Rapport Général

Iteration 2 : Du 03/06/2020 au 09/06/2020

Cahier des charges

Nous avons d'abord défini le cahier de charges pour bien cerner le besoin des différents utilisateurs du système, définir les différentes fonctions principales et fonctions contraintes du système et enfin, envisager différentes réponses à ces besoins.

CF : cahier_des_charges.odt

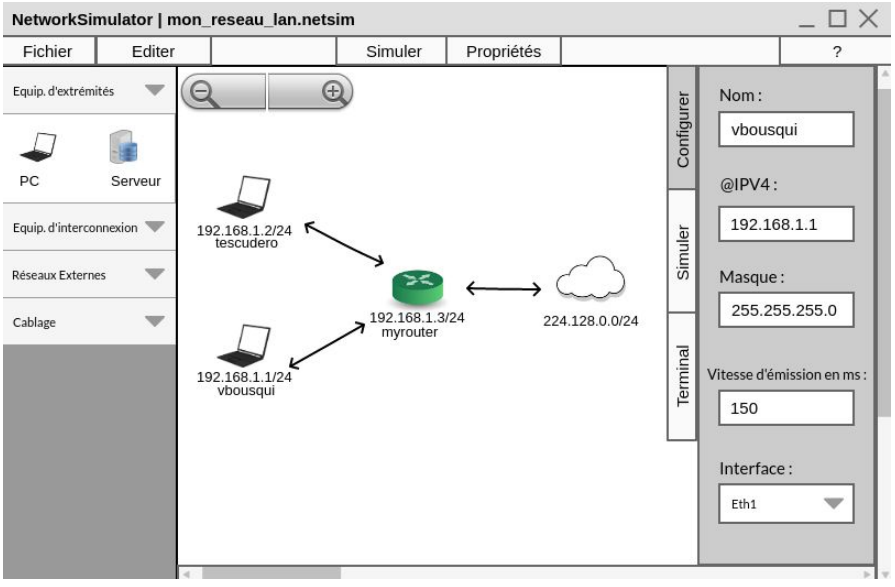
Maquettes

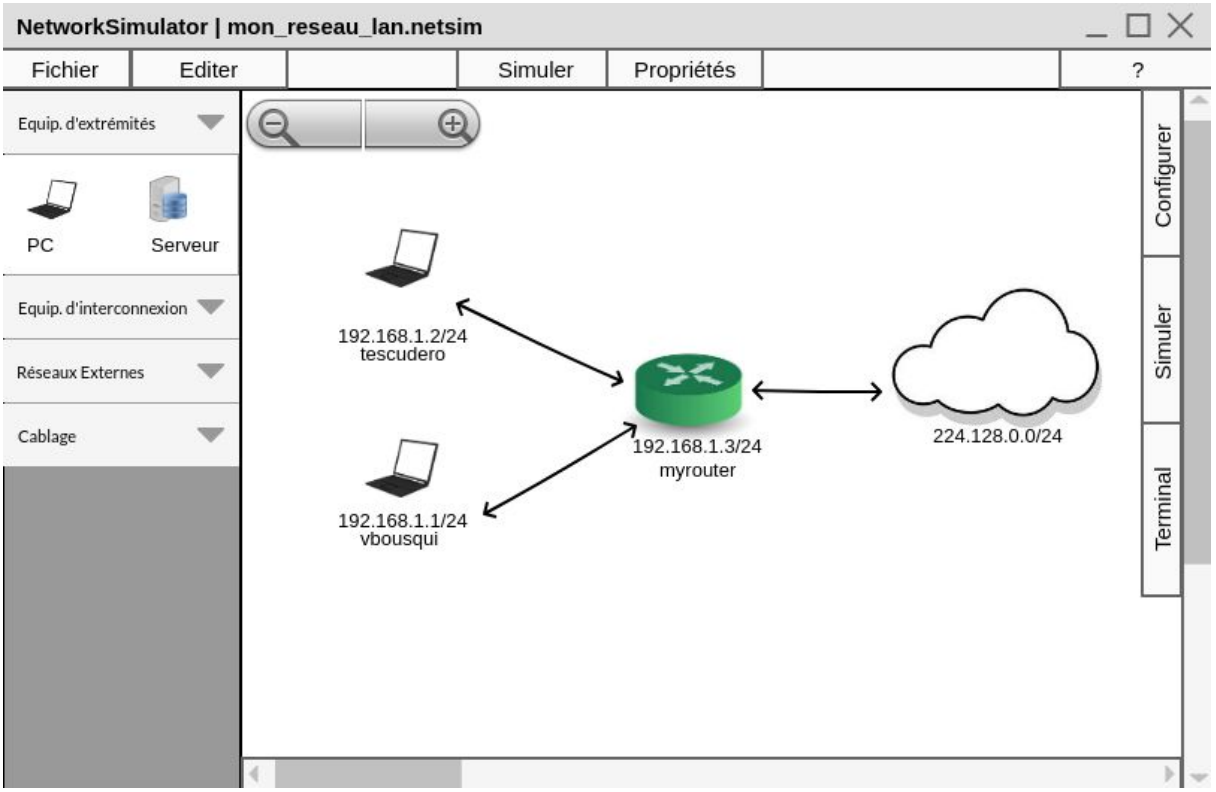
Nous avons ensuite fait plusieurs maquettes pour représenter le système que l'on voudrait mettre en place, avec chacun sa représentation.

