SAE 3.01 - Développement d'une application

ARDHUIN / GRAINE / ATTENOT

Liste des Fonctionnalités :

1. Créer une tache
2. Visualiser les dépendances des taches
3. Modifier une tache
4. Archiver des taches
5. Glisser déposer des taches
6. Vue des taches en bureau
7. Vue des taches en liste
8. Vue des taches en diagramme de Gantt
9. Gérer des sous-tâches
10. Supprimer définitivement
11. Gérer les priorités
12. Ajouter une barre de progression

Diagramme de cas d'utilisation globale :

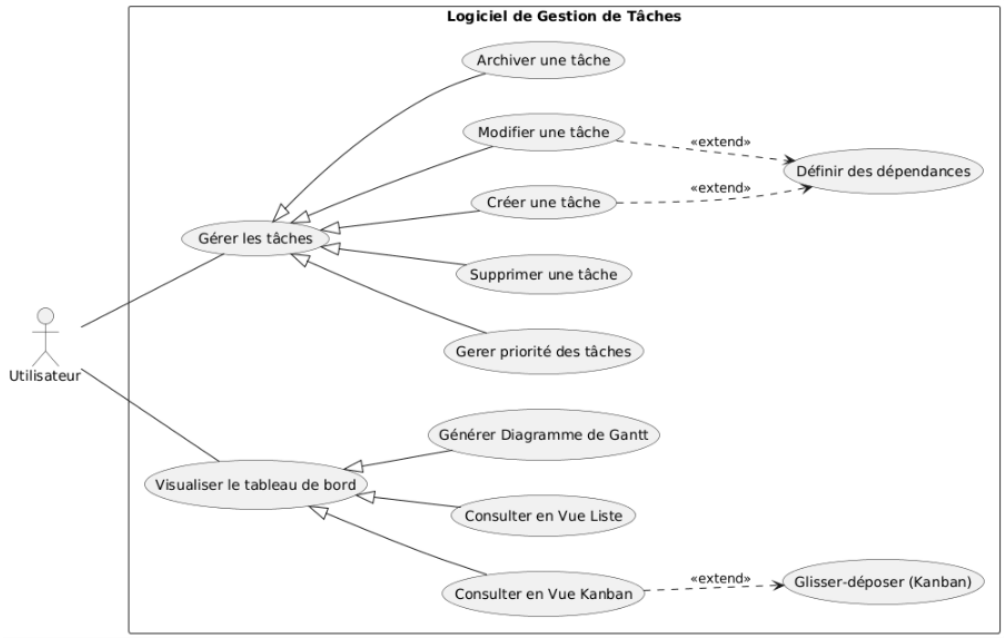


Diagramme D’activité de l’application :

Cas d’utilisation : Créer une tâche

- **Acteur :** Utilisateur
- **Préconditions :** L'application est lancée et l'utilisateur est sur une vue permettant la création.

Scénario :

- L'Utilisateur clique sur le bouton de création d'une nouvelle tâche.
- Le Système affiche le formulaire de création vide (champs : Titre, Description, Priorité...).
- L'Utilisateur saisit les informations de la tâche.
- L'Utilisateur valide la création.
- Le Système vérifie la validité des données (ex: titre non vide).
- Le Système enregistre la tâche avec le statut "À faire".
- Le Système met à jour l'affichage courant avec la nouvelle tâche visible.

Exceptions :

Données invalides : Si le titre est vide, le Système affiche un message d'erreur et demande à l'utilisateur de corriger.

Postconditions : La tâche est créée en base de données.

Cas d'utilisation : Visualiser les dépendances

- **Acteur :** Utilisateur
- **Préconditions :** La tâche sélectionnée existe.

Scénario :

- L'Utilisateur sélectionne une tâche et demande à voir ses détails/dépendances.
- Le Système récupère la liste des tâches liées (tâches bloquantes ou sous-tâches).
- Le Système génère une représentation visuelle (liste hiérarchique ou icônes de blocage).
- Le Système affiche cette visualisation à l'Utilisateur.

