SAE 3.01 Réaliser un développement d’application : RAPPORT FINAL

PIERROT Nathan

PINOT Gaëtan
S3A

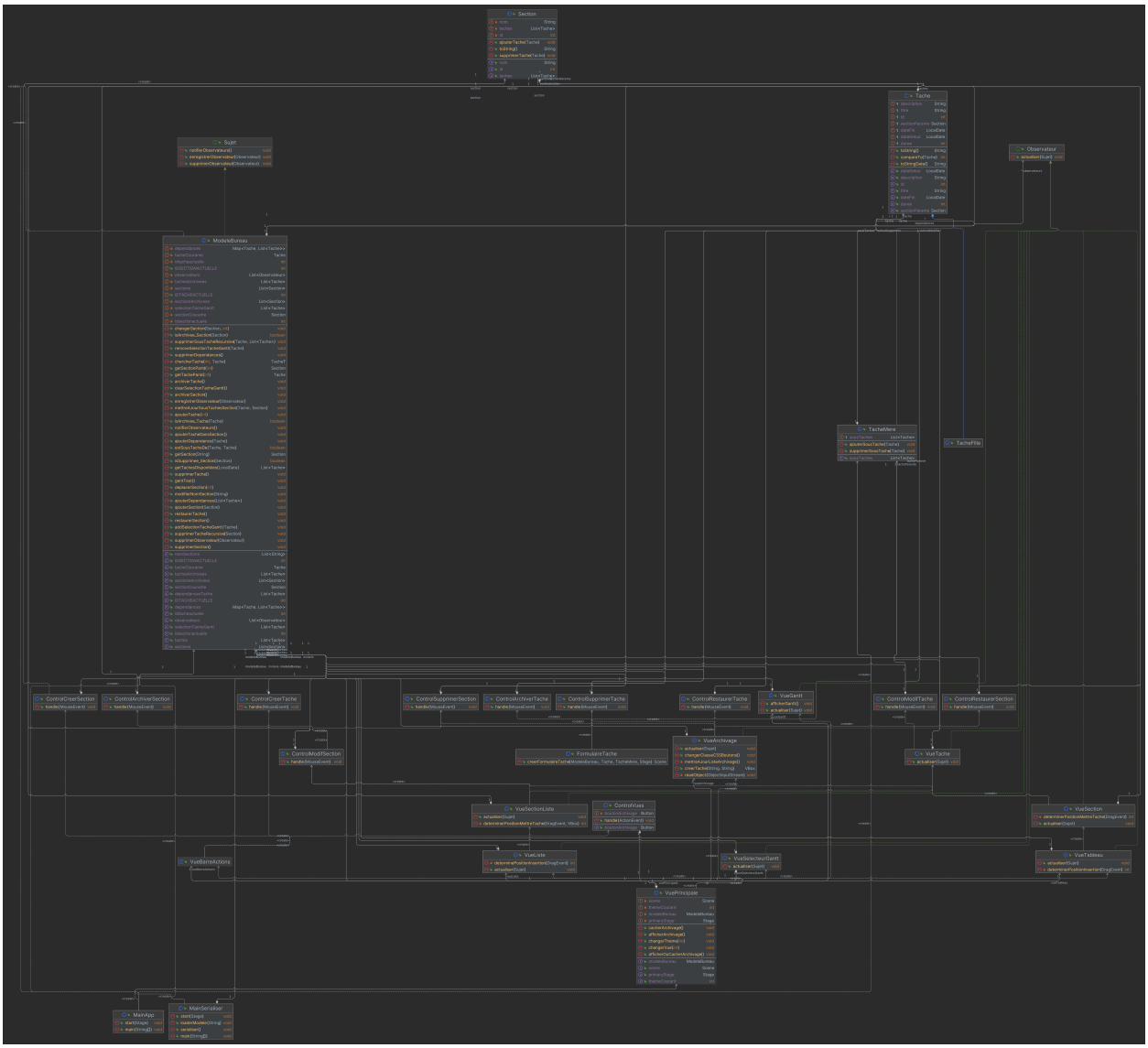
TROHA Stanislas

Fonctionnalités implémentées

- Créer une tâche
- Créer une section
- Supprimer une tâche
- Supprimer une section
- Modifier une section
- Modifier une tâche
- Drag and drop :
  - Déplacer une tâche d’une section à l’autre (et prend la position de l’endroit où elle est déposée)
  - Déplacer une tâche dans une autre pour qu’elle devienne sous-tâche
  - Déplacer une sous-tâche d’une tâche pour qu’elle ne soit plus sous-tâche mais tâche
  - Déplacer une section pour la changer de place
- Sérialisation des données de l’application
- Archivage des tâches et sections