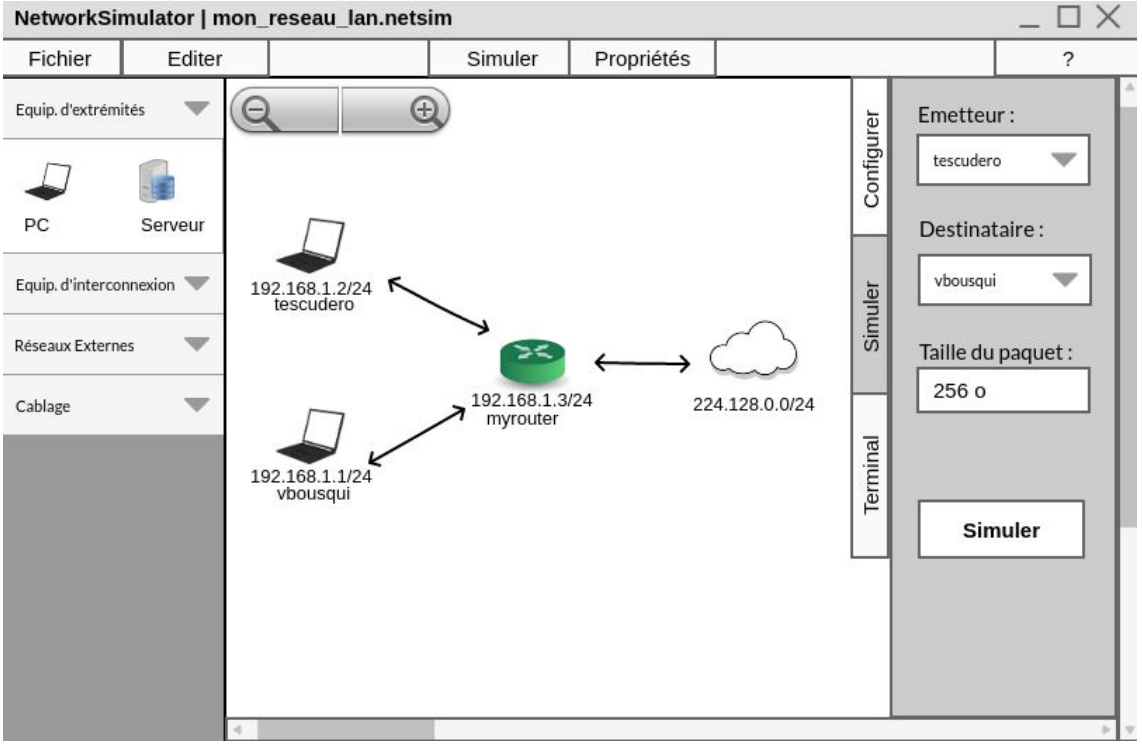
CF : maquette_ALFRED.epgz
CF : maquette_BOUSQUIE.epgz
CF : maquette_TEISSEDRE.epgz

Les maquettes ont été réalisées avec le logiciel PENCIL.

