

A thick dark blue vertical bar is positioned on the left side of the page. A blue arrow-shaped banner points to the right from this bar, containing the date. Below the banner, several thin, curved lines in dark blue and light grey sweep upwards from the bottom left corner.

05/2020 au 06/2020

Rapport général

Modélisation et simulation de réseau

BOUSQUIE Valentin, CISS Ignace, ESCUDERO
Thomas, FAKOREDE Samir et TEISSEDRE Mathieu
APPRENTIS ENSEEIHT 1ER ANNEE

Table des matières

I.	Préparation du projet.....	2
A.	Cahier des charges.....	2
B.	Maquettes	2
C.	Gestion de projet.....	5
1.	Trello.....	5
2.	Diagramme de Gantt	5
3.	Git et GitHub.....	6
II.	Conception du modèle	7
A.	Diagramme de classe.....	7
III.	Conception de la vue	9
A.	Diagramme de cas d'utilisation	9
B.	Diagramme de composant	10
IV.	Conclusion	11
A.	Bilan du 1 ^{er} sprint	11
B.	Bilan du 2 nd sprint.....	11
C.	Bilan du 3 ^{ème} sprint	11
D.	Evolution possible	11
V.	Liste des illustrations.....	12

I. Préparation du projet

A. 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 : *doc/cahier_des_charges.odt*

B. Maquettes

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

CF : *doc/maquette_ALFRED.epgz*

CF : *doc/maquette_BOUSQUIE.epgz*

CF : *doc/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 :