- Restauration des tâches et sections
- Style de l'application (CSS)
- Possibilité de choisir entre plusieurs thèmes
- Affichage Tableau
- Affichage Liste
- Affichage Gantt (sélection de tâches)
- Affichage archivage (sections et tâches archivées)
- Changer de vue

Diagramme de classe final :



Le diagramme étant très volumineux il est disponible dans ce même dossier que le bilan final en tant que diagrammeClassesFinal

Répartition du travail entre les étudiants :

- Conception : PIERROT – PINOT – TROHA
- Implémentation de la structure du projet : PINOT – TROHA
- Structure de l’application graphique : TROHA

- Développement du Model : PIERROT – PINOT – TROHA
- Test Unitaires : PIERROT – PINOT
- JavaFX : TROHA
- Vue Tableau : TROHA
- Vue Liste : TROHA
- Vue Gantt : PINOT
- Vue Archivage : PIERROT - TROHA
- Style de l’application : PIERROT
- Drag & Drop : TROHA

Présentation d’un élément orignal dont vous êtes fiers :

PIERROT : Le choix du thème de l’application et les différents thèmes

J’ai réalisé entièrement le style de l’application et nous nous sommes dit que de permettre à l’utilisateur de changer de thème serait une bonne idée. J’ai alors programmé un nouveau bouton en haut à droite de l’application qui permet de changer le thème de l’application et qui laisse le choix entre 6 thèmes différents (Brouillard : nuances de gris, Océan : nuances de bleu clair, Crépuscule : nuances de rouge/orange, Foret : nuances de vert, Nuit : nuances de bleu foncé, Plage : nuances de jaune), tous plus beaux les uns que les autres. La difficulté à été de choisir les bonnes palettes de couleur pour garder une ambiance agréable. Et ensuite il a fallu remplacer toutes les couleurs selon la nouvelle palette et arranger certains éléments qui ne correspondaient pas bien au thème.

PINOT : La vue Gantt :

Il fallait d’abord afficher un menu de sélection des tâches que l’utilisateur veut afficher, puis afficher les tâches sélectionnées à partir du jour où elle commence jusqu’au jour où elle se termine, le principe de gantt c’est aussi d’afficher les dépendances chronologiques entre les tâches, ce qu’on fait ici avec des lignes. J’ai aussi décidé d’afficher une colonne sur deux contrasté pour qu’on puisse facilement savoir quelle tâche est dans quelle date.

TROHA : Le Drag and Drop « libre »

Le Drag and Drop libre signifie qu’on ne peut pas seulement déplacer les tâches entre les sections mais on peut absolument drag n’importe quelle tâche, n’importe quelle section, et l’endroit où on la dépose est pris en compte, la tâche/section ne s’ajoute pas tout simplement à la fin du conteneur dans lequel elle a été déposée.

J’ai donc implémenté la fonctionnalité de pouvoir déplacer les sections en plus des tâches. Mais lors du Drag and Drop, que ce soit les tâches ou les sections la position où la tâche/section est déposée est traitée par l’application, et la tâche/section déposée à telle position s’y retrouve bien.

Par exemple si on a une section 1 avec une Tache1 Tache2 et Tache3. Si je décide de glisser déposer la tâche 3 entre la tâche1 et la tâche2, à l’affichage elle sera bien disposée entre la tâche1 et tâche2. (Également stocké dans cet ordre dans le modèle). J’ai trouvé qu’ajouter cette fonctionnalité de pouvoir vraiment glisser-déposer librement apportait beaucoup plus de liberté à l’utilisateur, sans compter le fait que c’est bien plus facile de glisser déposer une tâche que de devoir manuellement choisir parmi une liste de section dans quelle section déplacer une tâche.

