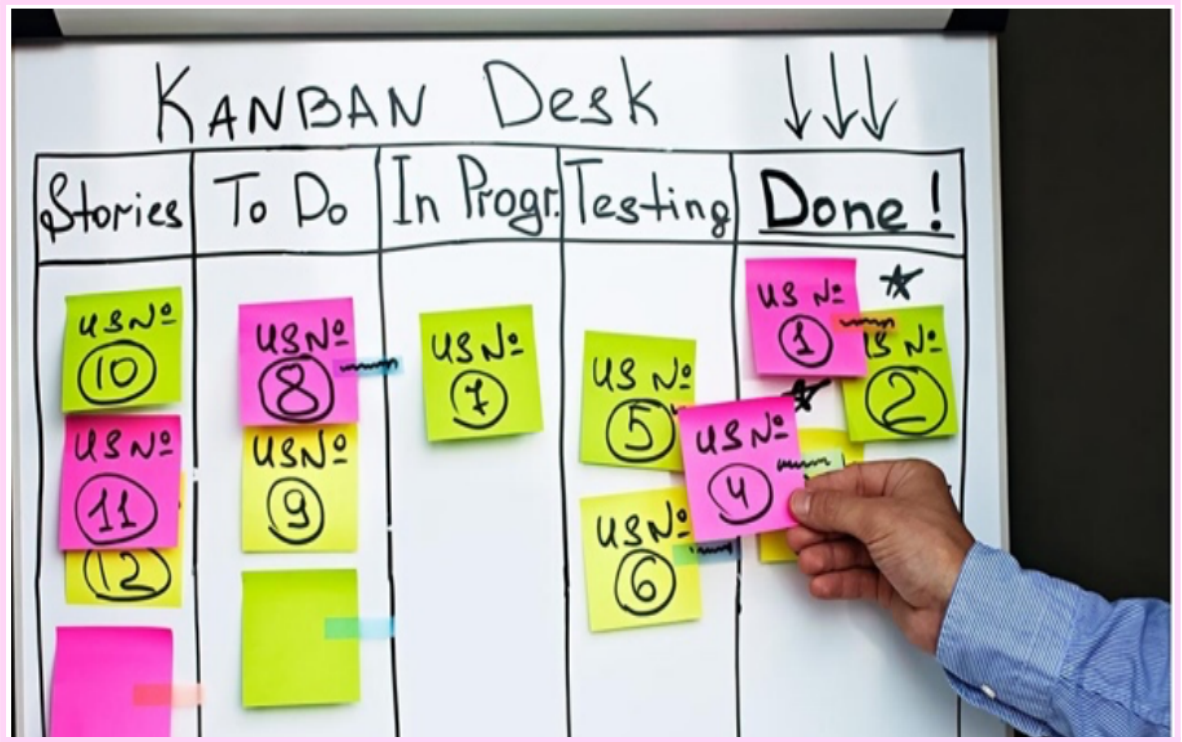
Arcelin Nino	Bitsindou Maëlle	Ponroy Célie	Prykhodko Yehor

SAE 3.01 – Développement d'une application

Etude préalable
Du logiciel d'organisation de tâches personnelles



Arcelin Nino	Bitsindou Maëlle	Ponroy Célie	Prykhodko Yehor

SOMMAIRE

Arcelin Nino	Bitsindou Maëlle	Ponroy Célie	Prykhodko Yehor

I. Introduction

Cette étude préalable nous permet d'analyser et de développer au mieux les fonctionnalités de notre application de gestion de tâches personnelles.

Cette phase initiale va explorer les fonctionnalités envisagées, les différents cas d'utilisation et les acteurs impliqués. Des scénarios pertinents que nous aurons choisis seront analysés pour mieux appréhender les interactions de l'utilisateur avec le système. En utilisant de multiples diagrammes et de patrons de conception, cette étude va définir précisément les contours de l'application et à anticiper les étapes de son développement.

Arcelin Nino	Bitsindou Maëlle	Ponroy Célie	Prykhodko Yehor

II. La liste de fonctionnalités

Voici la liste des fonctionnalités que nous envisageons de programmer dans pour notre application de gestion de tâches:

1. Gérer une tâche
   → supprimer une tâche
   → ajouter une tâche,
   → modifier le nom d'une tâche
   → ajouter la durée d'une tâche