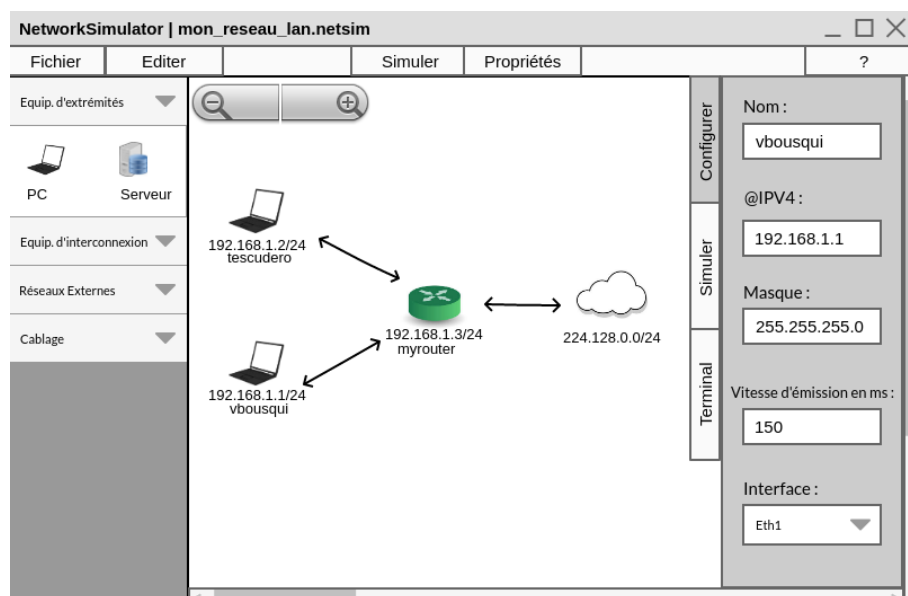


Figure 1: Aperçu de l'onglet configurer

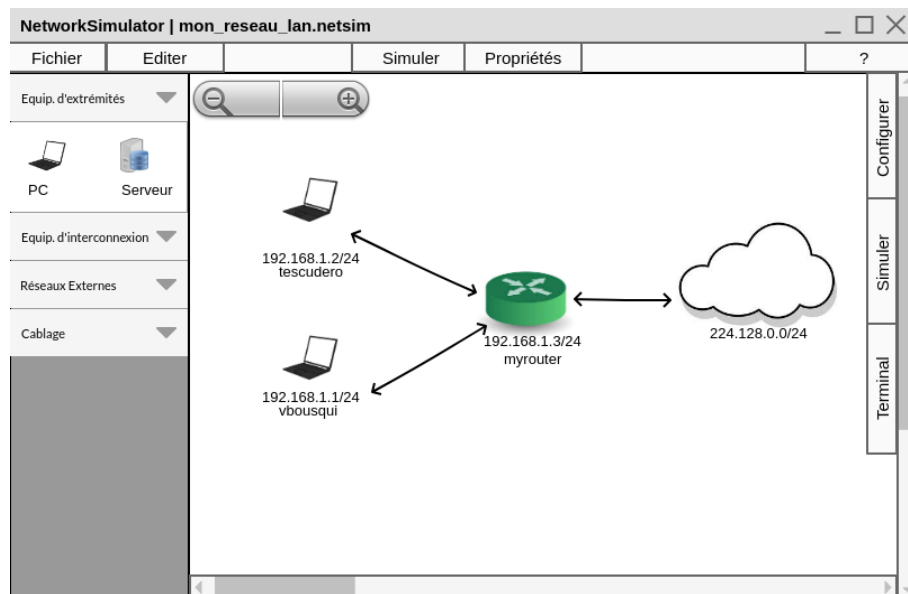


Figure 2: Aperçu de la vue sans onglet

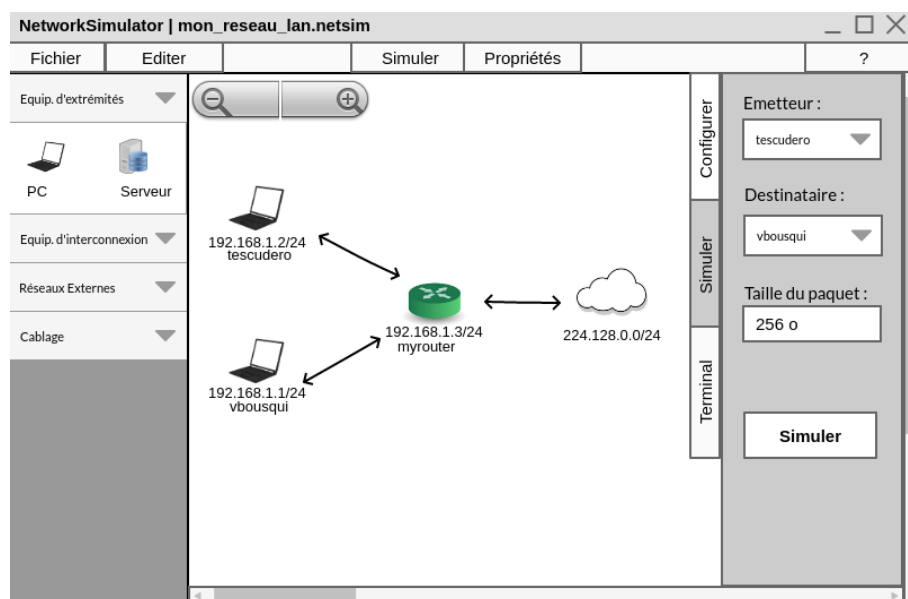


Figure 3: Aperçu de l'onglet émettre

```
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.
```

Figure 4: Aperçu du terminal

Propriétés mon_reseau_lan.netsim	
Nombre d'équipements :	<input type="text" value="3"/>
@ de réseau:	<input type="text" value="192.168.1.0"/>
Masque de réseau :	<input type="text" value="255.255.255.0"/>
Nombre de sous-réseaux:	<input type="text" value="0"/>
@ de sous-réseaux:	<input type="text"/>
	<input type="text"/>
<input type="button" value="Ok"/>	

Figure 5: Aperçu de la configuration générale du réseau

C. Gestion de projet

1. Trello

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.