Idem si on veut ré-arranger ses sections comme sur Trello (avec le principe de A faire – En cours – Terminé)

Les éléments qui ont été modifiés par rapport à l'étude préalable :

- Le diagramme de classe, il est énormément plus important à la fin du développement de l’application. Ce n’était pas possible pour nous de trouver un pareil diagramme. L’interface graphique (JavaFX) a rajouté beaucoup plus de

classe que ce que nous pensions au départ.
Les vues et contrôleurs se sont décuplés par rapport à notre étude. A noter qu'avant le commencement du développement de l'interface graphique, nous n'avions pas assez bien cerné comment allaient fonctionner les contrôleurs et vues, et comment ils allaient être en relation avec le modèle.

- Au niveau des dépendances chronologiques ou de la notion de tâches et sous-tâches, nous pensions pouvoir ajouter des dépendances chronologiques même entre les sous-tâches lors de la conception et de l'analyse du sujet, mais en développant le modèle et en se rendant compte d'à quel point la gestion des dates et dépendances était minutieuse, nous avons fini par décider que les sous-tâches ne seraient que des tâches en composants d'autres, sans aucun données relatives aux dates ni aux dépendances chronologiques.
-

Les éléments qui n'ont pas été modifiés par rapport à l'étude préalable :

- Nous avons su correctement suivre notre programme par itération. Dans l'ensemble nous avons respecté de que nous avions prévu : commencer uniquement par du développement backend (le model), ensuite nous avons ajouté le début de la vue Tableau avec Java FX, puis nous avons mis en place le maximum de fonctionnalités (création, modification, suppression, archivage, restauration des tâches et des section) sur la vue Tableau avant de commencer les autres. Nous avons alors pu implémenter le drag & drop et ajouter du style à notre application. Avant de continuer avec la vue Gantt, la vue Liste et l'ajout de la possibilité pour l'utilisateur de changer le thème de l'application. Nous avons terminé par corriger tous les bugs qu'il pouvait y avoir et ajouter tous les commentaires nécessaires à la bonne compréhension de notre code.
- Autre point où nous avons réussi à nous y tenir est la disposition des éléments sur notre application. Nous avons bien respecté les maquettes que nous avions réalisées lors de l'étude préalable.

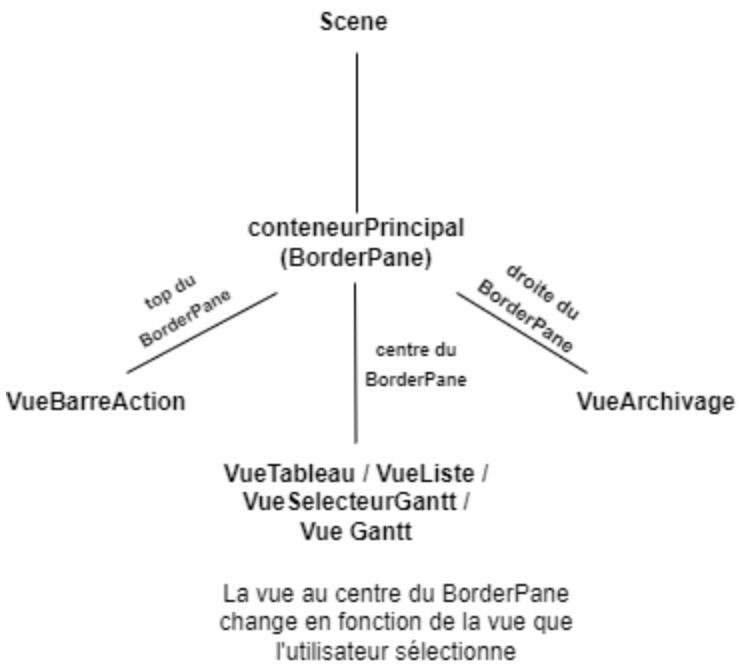
Les patrons de conception et d'architecture mis en œuvre dans le programme :

- Patron MVC : Notre application est graphique nous avons donc choisi d’utiliser MVC, cela nous semblait le plus cohérent
- Patron Composite : L’application permet de donner des sous tâches à des tâches, nous avons donc pensé à utiliser le patron Composite
- Patrons Singleton : lors de la programmation de l’application nous avons rencontré des problèmes avec la scène, nous avons alors décidé d’implémenter le patron Singleton afin de ne pas avoir de problème de duplication de la Scene.