2. Gérer une colonne
   → supprimer une colonne
   → ajouter une colonne
   → modifier le nom d'une colonne

3. Gérer un tableau
   → supprimer un tableau
   → modifier le nom d'un tableau

4. Gérer les dépendances d'une tâche
   → attribuer les antériorités

5. Générer le diagramme de Gantt
   → prendre en compte la durée des tâches et leurs antériorités

6. Afficher le tableau en liste
   → gérer la vue sous forme de liste

7. Afficher le tableau en bureau
   → gérer la vue sous forme de liste

8. Gérer la liste des tâches archivées
   → gérer la liste d'archives

Arcelin Nino	Bitsindou Maëlle	Ponroy Célie	Prykhodko Yehor

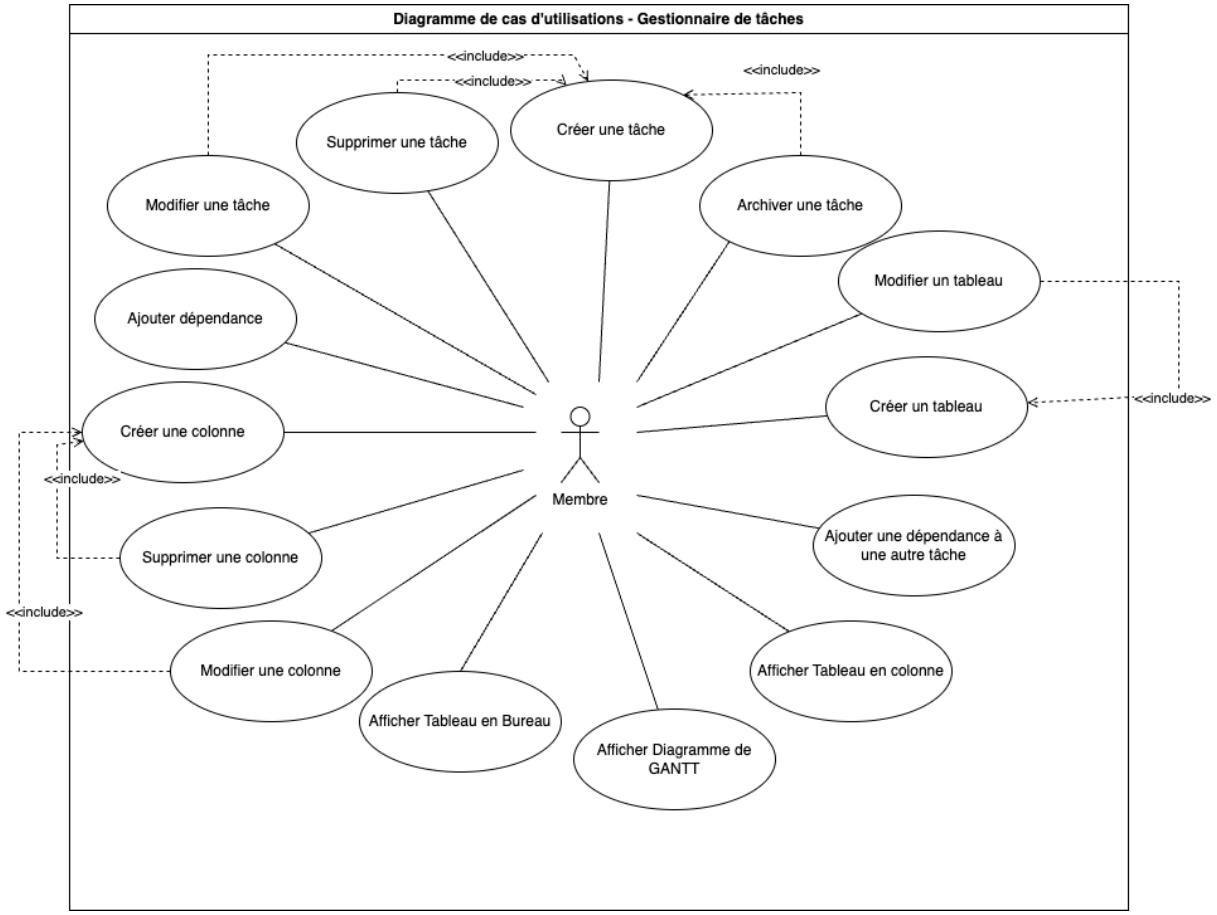
III. Les cas d'utilisations et les acteurs

Nous avons d'abord listé tous les cas d'utilisation et les acteurs de notre application.

Acteur	- Membre (personne qui a accès à l'application)
Cas d'utilisation	- Créer un tableau - Créer une colonne - Supprimer colonne - Modifier colonne - Modifier tâche - Modifier colonne - Créer une tâche - Archiver une tâche - Désarchiver une tâche - Déplacer une tâche - Supprimer une tâche - Générer un diagramme de Gantt - Afficher le tableau en liste - Afficher le tableau en bureau - Ajouter dépendance d'une tâche à une autre - Choisir un tableau

- Diagramme de cas d'utilisation

Nous avons ensuite regroupé et organisé tous ces cas d'utilisations qui décrivent le cas pratique. Comme l'application ne sera pas connectée via le réseau, il y aura un seul utilisateur à la fois sur l'application et donc un seul acteur dans ce cas.
Le Membre, utilisateur qui accède à l'application pourra donc effectuer différentes actions dans cette application.



Fait sur Balsamiq

- Les scénarios

Pour analyser au mieux l'élaboration de cette application, nous avons décidé de décrire 2 scénarios que nous trouvions intéressant de développer.

On admet dans ces scénarios, que l'utilisateur est un membre de l'application et qu'un tableau avec des colonnes existe déjà.

Scénario n°1: Ajout d'une tâche
Le membre va pouvoir ajouter une tâche à l'une des colonnes d'un tableau. On suppose que le tableau en question ainsi que ses colonnes sont déjà réglés.