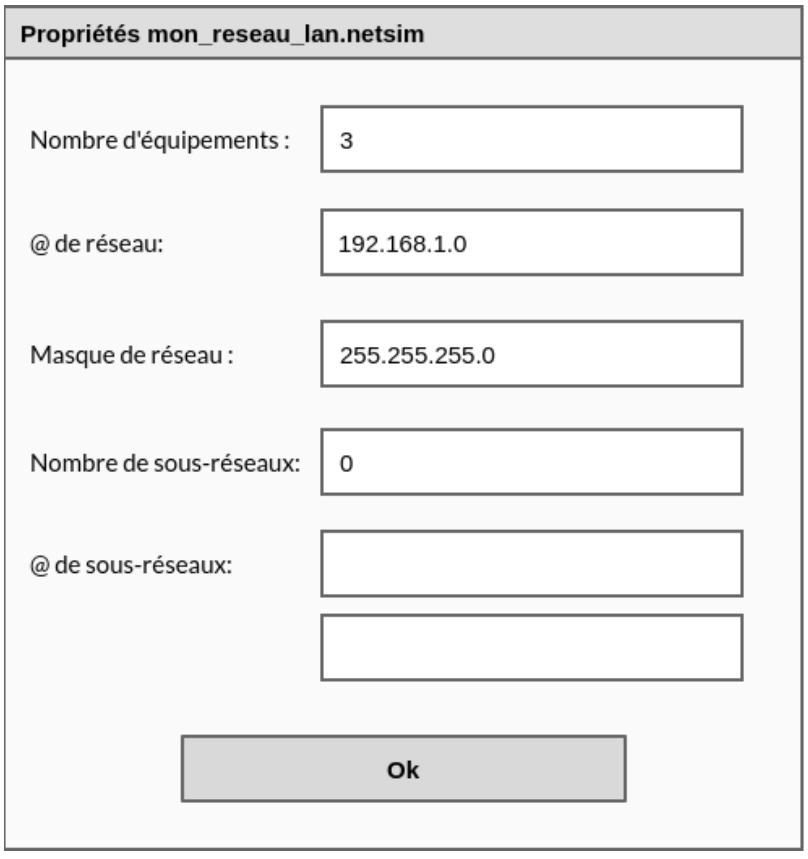
Nous nous sommes ensuite mis d'accord sur une maquette regroupant les avantages de chacune des précédentes :




NetworkSimulator | mon_reseau_lan.netsim
Fichier
Editer
Simuler
Propriétés
?
Equip. d'extrémités
PC
Serveur
Equip. d'interconnexion
Réseaux Externes
Cablage
192.168.1.2/24
tescudero
192.168.1.3/24
myrouter
224.128.0.0/24
192.168.1.1/24
vbousqui
Configurer
Simuler
Terminal


NetworkSimulator | mon_reseau_lan.netsim
Fichier
Editer
Simuler
Propriétés
?
Equip. d'extrémités
PC
Serveur
Equip. d'interconnexion
Réseaux Externes
Cablage
192.168.1.2/24
tescudero
192.168.1.3/24
myrouter
224.128.0.0/24
192.168.1.1/24
vbousqui
Configurer
Simuler
Terminal
Emetteur :
tescudero
Destinataire :
vbousqui
Taille du paquet :
256 o
Simuler

```
mon_reseau_lan $ >    ls
vbousqui
tescudero
myrouter

mon_reseau_lan $ >    cd tescudero

mon_reseau_lan / tescudero >    ipconfig    192.168.1.2  255.255.255.0

mon_reseau_lan / tescudero >    ip
192.168.1.2 /24

mon_reseau_lan / tescudero >    ping vbousqui

Ok .
mon_reseau_lan / tescudero >    ping mteissedre
Réseau Inconnu.
```



Gestion de projet

Outil

Pour organiser notre projet nous avons mis en place un tableau de bord sur Trello. Ainsi on a pu décomposer le projet en sous-tâches, grâce aux users stories. Cet outil nous permet de mettre à jour l'avancement d'une tâche grâce à des TODO List.