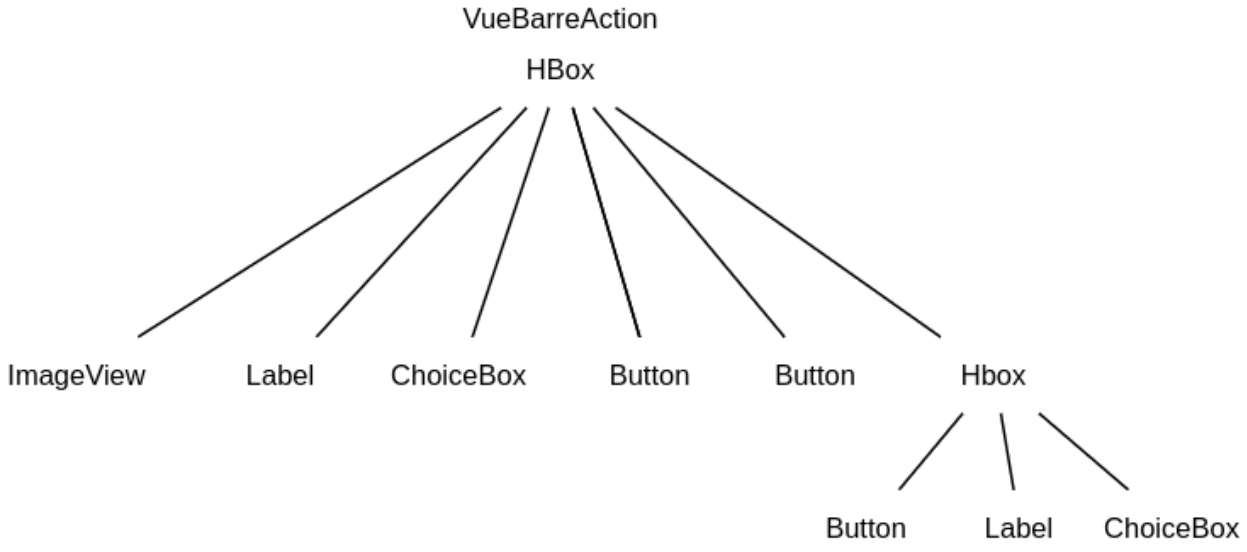
Le graphe de scène de l'interface graphique :

On retrouve les graphes dans le dossier GrapheScene/
Nous avons décomposé le graphe en plusieurs graphes pour une question de visibilité.