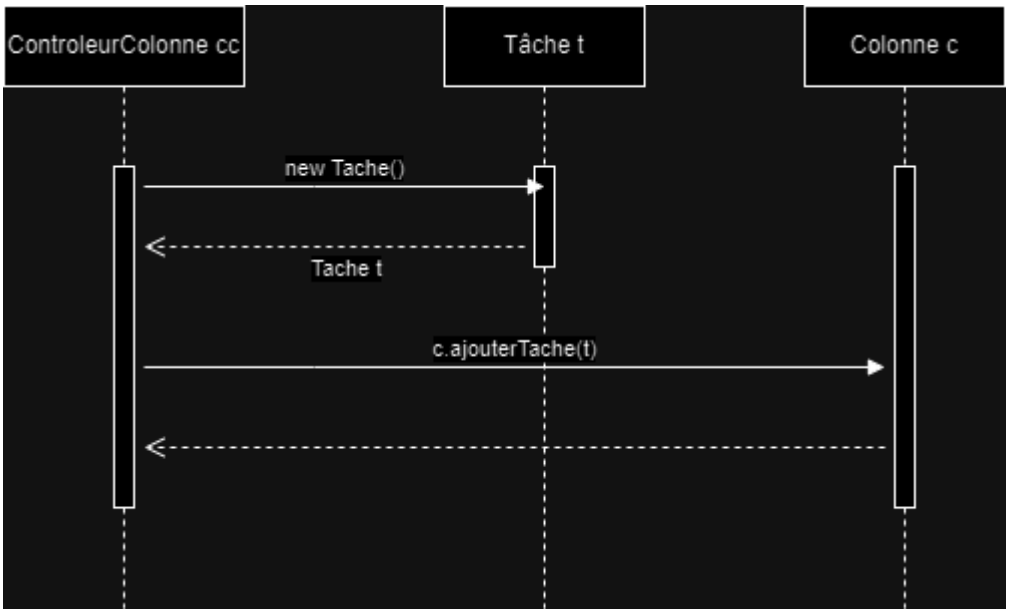
1. Le Membre sélectionne un tableau.
2. Le Membre sélectionne une colonne.
3. Le Membre clique sur un bouton ajouter une tâche.
4. Le Membre entre les infos de la tâche.
5. Le Membre sauvegarde la tâche.
6. Le Système ajoute une tâche.

Scénario n°2: Archivage d'une tâche
Le membre va pouvoir archiver une tâche créée dans la liste des tâches archivées. On suppose que la tâche à archiver existe déjà.

1. Le Membre sélectionne l'un des tableaux.
2. Le Membre sélectionne une colonne de son choix.
3. Le Membre clique sur un bouton pour archiver une tâche.
4. Le Système va archiver la tâche en question dans la liste des tâches archivées.

- Le diagramme de séquence de notre scénario

  Pour venir compléter la description du scénario choisi (l'ajout de la tâche), nous avons décidé de faire un diagramme de séquence.



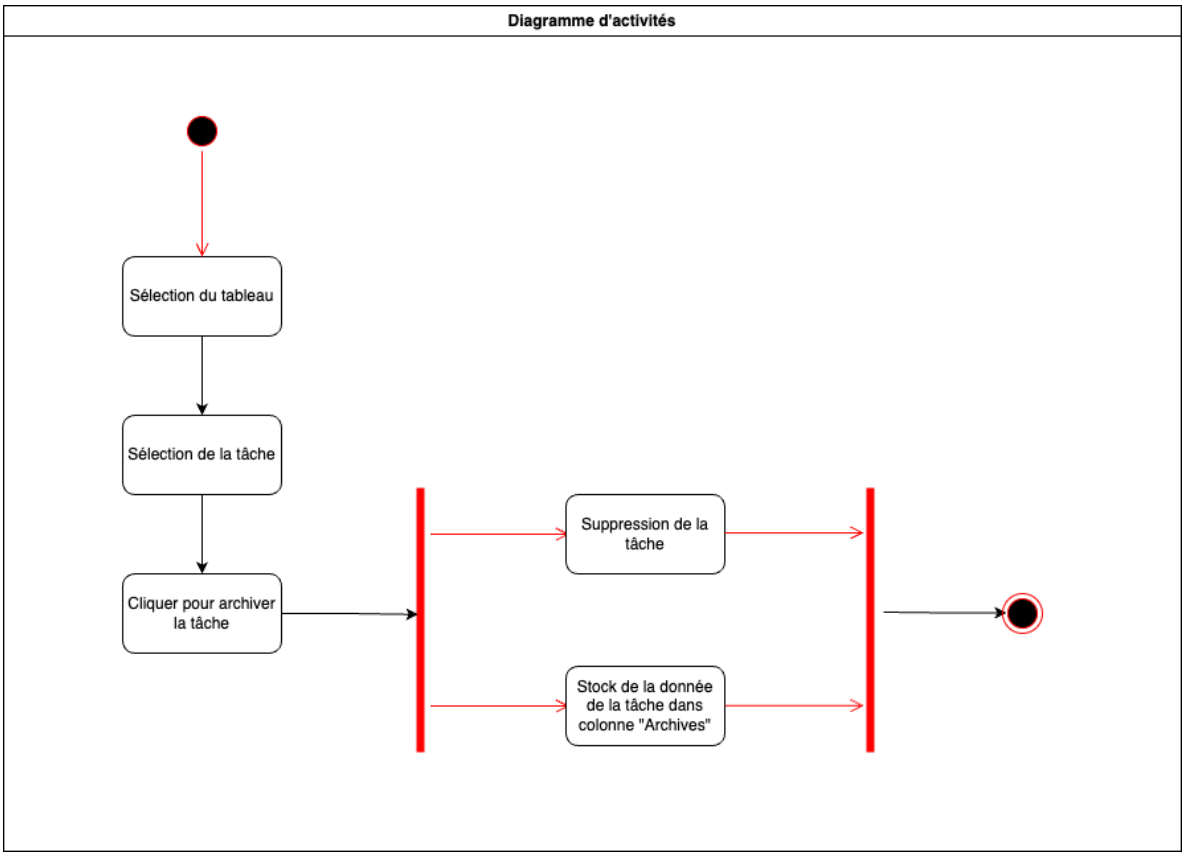
Fait sur Draw.io

Dans ce diagramme, le controller de la colonne va venir créer une nouvelle tâche t grâce à sa méthode handle. Il faut garder à l'esprit qu'une tâche est un composant et que donc une colonne est une liste de composants et que donc pour pouvoir ajouter une tâche à une colonne, on utilise la méthode ajouterComposant qui ajoutera la tâche à la colonne.