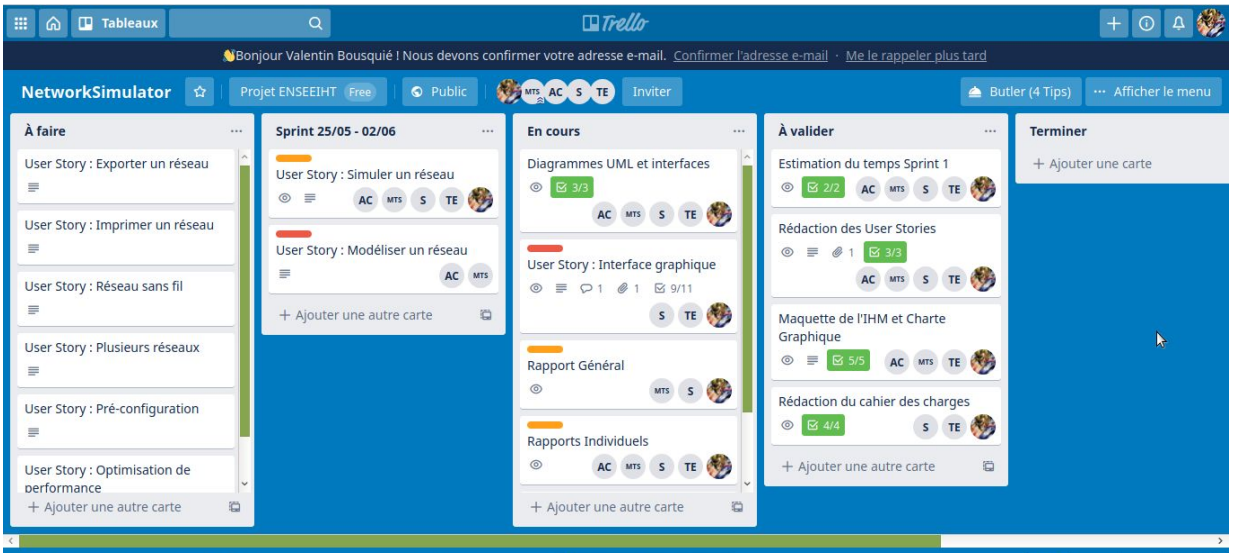
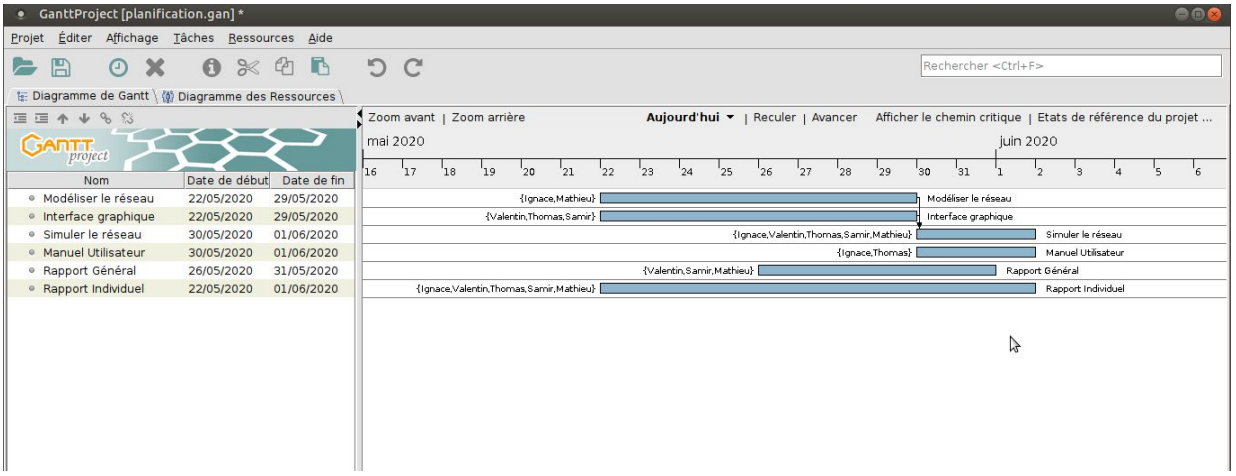