Figure 6: Aperçu du Trello

2. Diagramme de Gantt

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

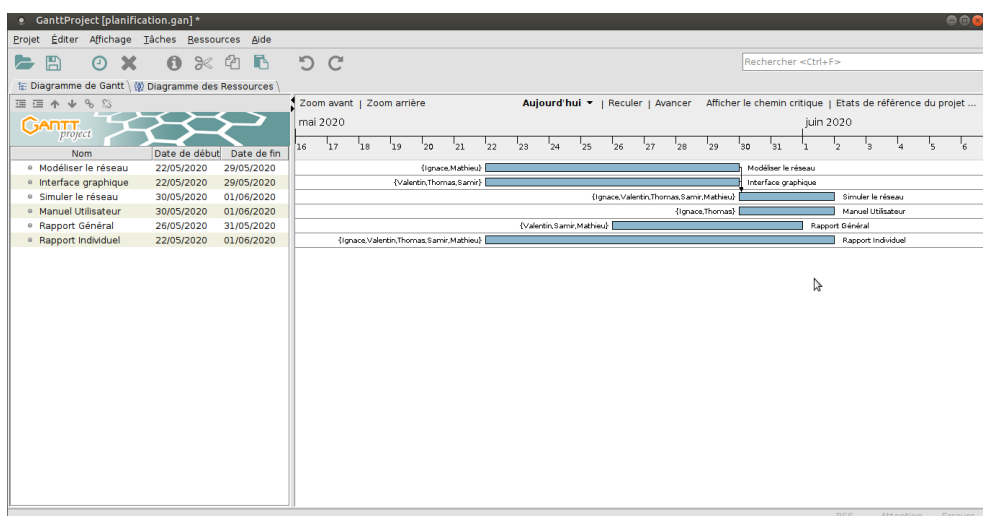


Figure 7: Aperçu de GanttProject

3. Git et GitHub

Pour gérer la problématique du développement à plusieurs nous avons décidé de mettre en place l'origine de nos sources sur GitHub. Ensuite, à l'aide de Git nous avons pu créer différentes branches en fonction des fonctionnalités. Pour rendre l'utilisation de Git plus conviviale pour ce qui ne connaissait que peu cet outil, nous avons utilisé un outil graphique permettant d'afficher l'arbre des commits. Il s'agit de GitKraken :

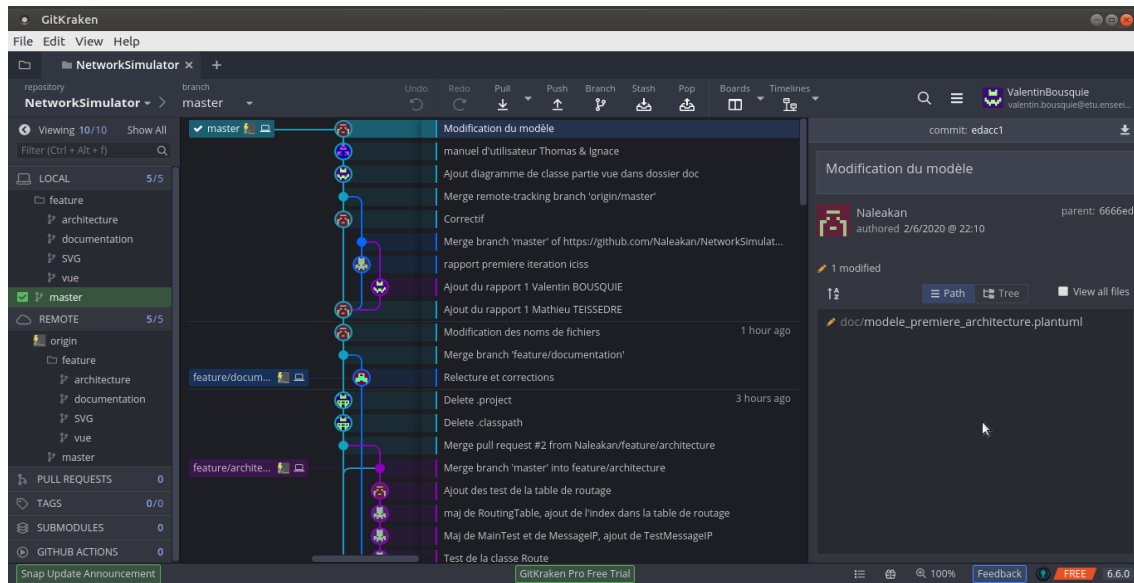


Figure 8: Aperçu de GitKraken

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.