IV. Les diagrammes d'activité et le diagramme d'état

- Le diagramme d'activité du scénario "Archiver une tâche"

L'utilisateur sélectionne le tableau dans lequel il souhaite archiver une tâche, il sélectionne ensuite la tâche et clique sur le bouton avec le signe "archiver". Suite à cette action la tâche est supprimée de la colonne dans laquelle la tâche était et parallèlement elle est stockée dans la colonne "Archives".

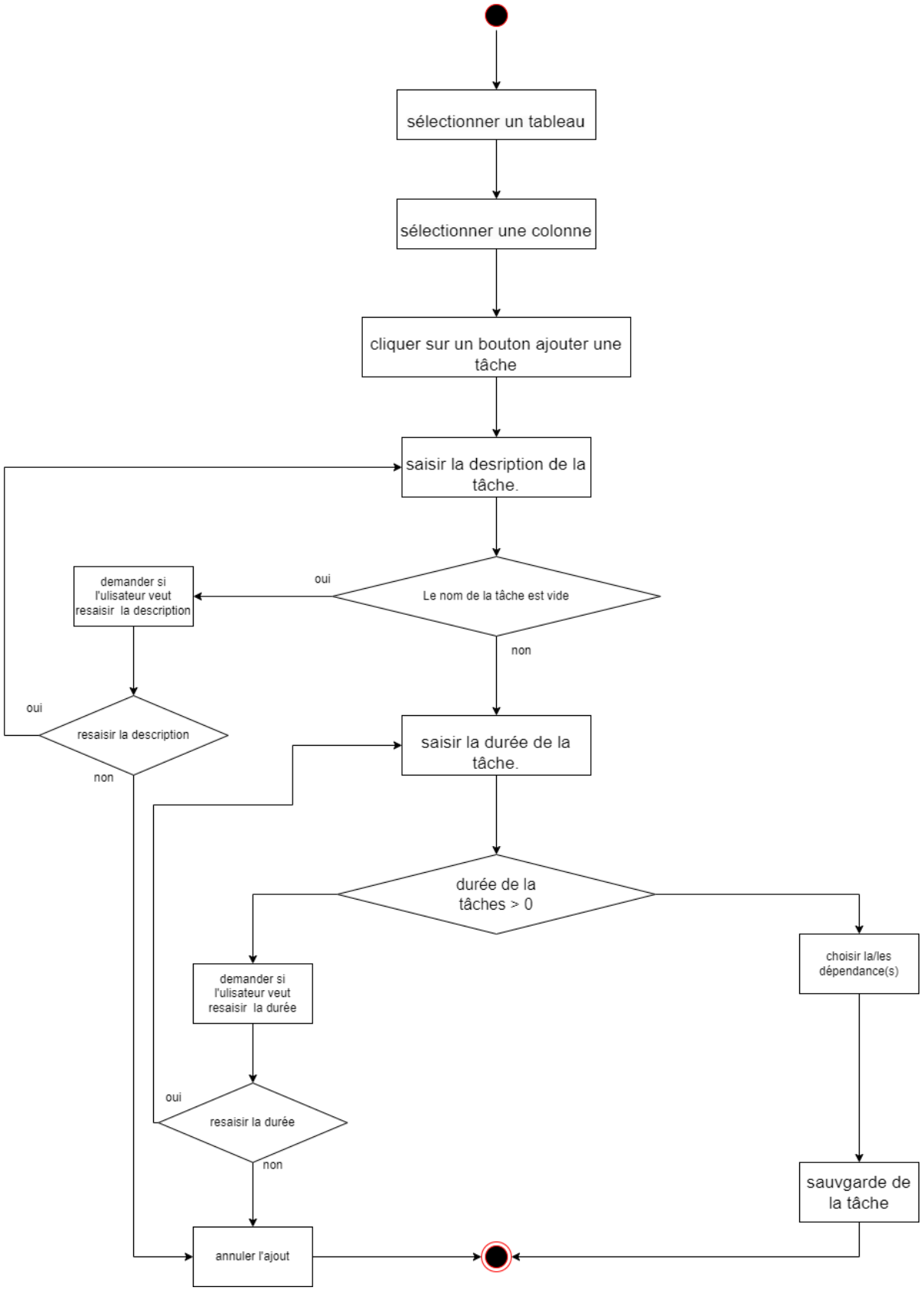


Fait sur Draw.io

- Le diagramme d'activité du scénario "Ajouter une tâche"

L'utilisateur sélectionne le tableau dans lequel il souhaite ajouter une tâche, il clique sur le bouton "Ajouter une tâche". Suite à cette action, la tâche va être créée dans la colonne en question et l'utilisateur va pouvoir choisir la description de celle-ci.

