Rapport de l'itération 3:

Le but de cette itération est que l'utilisateur puisse choisir son affichage soit bureau ou liste. Le visuel a aussi été fait avec du css.

Pour le diagramme de classe les classes qui sont coloriés en vert sont celles qu'on a faites durant l'itération 3 (ou modifié).

Pour lancer notre application il faut lancer le fichier PrincipalFx.java

Fonctionnalités:

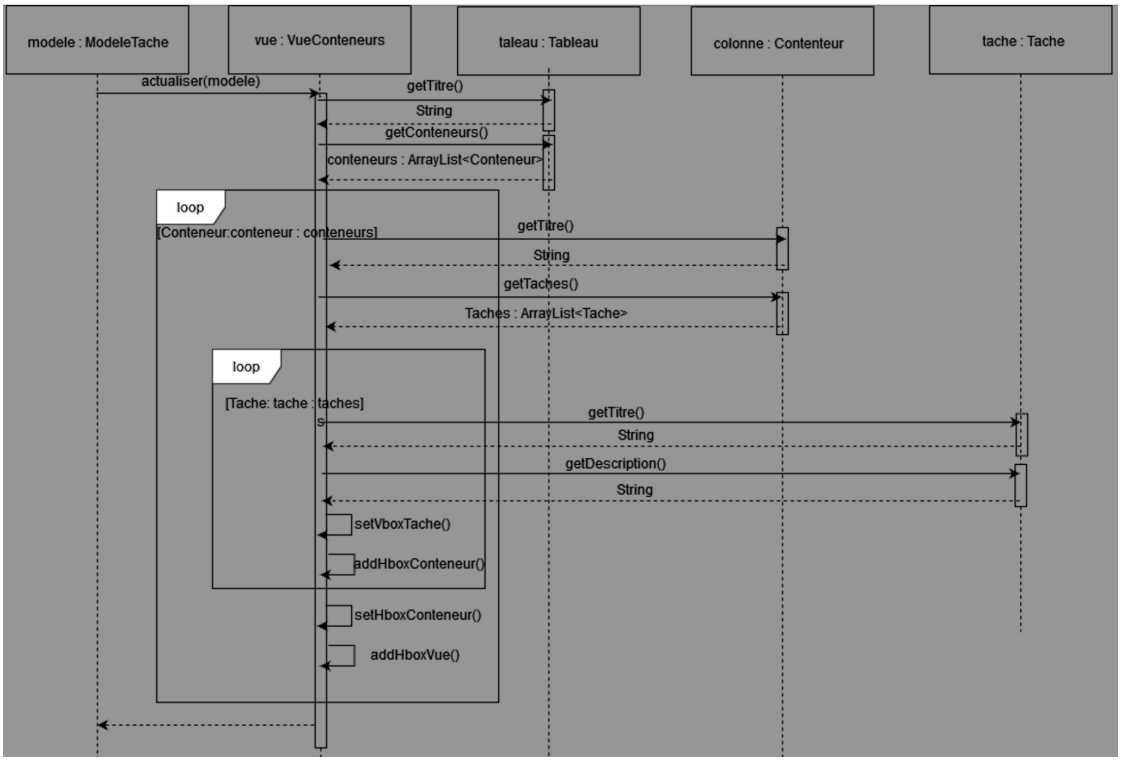
Affichage Bureau:

L'affichage bureau est l'affichage classique avec différentes colonnes qui sont mises à côté les unes des autres. Les tâches sont directement affichées à l'intérieur de chaque une et l'utilisateur peut interagir directement avec elles. **Fait par Hugo Rysak**

Affichage en liste:

L'affichage en liste est une autre façon de voir les listes/colonnes de la vue bureau, pour que l'utilisateur puisse voir les différentes tâches qui sont liées aux listes/colonnes, il suffit qu'il clique sur la liste et les tâches apparaîtront sous forme de menu déroulant. **Fait par Maxime Bechard**