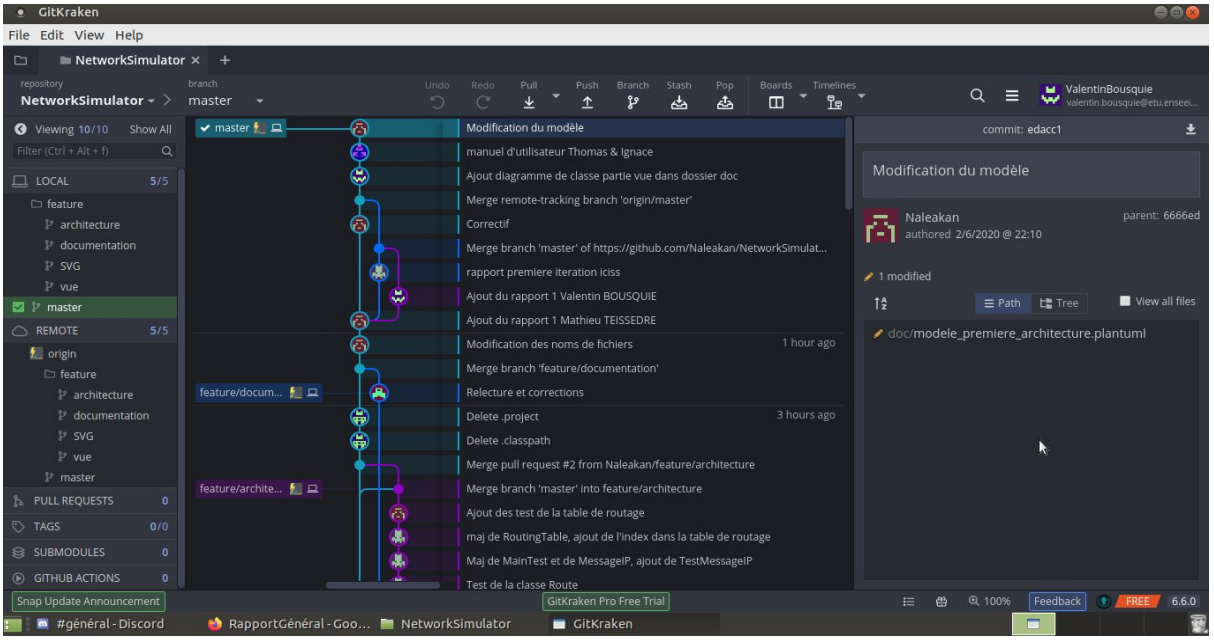
Diagramme de Gantt

Pour nous aider à planifier le sprint nous avons décidé de créer un diagramme de Gantt avec l'outil GanttProject.