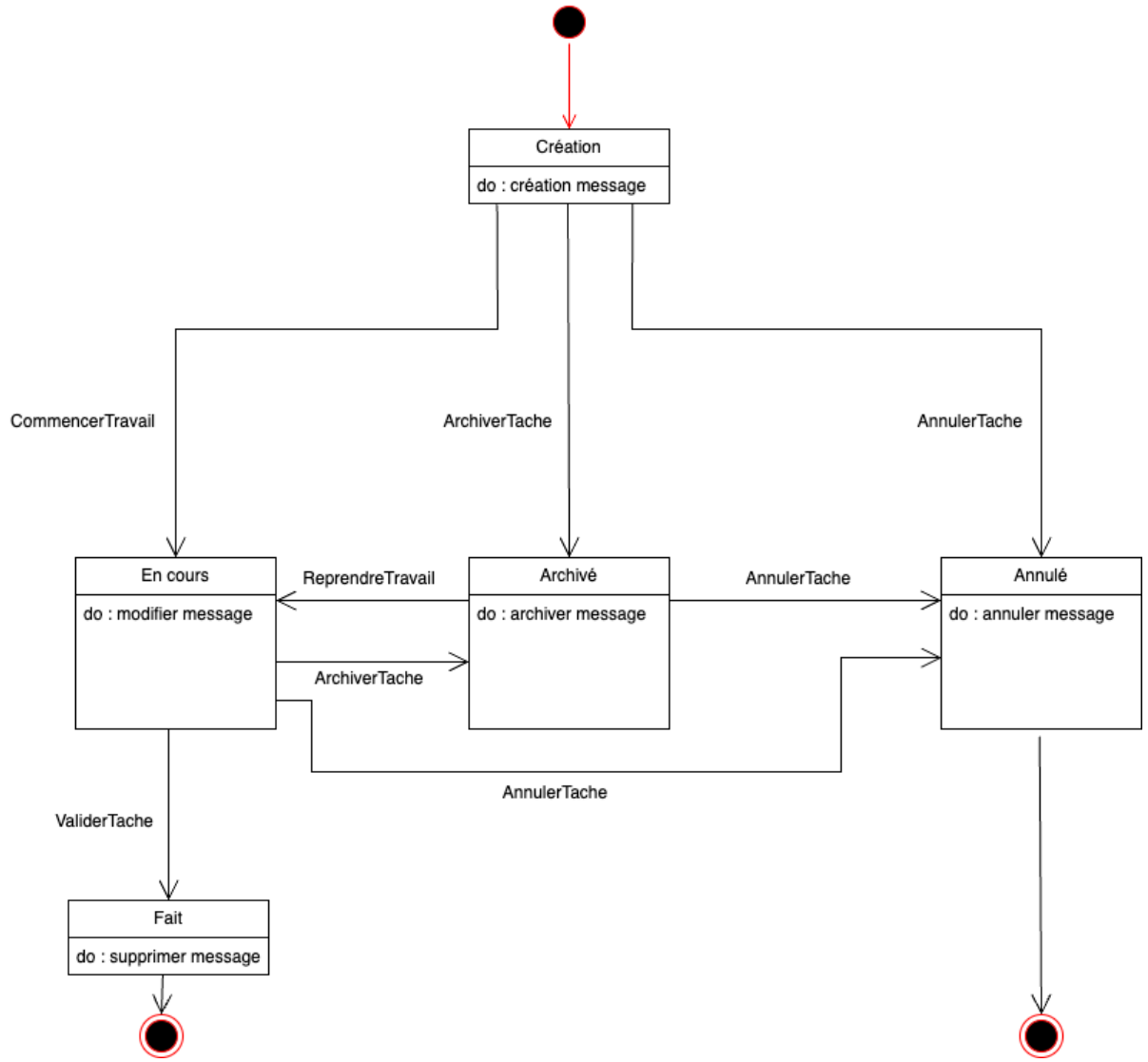
Arcelin Nino	Bitsindou Maëlle	Ponroy Célie	Prykhodko Yehor



Fait sur Draw.io

- Le diagramme d'état

Nous avons ensuite établi un diagramme d'état pour pour comprendre, visualiser et spécifier le comportement du système de l'application.

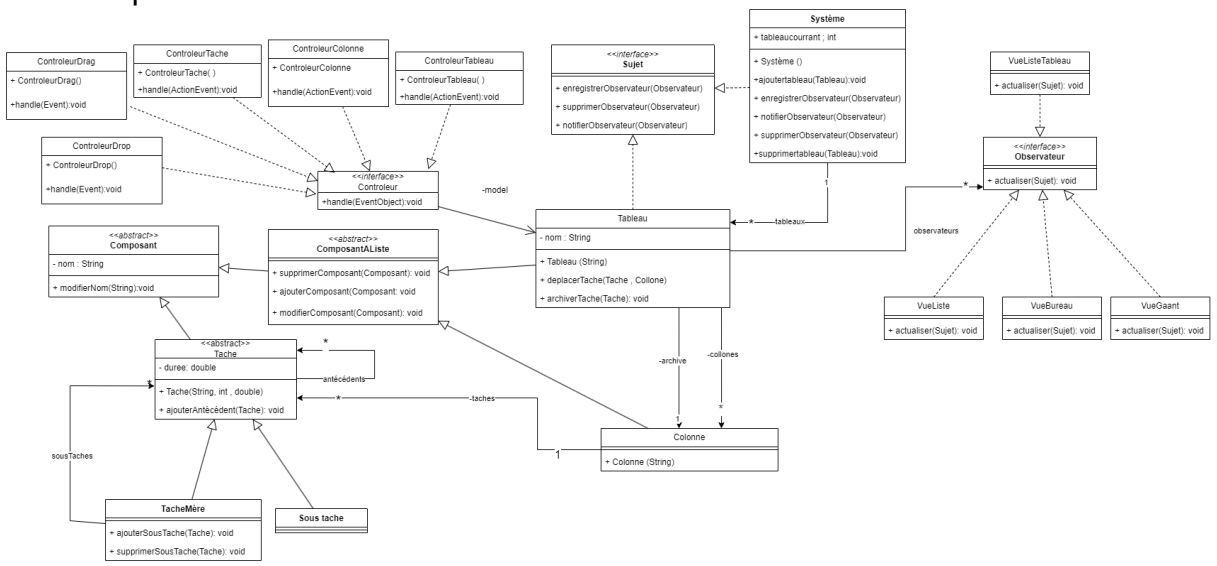


Fait sur Draw.io

V. Le diagramme de classe et ses patrons

- Le diagramme de classe

Enfin nous avons conçu un diagramme de classe regroupant toutes les classes utiles à la conception de l'application. Dedans, nous avons utilisé plusieurs patrons de conception et d'architecture.