II. Conception du modèle

A. Diagramme de classe

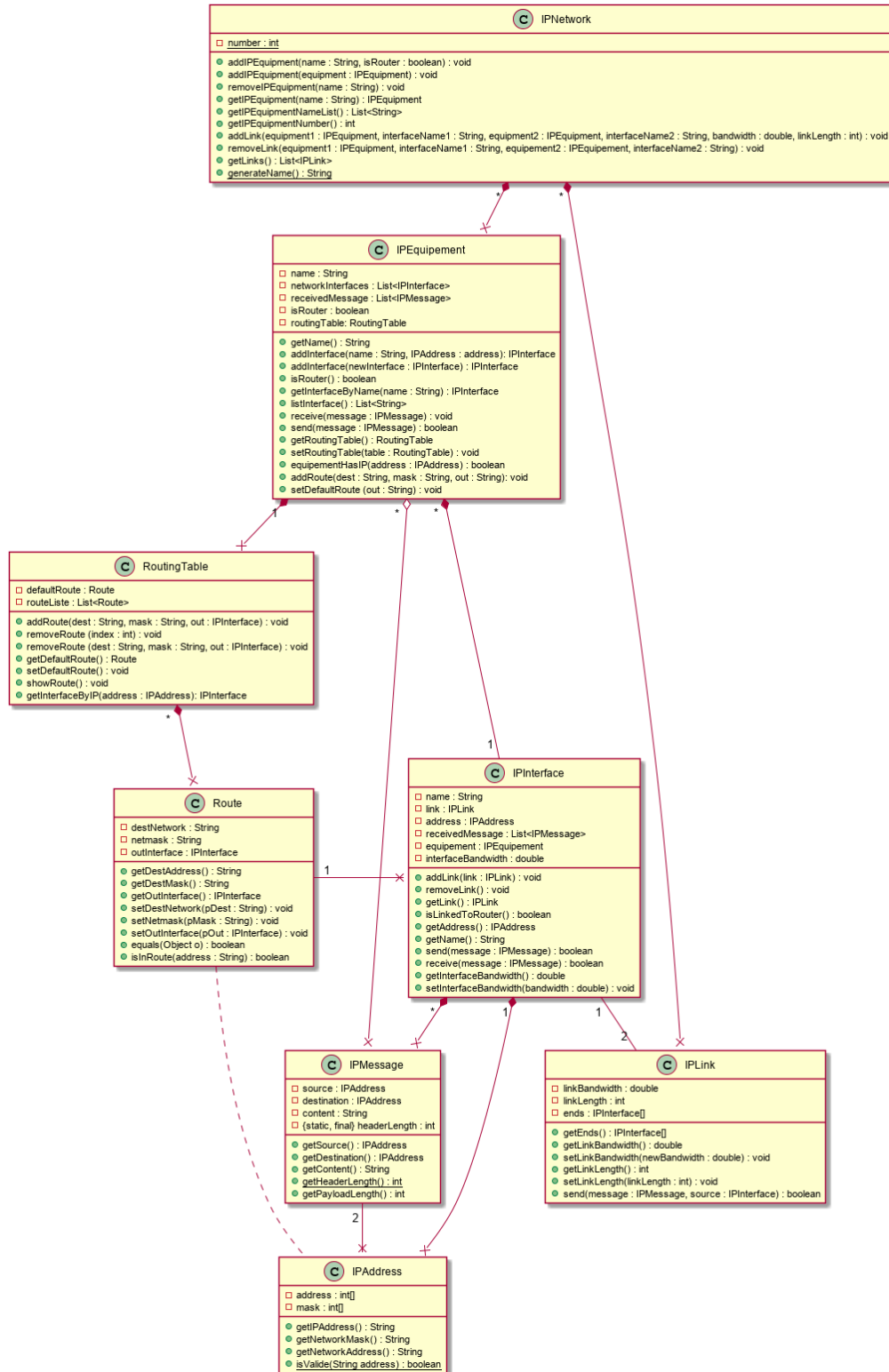


Figure 9: Diagramme de classe du modèle

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 rendu compte que cette conception était bancable. 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.