Postconditions : L'utilisateur a pris connaissance des liens entre les tâches.

Cas d'utilisation : Modifier une tâche

- **Acteur :** Utilisateur
- **Préconditions :** La tâche est visible à l'écran.

Scénario :

- L'Utilisateur sélectionne une tâche pour l'éditer.
- Le Système récupère les informations actuelles et affiche le formulaire pré-rempli.
- L'Utilisateur modifie un ou plusieurs champs (ex: change la priorité ou la date).
- L'Utilisateur valide les modifications ("Enregistrer").

- Le Système vérifie la validité des nouvelles données.
- Le Système met à jour la tâche dans la base de données.
- Le Système rafraîchit l'affichage pour refléter les changements.

Exceptions :

- *Conflit de dépendance :* Si l'utilisateur tente de créer une boucle (A dépend de B qui dépend de A), le Système refuse la modification et affiche une erreur.

Postconditions : Les attributs de la tâche sont mis à jour.

Cas d'utilisation : Archiver une tâche

- **Acteur :** Utilisateur
- **Préconditions :** La tâche est dans l'état "Terminée".

Scénario :

- L'Utilisateur sélectionne une tâche terminée.
- L'Utilisateur clique sur l'option "Archiver".
- Le Système vérifie que la tâche peut être archivée (elle ne doit pas être en cours).
- Le Système modifie l'état de la tâche à "Archivée".
- Le Système retire la tâche de la vue active (Kanban/Liste).
- Le Système rend la tâche accessible uniquement dans la vue "Archives".

Exceptions :

- *Tâche non finie :* Si la tâche est "En cours", le Système interdit l'archivage et notifie l'utilisateur.

Postconditions : La tâche n'est plus visible sur le bureau mais reste en base de données.

Cas d’utilisation : Glisser-Déposer (Vue Kanban)

- **Acteur :** Utilisateur
- **Préconditions :** L'utilisateur est sur la Vue Bureau (Kanban).

Scénario :

- L'Utilisateur clique sur une tâche et maintient le clic.
- L'Utilisateur déplace la tâche vers une autre colonne (ex: de "À faire" vers "En cours").
- L'Utilisateur relâche la tâche.
- Le Système détecte la nouvelle colonne.
- Le Système met à jour le statut de la tâche en base de données.
- Le Système réorganise l'affichage pour confirmer le déplacement.

Postconditions : L'état de la tâche a changé.

Cas d’utilisation : Supprimer une tâche

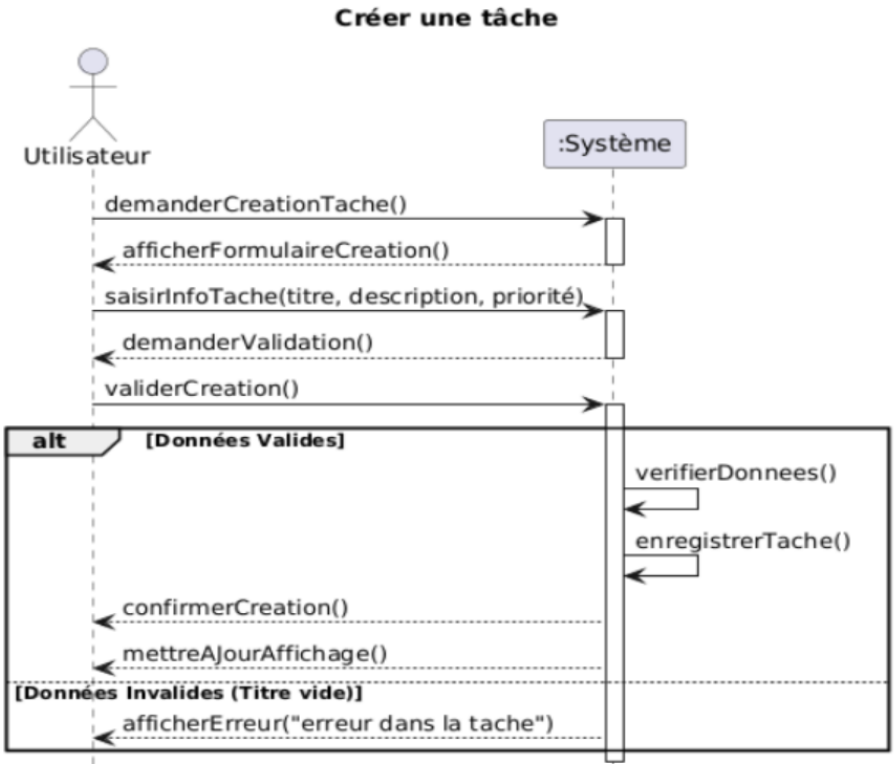
- **Acteur :** Utilisateur
- **Préconditions :** La tâche existe (active ou archivée).