Fait sur Draw.io

Notre diagramme est basé sur différents patrons de conception, dont le patron MVC. L'interface Sujet va donc implémenter la classe Tableau (représentant le tableau contenant les colonnes) et la classe Système (contenant des tableaux).

Pour ces 2 sujets, il y a différentes classes de vues passant par l'interface Observateur, il y a la classe pour la vue en liste, en bureau, en diagramme de Gantt et en liste de tableau (à voir sur les maquettes).

Chaque tableau contient une liste de colonnes dont la colonne des archives. Chaque colonne contient une liste de tâches. Et une tâche dite mère contient une liste de sous-tâches.

Les classes Tableau, Colonne et Tâche sont "extends" de la classe Composant, mais les classes Colonne et Tableau ont juste la spécificité d'être des objets de type ComposantAListe car les modifications ne seront pas prises en compte de la même façon.

Pour pouvoir faire des actions sur l'interface de l'application, nous avons ajouté des controllers. On y retrouve un contrôleur qui gère le "drag", un autre pour le "drop", un pour contrôler une tâche, un pour contrôler une colonne et un pour gérer le tableau. Tous ces contrôleurs implémentent l'interface Controler.

- Patrons de conceptions et d'architecture utilisés

Pour concevoir notre application, nous avons visualiser différents patrons de conception et d'architecture tels que:

➢ le patron de conception **Observateur** pour les différentes vues (sous forme de listes des tâches, des tableaux, de bureau et de diagramme de Gantt).

➢ le patron de conception **Composite** pour prendre en compte les tâches et les sous-tâches. Comme dit précédemment, une tâche (dite "mère") peut avoir plusieurs sous-tâches.

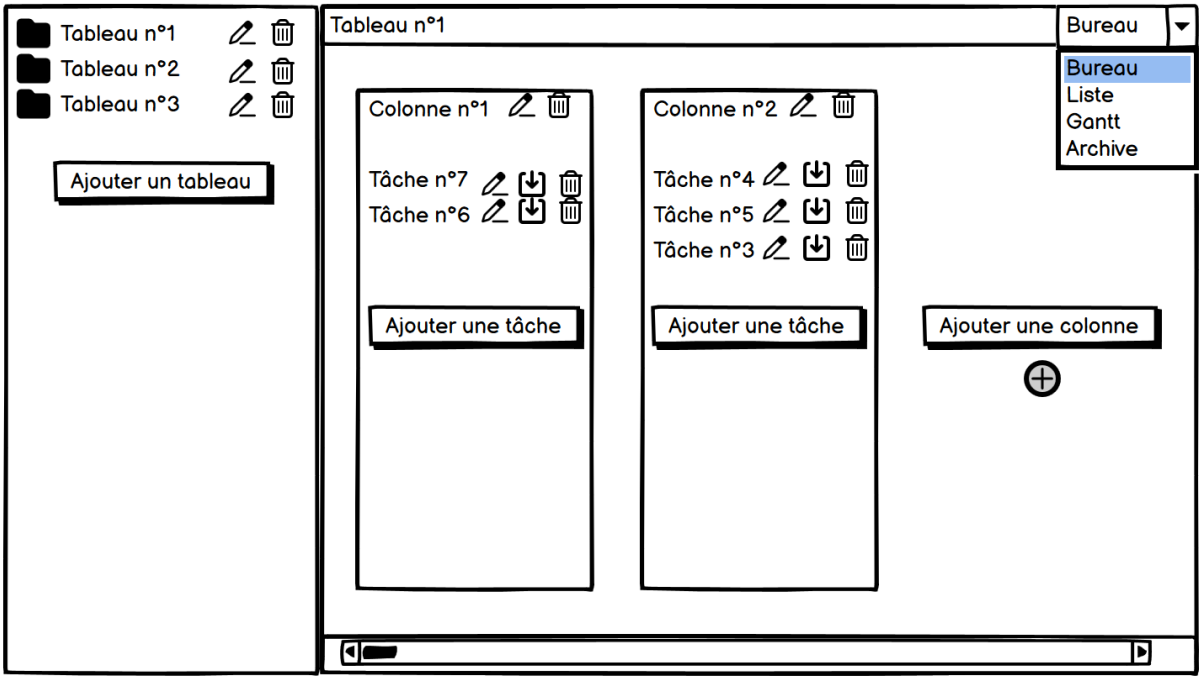
➢ le patron d'architecture **MVC** pour le contrôle et le visuel de l'application.

VI. Maquettage de l'application

A travers ces maquettes nous avons essayé de représenter au mieux l'interface qu'aura notre future application en fonction des différentes vues. Sur toutes ces maquettes, l'utilisateur peut à tout moment changer de tableau.

Maquette de la vue sous forme de tableau:

Cette vue nous permet de visualiser toutes les colonnes de l'application avec les tâches qu'elles contiennent. Sur cette vue, l'utilisateur peut modifier le nom d'une tâche, archiver une tâche, ou bien la supprimer. Il peut faire les mêmes actions sur les colonnes et les tableaux sauf l'action d'archivage.

