que d'autres mais les réunions hebdomadaires nous obligeaient à produire régulièrement dans tous les cas. Nous avons donc bien géré le projet, nous avons réalisé les fonctionnalités importantes au fur et à mesure afin de ne pas être submergés de travail à la fin et devoir faire trop de concessions. Nous sommes donc assez satisfaits de notre travail. De plus, nous sommes très satisfaits du choix de l'équipe car tous les membres sans exception ont été très impliqués dans la réalisation de ce projet, ce qui a renforcé la motivation du groupe. Généralement, chaque personne travaillait de son côté (ou en binôme) et nous nous répartissions les tâches et expliquions nos avancées. Ceci nous permettait d'avoir plusieurs objectifs lors de chaque sprint. Nous avons également beaucoup communiqué ensemble et nous nous sommes fait confiance. En effet, après chaque réunion et régulièrement nous faisions le point sur ce qu'il nous restait à faire et sur l'avancée des travaux entre autres. Comme pistes d'amélioration, nous aurions peut-être pu travailler tous au même moment, cela aurait été certes contraignant mais peut-être encore plus prolifique.

Annexes

Bibliographie

« MouseEvent (Java Platform SE 7) ». https://docs.oracle.com/javase/7/docs/api/java/awt/event/MouseEvent.html#MOUSE_DRAGGED.

« Le Tutoriel de JavaFX ContextMenu », *devstory*. https://devstory.net/11115/javafx-contextmenu .

B. Jacques, « IHM-1 JavaFX - 10 Boîtes de dialogue simples et spécialisées », p. 32. http://remy-manu.no-ip.biz/Java/Tutoriels/JavaFX/PDF/ihm1_fx_10_man.pdf

« Introduction à JSON-Java (org.json) ». https://www.codeflow.site/fr/article/java-org-json .

« Cyril Rabat ». https://www.cyril-rabat.fr/articles/index.php?article=50.

« Exemple JSON.simple - Lecture et écriture JSON ». https://www.codeflow.site/fr/article/java__json-simple-example-read-and-write-json .

« Java: Créer un fichier PDF - | ». http://java.mesexemples.com/fichiersrepertoires/java-creer-un-fichier-pdf/ .

« How can I convert a PNG file to PDF using java? », *Stack Overflow*. https://stackoverflow.com/questions/8361901/how-can-i-convert-a-png-file-to-pdf-using-java .

Résumé

Le projet d'Éditeur Réseau est un logiciel qui permet d'aider les techniciens de l'IUT ou de n'importe quelle structure à créer et gérer la configuration du réseau local. En effet l'utilisateur peut visualiser la disposition des équipements, leurs liens et leurs caractéristiques (adresse, routeur associé). Le logiciel a été développé en Java et conformément à la structure MVC. Il combine également l'utilisation de JavaFX pour l'interface graphique ainsi que le langage JSON qui s'occupe de la persistance des données pour gérer la sauvegarde d'une configuration que l'utilisateur aura créé.

Mots clés : Java, Développement logiciel, Java FX, MVC, JSON, Gitlab.

Summary

The Network Editor project is a software that can help the technicians from the IUT or any other structure to create and manage the configuration of the local network. Indeed the user can visualize the layout of the equipments, their links and their characteristics (address, associated router). The software has been developed in Java and according to the MVC structure. It also combines the use of JavaFX for the graphical interface as well as the JSON language which deals with data persistence to manage the saving of a configuration the user has created before.

Keywords : Java, Software development, Java FX, MVC, JSON, Gitlab.