Suivi de version

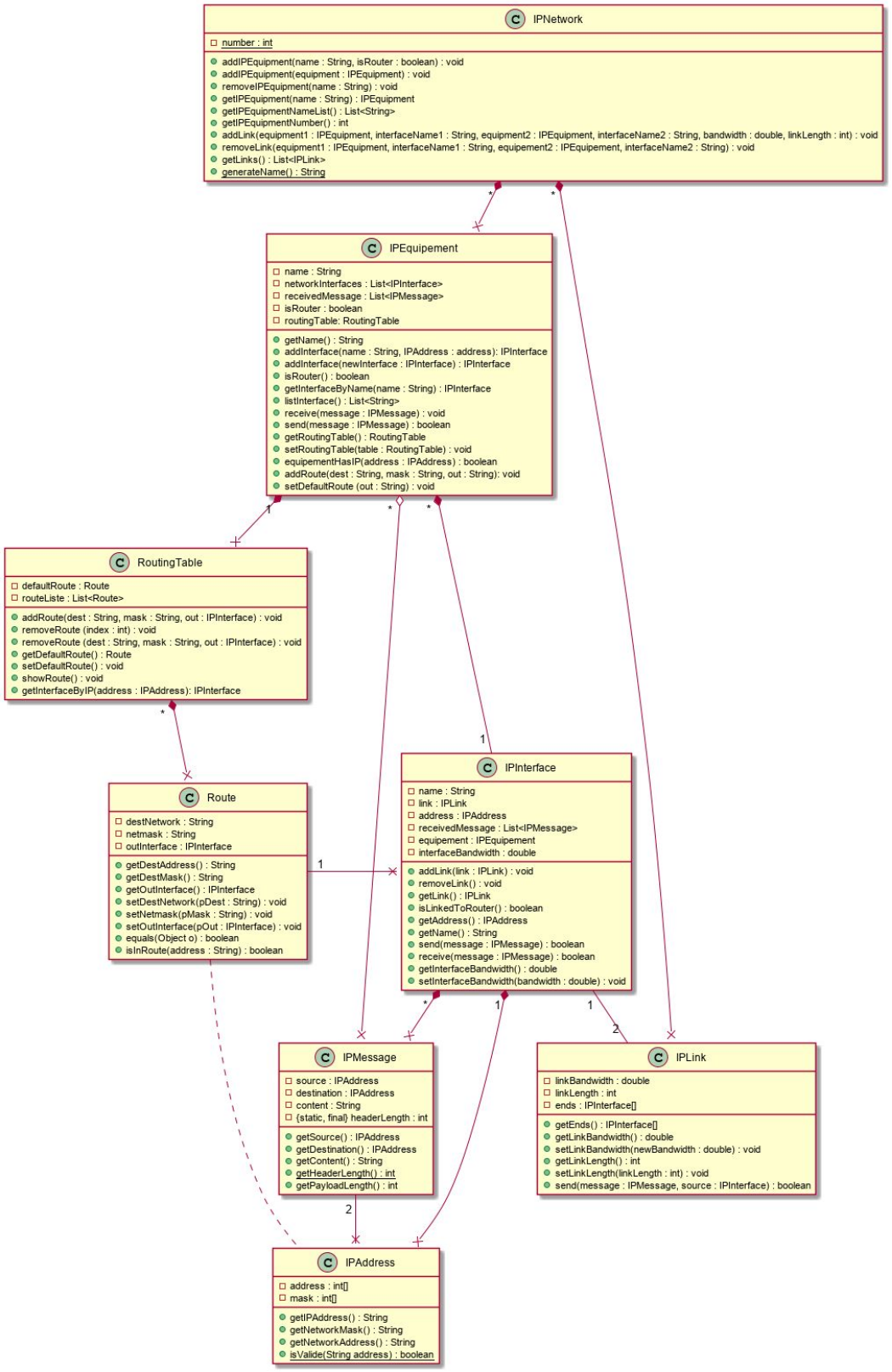
Pour gérer la problématique du développement à plusieurs nous avons décidé de mettre en place un remote sur GitHub. Ensuite grâce à git nous avons pu créer différentes branches en fonction des fonctionnalités. Pour faciliter l'utilisation de git nous avons utilisé un outil graphique permettant d'afficher l'arbre des commits. Il s'agit de GitKraken :



Cet outil graphique a permis de mieux appréhender le système de branche de git.
Il permet également de mieux visualiser les fichiers qui ont été modifiés avant de commit.

Conception du modèle

Diagramme de classe



Nous avons d'abord commencé à faire un diagramme de classe composé d'interfaces pour représenter le réseau de manière générique, mais nous nous sommes rendus compte que cette conception était bancale. Nous avons donc décidé de nous limiter au niveau IP.

Chaque équipement aura comme attributs un nom, une file de réception, une table de routage, une précision de s'il s'agit d'un routeur ou pas et un ensemble d'interfaces Réseau. Sur un équipement IP, on pourra envoyer et recevoir des messages, modifier la table de routage, ajouter des interfaces IP.

Chaque classe a été implémentée en Java, testée avec JUnit et différents tests d'intégration ont été réalisés pour vérifier que toutes les méthodes se coordonnent bien.