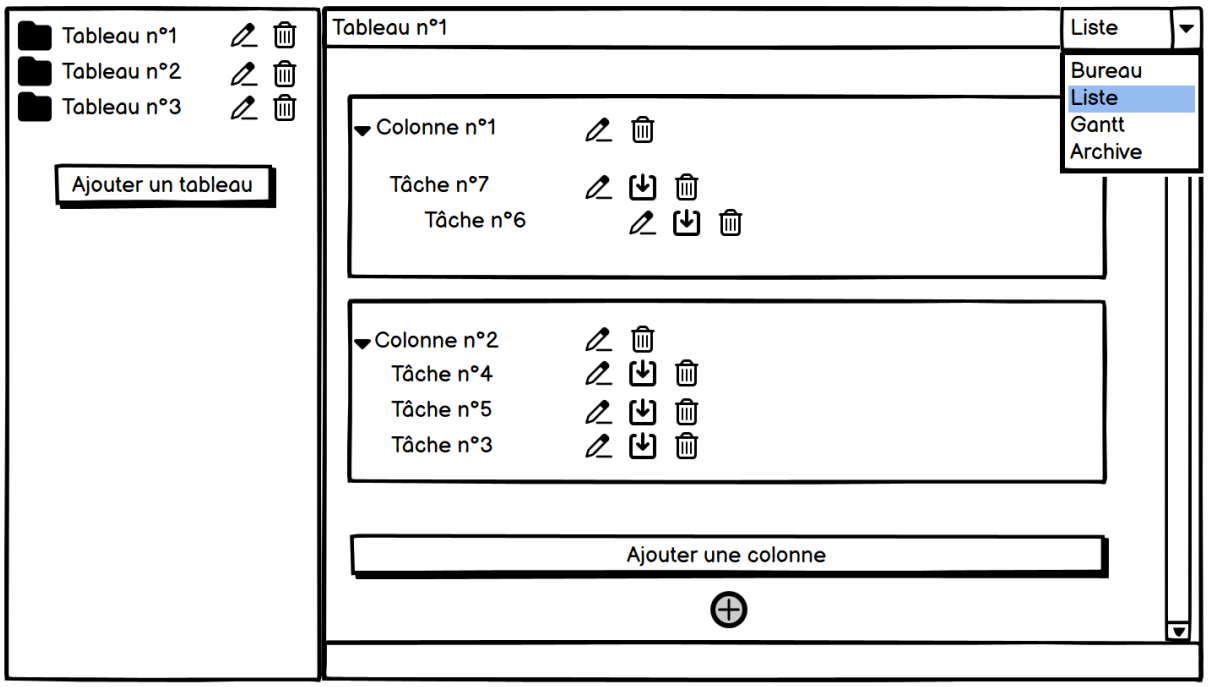


Fait sur Balsamiq

Arcelin Nino	Bitsindou Maëlle	Ponroy Célie	Prykhodko Yehor

<u>Maquette de la vue sous forme de liste:</u>

Cette vue nous permet de visualiser toutes les colonnes de l'application avec les tâches et sous-tâches qu'elles contiennent sous forme de liste horizontale. Sur cette vue, l'utilisateur peut faire les mêmes actions que sur la vue sous forme de tableau.

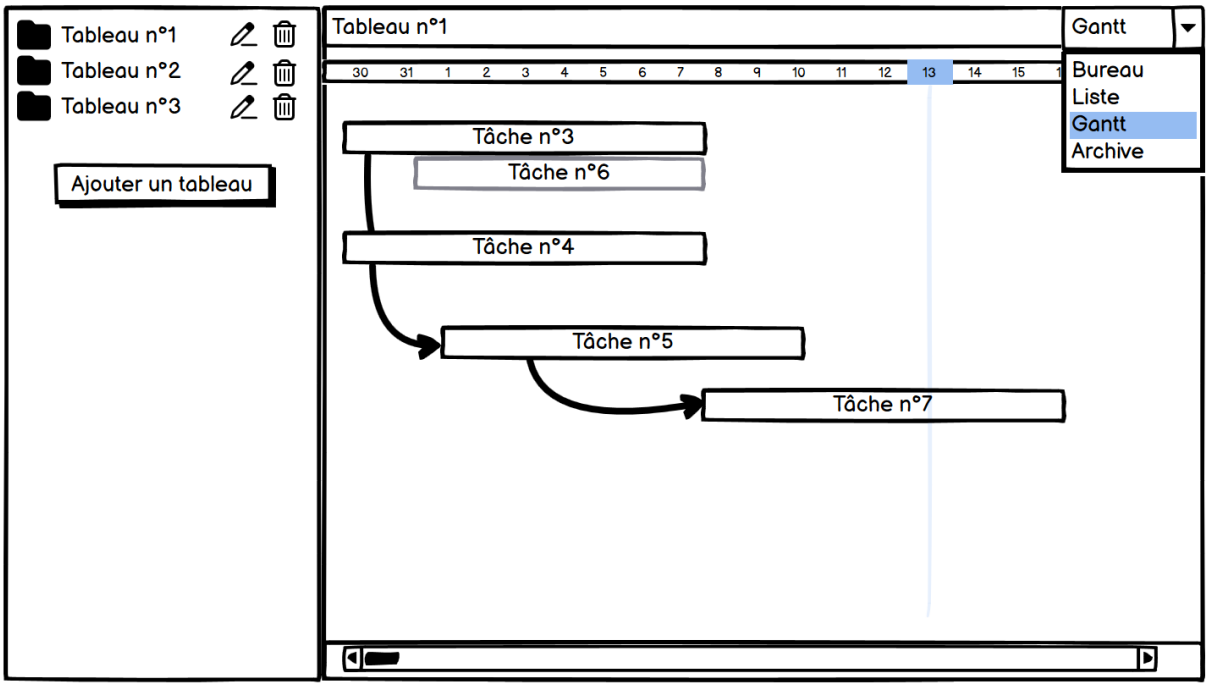


Fait sur Balsamiq

Arcelin Nino	Bitsindou Maëlle	Ponroy Célie	Prykhodko Yehor

<u>Maquette de la vue sous forme de diagramme de Gantt:</u>

Cette vue nous permet de visualiser le diagramme de Gantt de toutes les tâches et donc les sous-tâches sélectionnées. L'utilisateur ne peut pas interagir avec cette vue.