III. Conception de la vue

A. Diagramme de cas d'utilisation

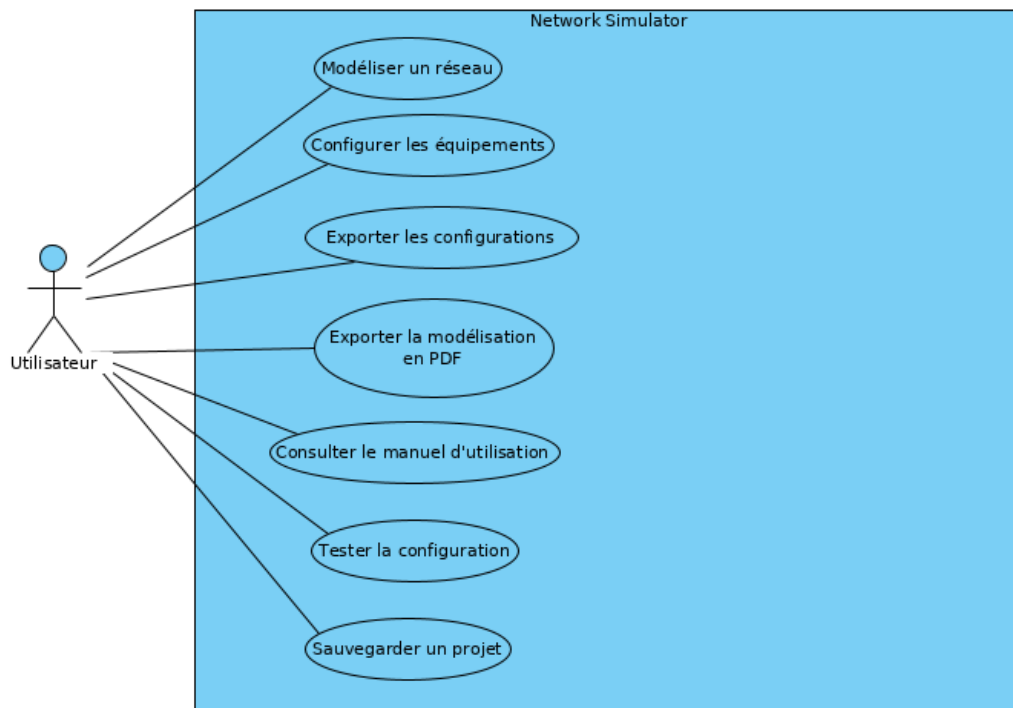


Figure 10: Diagramme de cas d'utilisation

Ce diagramme de cas d'utilisation représente les fonctionnalités disponibles dans la version actuelle de l'application. Après l'itération n°3, l'utilisateur peut maintenant tester sa configuration, les exporter dans des fichiers de commandes Bash, sauvegarder ses projets...

Pour configurer et tester, l'utilisateur a également la possibilité d'utiliser un terminal de commandes.

Enfin, cette version du diagramme de cas d'utilisation représente les fonctionnalités de l'application finale.