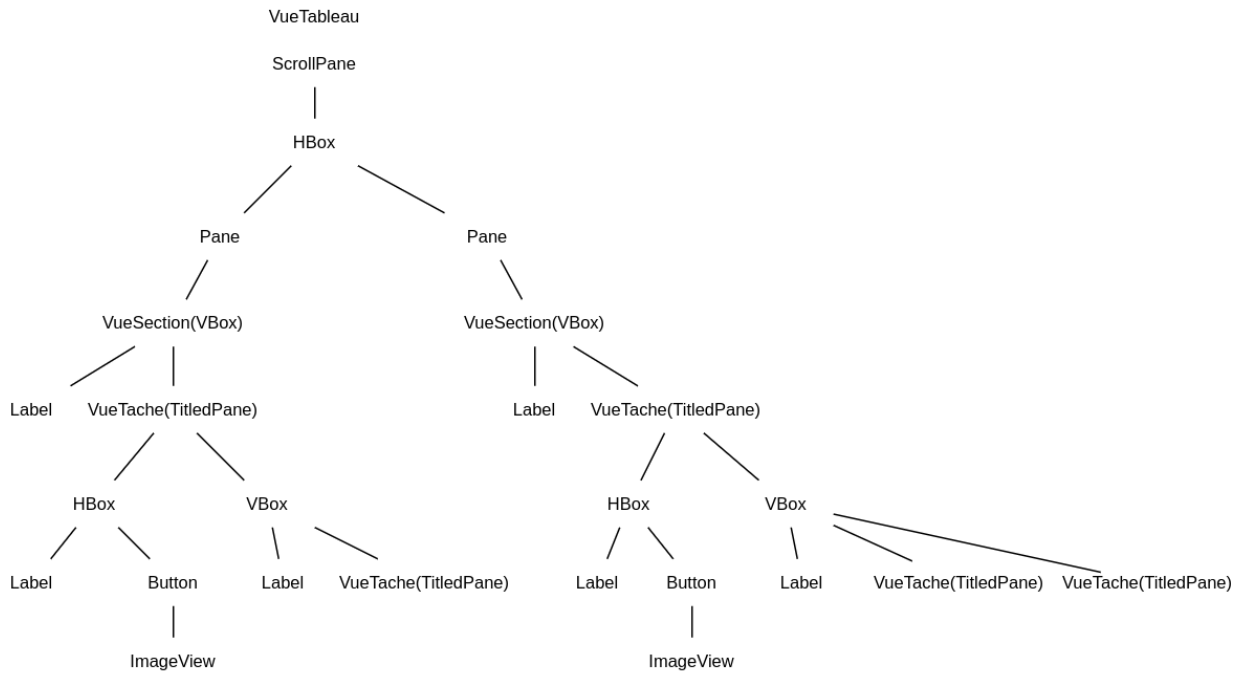
Tout d’abord voici le graphe principal de la scène de l’interface graphique :



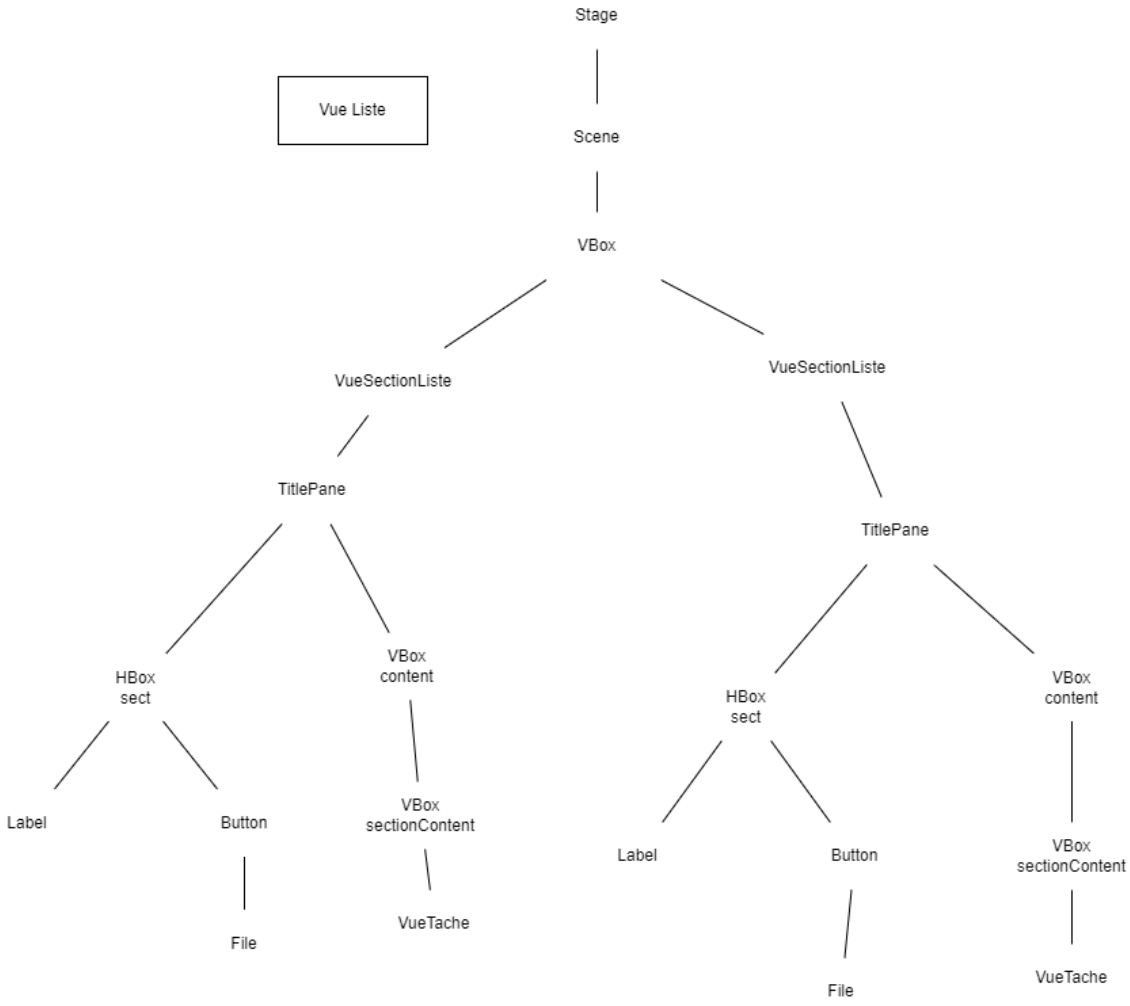
Graphe de la VueBarreAction :