Un objet adresse IP est composé de deux tableaux d'entiers que sont l'adresse IP et le masque. Elle implémente également une méthode qui renvoie l'adresse réseau correspondante. Toutes les adresses renvoyées sont sous forme de chaîne de caractères, ce qui permet d'être plus lisible.

Un message IP est composé d'une adresse source, de l'adresse de destination et du contenu du message. On peut obtenir la taille d'un message IP pour pouvoir faire des calculs sur le temps du paquet pour parcourir le réseau.

Enfin, une table de routage est une liste chaînée de routes. Une route est composée de quatre informations: l'adresse du réseau de destination, son masque, l'adresse de la passerelle et l'interface de sortie. On pourra ajouter une route à cette table, supprimer une route de deux manières: en donnant en paramètre la route donnée ou en donnant l'index de la ligne à supprimer.

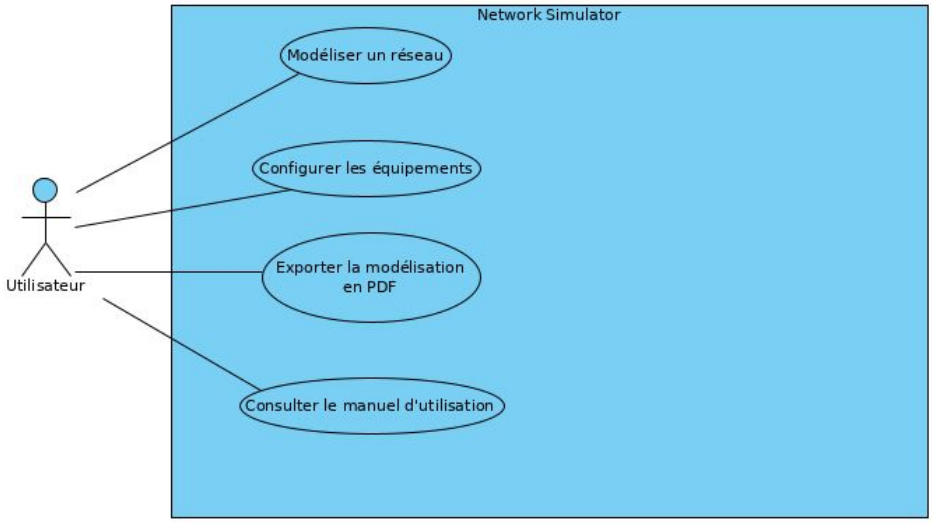
Le routage a été implémenté de la manière suivante: l'appel à la méthode envoyer sur un équipement avec les trois paramètres mentionnés plus haut fait appel à la méthode envoyer sur l'interface de sortie correspondante. Cette interface est obtenue par une méthode de parcours de la table de routage qui renvoie l'interface de sortie en fonction de l'adresse de destination.

Enfin, nous avons une classe Réseaux IP qui permet de gérer des équipements IP et les liens entre les différentes interfaces.

Si lors de la réception, l'équipement n'est pas destinataire mais qu'il est routeur, il refait appel à la méthode envoyer pour retransmettre le message. S'il n'est pas routeur, le paquet est supprimé et une exception est levée.