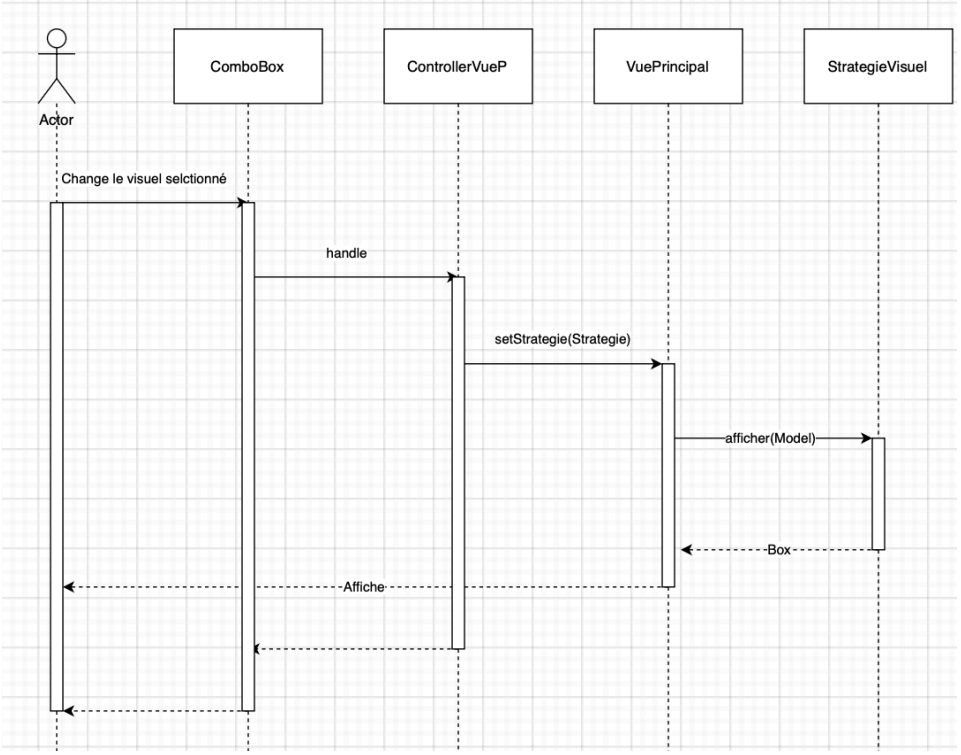
Diagramme de séquence liée à la fonctionnalité:



Changer d’affichage:

L’utilisateur peut changer de système d’affichage de façon très simple en sélectionnant le mode d’affichage qu’il désire avec un petit menu déroulant en haut à droite de l’application. **Fait par Timothée Benchergui et Jawad Komodzinski**

Diagramme de séquence liée à la fonctionnalité:



<u>Toutes les fonctionnalités ci-dessus fonctionnent et ont été testé.</u>

<u>Visuel de l’application:</u>

Pour le visuel choisi, nous sommes parties sur un style plutôt violet que nous trouvions agréable pour les yeux en plus d’être passe partout, nous sommes rester sur un même style pour toutes les pages pour ne pas agresser visuellement l’utilisateur. Pour les boutons afficher tâche, nous avons décidé de mettre des logo que nous trouvions plus jolie et en accord avec notre visuel.

<u>Quelque information sur le code :</u>

Pour la partie code,nous avons ajouté à notre vue principal un attribut affichage qui est un attribut de type StratégieVisuel. Cette classe (qui hérite d’un pane) a pour

l’instant deux classes enfant : VisuelBureau et VisuelListe chacun réalisant le visuel accordé à son nom.

Nous avons décidé de partir sur un patron de type Stratégie pour les raison suivantes:
-Si plus tard nous voulons faire d’autre affichage il est facile d’en ajouter un nouveau, juste faire une classe qui hérite de stratégieVisuel
-Le code est beaucoup plus maintenable comme ceci
-Nous avons décidé de faire de stratégie un Pane plutôt que de lui faire renvoyer un affichage avec une méthode car quand l’affichage est mis à jour, il suffit de mettre la stratégie à jour (Le reste de la vuePrincipal n’est jamais modifié) c’est donc plus économe (en mémoire) que de tout détruire et de tout redessiner. Avec cette méthode nous pouvons aussi changer directement ce qui a était modifié par exemple une tâche rajouté au lieu de tout redessiner nous allons juste redessiner la tâche elle même, chose que l’on ne pourrait pas faire si stratégie renvoyait un affichage au lieu d’être un affichage lui même.
Pour changer l’attribut de type strategieVisuel, nous avons fait un controller lié à la comboBox qui va changer directement dans VuePrincipal l’attribut Stratégie et l’affichage en conséquence.