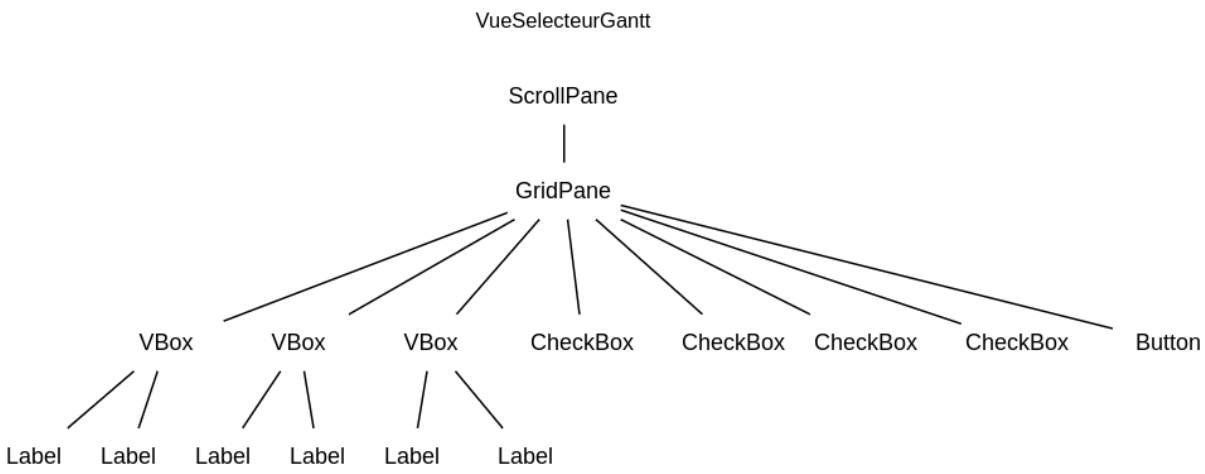
Graphe de la VueTableau :