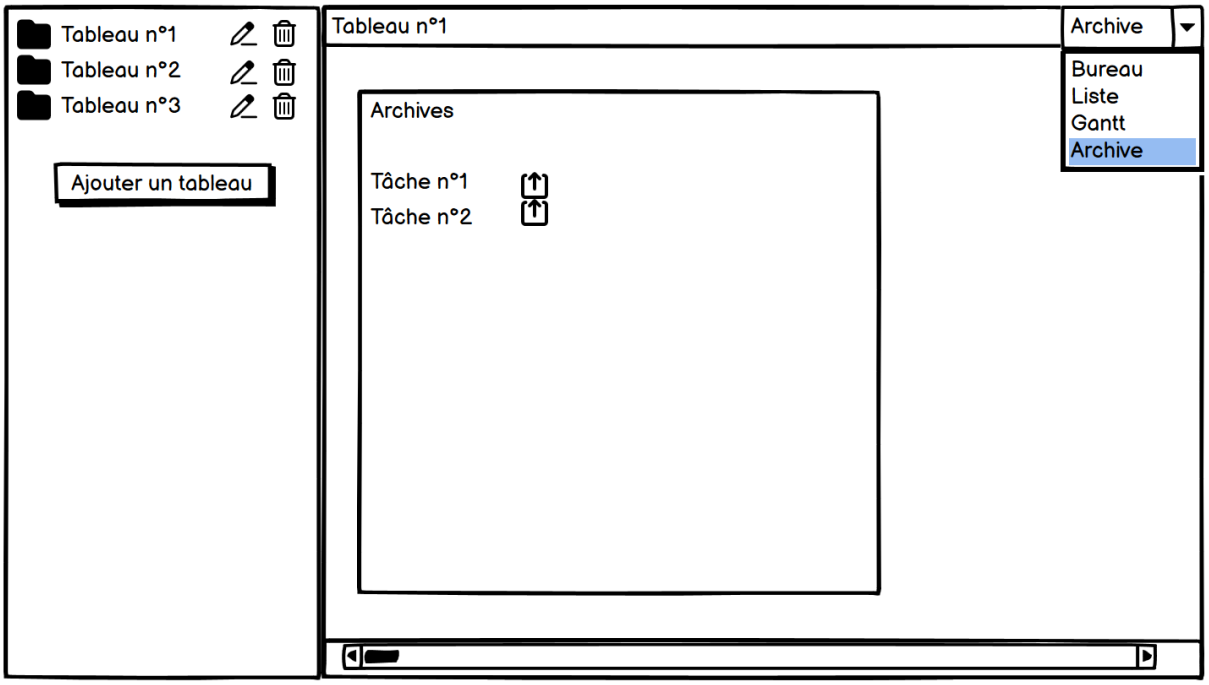


Fait sur Balsamiq

Arcelin Nino	Bitsindou Maëlle	Ponroy Célie	Prykhodko Yehor

Maquette de la vue de la liste des archives:

Cette vue nous permet de visualiser toutes les tâches archivées. L'utilisateur peut "désarchiver" les tâches qui se trouvent dans cette liste.



Fait sur Balsamiq

Arcelin Nino	Bitsindou Maëlle	Ponroy Célie	Prykhodko Yehor

VII. Planning des futures itérations et les potentiels risques à prévoir

Dans ce tableau, nous avons répartis au mieux les futures fonctionnalités que nous souhaitions concevoir dans lors de la conception. D'autre part, nous avons également essayé de trouver les potentiels problèmes que nous pourrions rencontrer.

	Fonctionnalités prévues	**Risques à prévoir**
Itération 1	➢ Gérer une tâche ➢ Gérer une colonne ➢ Gérer un tableau	➢ Vérifier l'unicité des tâches et des colonnes
Itération 2	➢ Afficher le tableau en bureau ➢ Gérer les contrôleurs du bureau	➢ Faire attention aux futures dépendances lors de la suppression d'une tâche
Itération 3	➢ Fonctionnalité "Drag and drop"	➢ Gérer les 2 contrôleurs pour le drag and drop
Itération 4	➢ Gérer les dépendances d'une tâche ➢ Générer le diagramme de Gantt	➢ Cohérence des dépendances ➢ Vérifier le bon affichage des dépendances ➢ Vérifier que la taille de la tâche corresponde à la durée
Itération 5	➢ Afficher le tableau en liste	➢ Vérifier le bon affichage de la liste
Itération 6	➢ Gérer la liste des tâches archivées	➢ Suppression de la tâche dans son ancienne colonne ➢ Vérifier qu'une tâche désarchivée retourne au bon endroit.

VIII. Conclusion et transition vers la conception

Cette étude préalable nous a permis d'établir les bases solides nécessaires à la conception de l'application de gestion de tâches personnelles. En examinant en détail les fonctionnalités, les interactions utilisateur-système, et en envisageant différents scénarios, nous avons pu identifier les besoins primordiaux de notre future application. Les diagrammes, tels que ceux de cas d'utilisation, d'activité, de séquence et d'état, ainsi que le diagramme de classe, fournissent une vision claire et structurée du système à développer. Les patrons de conception et d'architecture sélectionnés démontrent une approche réfléchie pour assurer une conception modulaire, flexible et extensible.

En envisageant les futures itérations et en identifiant les risques potentiels, nous nous préparons à aborder le développement de manière méthodique. Cette phase d'étude préalable marque le point de départ vers la réalisation d'une application répondant aux besoins de gestion de tâches personnelles de l'utilisateur.