B. Diagramme de composant

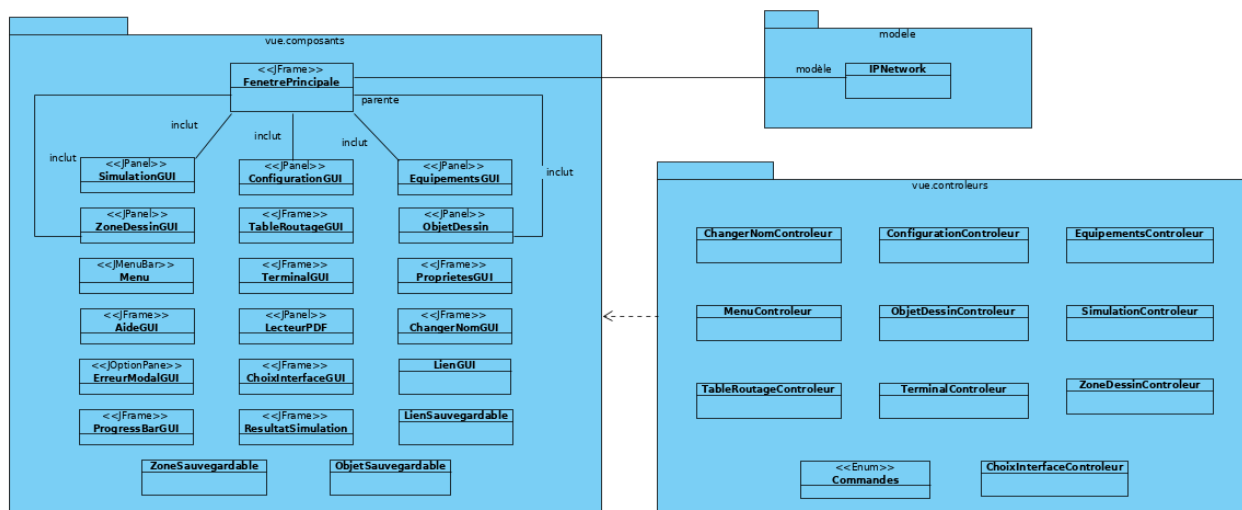


Figure 11: Diagramme de composant de la vue

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 premier sprint. En effet, la grande majorité des composants graphiques a été développée.

Durant le second 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 troisième sprint, nous avons travaillé sur de l'intégration et sur le développement des contrôleurs. Nous avons ajouté la possibilité de tester la configuration réalisée par l'utilisateur ainsi que la configuration des équipements à l'aide du terminal. Enfin nous avons ajouté la sauvegarde du projet et l'export des configurations dans des fichiers au format Bash.

IV. Conclusion

A. Bilan du 1^{er} 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.

B. Bilan du 2nd sprint

Le second sprint nous a permis d'améliorer la base réalisée lors du premier sprint et de commencer à intégrer les différents 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.

C. Bilan du 3^{ème} sprint

Ce troisième et dernier sprint nous a permis de finir l'intégration entre le modèle et la vue. Nous avons également ajouté des nouvelles fonctionnalités comme la sauvegarde d'un projet, l'export des configurations dans des fichiers de commandes Bash et un terminal de commandes pour configurer et tester le réseau.

D. Evolutions possibles

Actuellement plusieurs améliorations sont possibles :

- Au niveau modèle, on pourrait implémenter différents niveaux réseau comme la couche MAC ...
- On pourrait également envisager la gestion de services comme DNS, DHCP ...
- En ce qui concerne la simulation, on pourrait imaginer une gestion de charge des équipements. Et également des communications parallèles.
- Au niveau graphique, une amélioration sur certains composants serait également envisageable.
- Enfin, un modèle générique où l'on pourrait implémenter tout type de réseau.

V. Liste des illustrations

Figure 1: Aperçu de l'onglet configurer.....	2
Figure 2: Aperçu de la vue sans onglet.....	3
Figure 3: Aperçu de l'onglet émettre	3
Figure 4: Aperçu du terminal.....	4
Figure 5: Aperçu de la configuration générale du réseau	4
Figure 6: Aperçu du Trello	5
Figure 7: Aperçu de GanttProject.....	5
Figure 8: Aperçu de GitKraken	6
Figure 9: Diagramme de classe du modèle	7
Figure 10: Diagramme de cas d'utilisation	9
Figure 11: Diagramme de composant de la vue.....	10