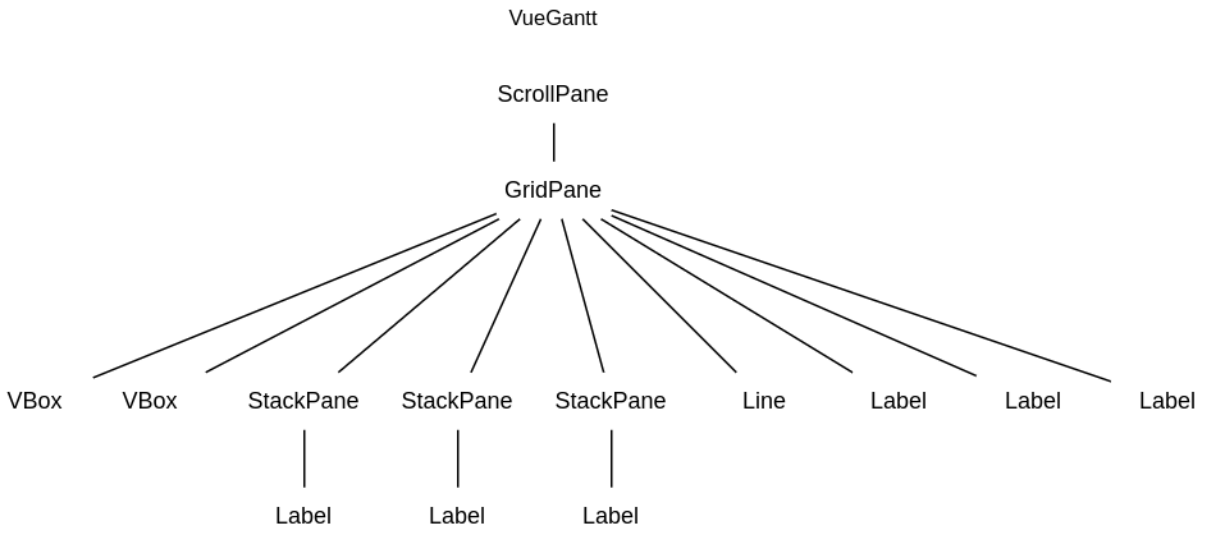
Graphe de la VueListe :



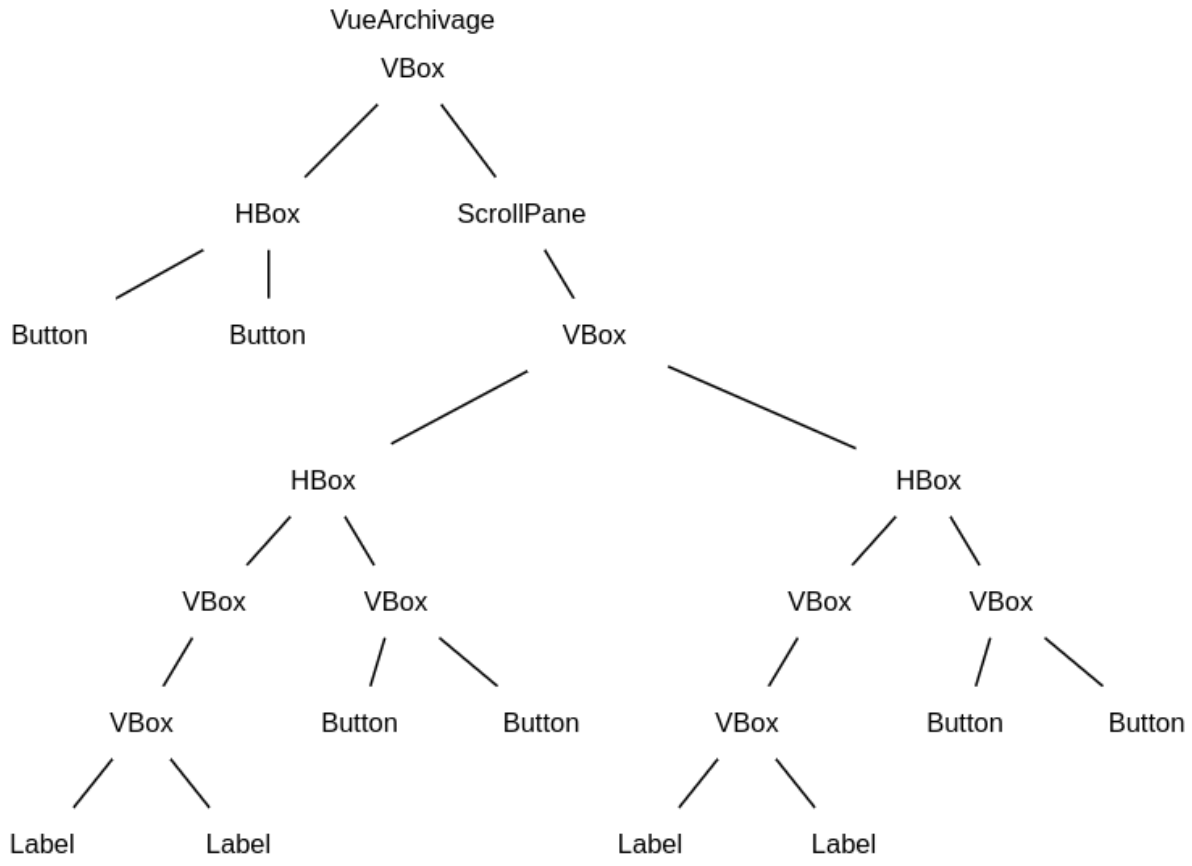
Graphe de la VueSelecteurGantt :