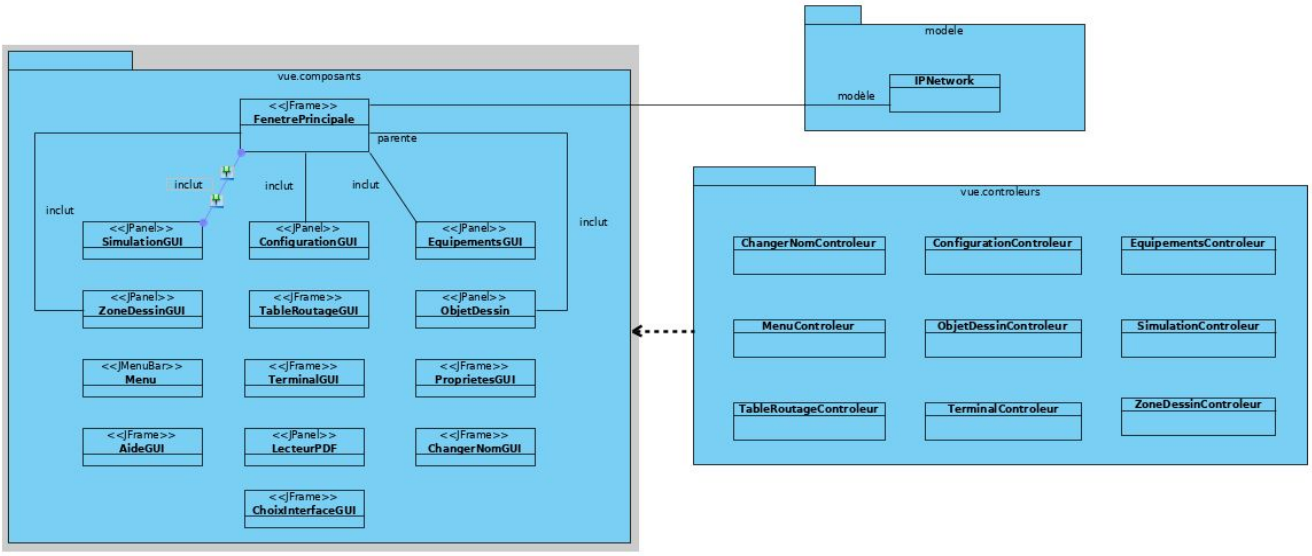
Conception de la Vue

Diagramme de cas d’utilisation



Ce diagramme de cas d’utilisation représente les fonctionnalités disponibles dans la version actuelle de l’application. L’utilisateur peut désormais supprimer des équipements et tracer des liens contrairement à la version 1. Il peut également exporter sa modélisation en PDF et lire le manuel utilisateur grâce à un lecteur PDF. Enfin, l’utilisateur peut configurer en partie ses équipements.

Diagramme de classes de la Partie Vue



Ce diagramme de classes représente les composants graphiques de notre application. Il est simplifié et met en avant les principales relations entre les composants et les classes fournies par le package javax.swing. On a également fait apparaître les relations entre contrôleurs, vue et modèle qui sont apparues lors de l'intégration.

Nous avons décidé de décomposer notre fenêtre principale en plusieurs composants graphiques. Le Borderlayout a été choisi pour le management de la disposition de cette fenêtre. Ainsi chaque position du layout est remplie avec un composant qui hérite (spécialise) la classe JPanel.

La partie "Vue" a été bien avancée au cours du sprint 1. En effet, la grande majorité des composants graphiques a été développée.

Durant le 2nd sprint, nous avons développé la totalité des composants graphiques et avons commencé à intégrer vue et modèle. Pour l'instant, l'utilisateur peut configurer en partie les équipements.

Durant le 3ème sprint, nous travaillerons essentiellement sur de l'intégration et sur le développement des contrôleurs. On ajoutera la possibilité de tester la configuration réalisée par l'utilisateur.

Bilan 1er Sprint

Le premier sprint nous a permis de définir les grandes lignes de notre application, que ce soit de la représentation du réseau au niveau Vue, que de sa simulation au niveau Modèle.
Ce sprint nous a aussi permis de mettre en place les outils nécessaires pour travailler ensemble efficacement.

Il reste encore beaucoup de choses à améliorer pour le prochain sprint, comme le terminal pour envoyer des commandes, les liens entre les objets graphiques, les envois de messages ou la configuration des équipements.

Bilan 2nd Sprint

Le second sprint nous a permis d'améliorer la base réalisée lors du premier sprint et de commencer à intégrer les différentes éléments. Nous avons pu corriger les problèmes qui sont apparus au fur et à mesure. Il faut maintenant finir d'assembler les différents composants de l'application et faire des tests de réseaux. Il y aura également des fonctionnalités supplémentaires à implémenter lors du prochain sprint.

En début de projet nous avons également redéfini l'User Story "Simuler un réseau". En réalité nous souhaitons que l'utilisateur puisse tester sa configuration sans se soucier des problèmes de surcharge d'un équipement ou autre… L'utilisateur doit savoir si un message peut être envoyé d'un équipement à un autre. Cela lui permettra de vérifier s'il a bien rempli les tables de routage par exemple.