Graphe de la VueGantt :



Graphe de la VueArchivage :



Organisation graphique des différentes vues :

Pour chaque Vue d'affichage de sections et tâches, on crée d'abord la vue associée (ScrollPane pour VueTableau, VueListe).
Dans le cas d'une VueTableau, on va créer la VueTableau (scrollpane) et on va créer dans cette VueTableau un conteneur qui contiendra les sections.
Une vue principale (tableau, liste..) contient le modèle en attribut, ainsi on peut parcourir la liste de sections du modèle et créer une VueSection pour chaque Section.
Ce processus sera similaire pour les tâches et sous-tâches
Les sections sont quant à elles représentée par des VueSection, qui prennent en attribut le modèle (car on a besoin du modèle pour utiliser ses méthodes pour le drag and drop par exemple) et la section concernée.
Ensuite, dans le cas de la VueTableau on crée une HBox qui contiendra le titre de la section puis les tâches qu'elle comporte. Et lorsqu'on affiche les tâches que la section

comporte, on utilise la classe VueTache, pour créer une VueTache pour chaque tâche que la section comporte directement dans sa liste de tâches.
Dans la classe VueTache (représentée par un TItledPane dans VueTableau et VueListe), on vérifie ensuite si la tâche n'a pas de sous-tâche, si elle en a, alors on recrée des VueTache et ce jusqu'à tomber sur une sous-tâche qui n'a plus de sous-tâche elle-même.
Au niveau de la représentation graphique, une vue d'affichage est représentée par un ScrollPane. Dans la classe qui étend le ScrollPane, le parcours des sections est effectuée pour créer à son tour des VueSection, et dans les VueSection, le parcourt de la liste des tâches de la section y étant associée est fait pour créer une VueTache pour chaque tâche, et ce récursivement grâce aux sous-tâches que peuvent comporter une tâche.

C'est un peu différent pour la vueArchivage, qui elle ne met pas en place le principe d'afficher les sous-tâches archivées (car impossible d'archiver une sous-tâche).
Elle crée directement des éléments simples graphiques sous forme de VBox pour représenter une tâche ou bien une section archivée.

Pour la Vue Gantt c'est différent, les sous-tâches ne sont pas non plus prises en compte pour l'affichage.

Comment le changement de vue est réalisé :

Comme le montre le graphe de scène, la vue principale contient un BorderPane, qui lui contient, en son centre la vue actuelle (tableau, liste, gantt).

C'est donc grâce aux méthodes dans cette classe que le changement de vue est effectué. Il suffit d'appeler la méthode changerVue() en passant en paramètre de cette méthode la constante correspondant à la vue qu'on veut mettre (TABLEAU, LISTE, SELECTEUR GANTT, GANTT) et la classe changera l'élément graphique qu'elle contient au centre de son BorderPane.

La seule exception est pour la VueArchivage, étant donné qu'elle est positionnée (lorsqu'elle est visible) dans la section droite du BorderPane, ainsi, lorsqu'on clique pour la première fois sur le bouton « Archivage » de la barre d'actions en haut de l'application, elle est ajoutée à droite du BorderPane, et lorsqu'on reclique sur le bouton, alors elle en est enlevée. Car l'objectif de cette VueArchivage était d'apparaître sur

l’application en plus de la vue choisie par l’utilsateur pour visualiser les tâches et sections.