Scénario Nominal :

- L'Utilisateur demande la suppression définitive d'une tâche.
- Le Système affiche une fenêtre de confirmation ("Action irréversible").
- L'Utilisateur confirme.
- Le Système supprime la tâche et ses liens de la base de données.
- Le Système met à jour l'affichage (la tâche disparaît).

Postconditions : La tâche n'existe plus du tout.

DSS Créer une tache :



DSS déplacer une tache :

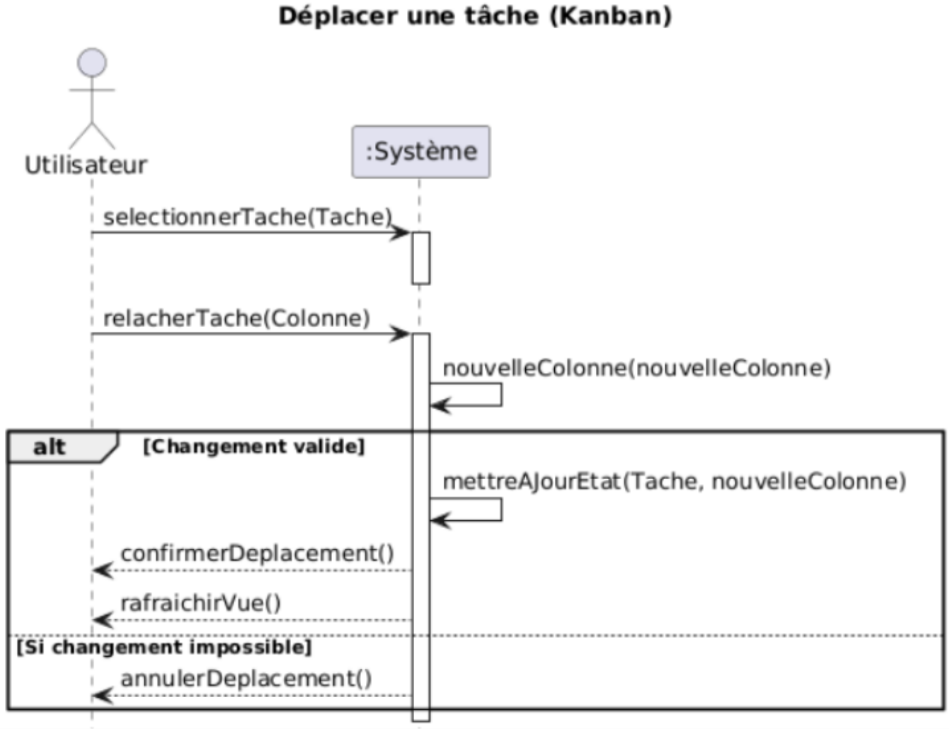


Diagramme d’état d’une tache :

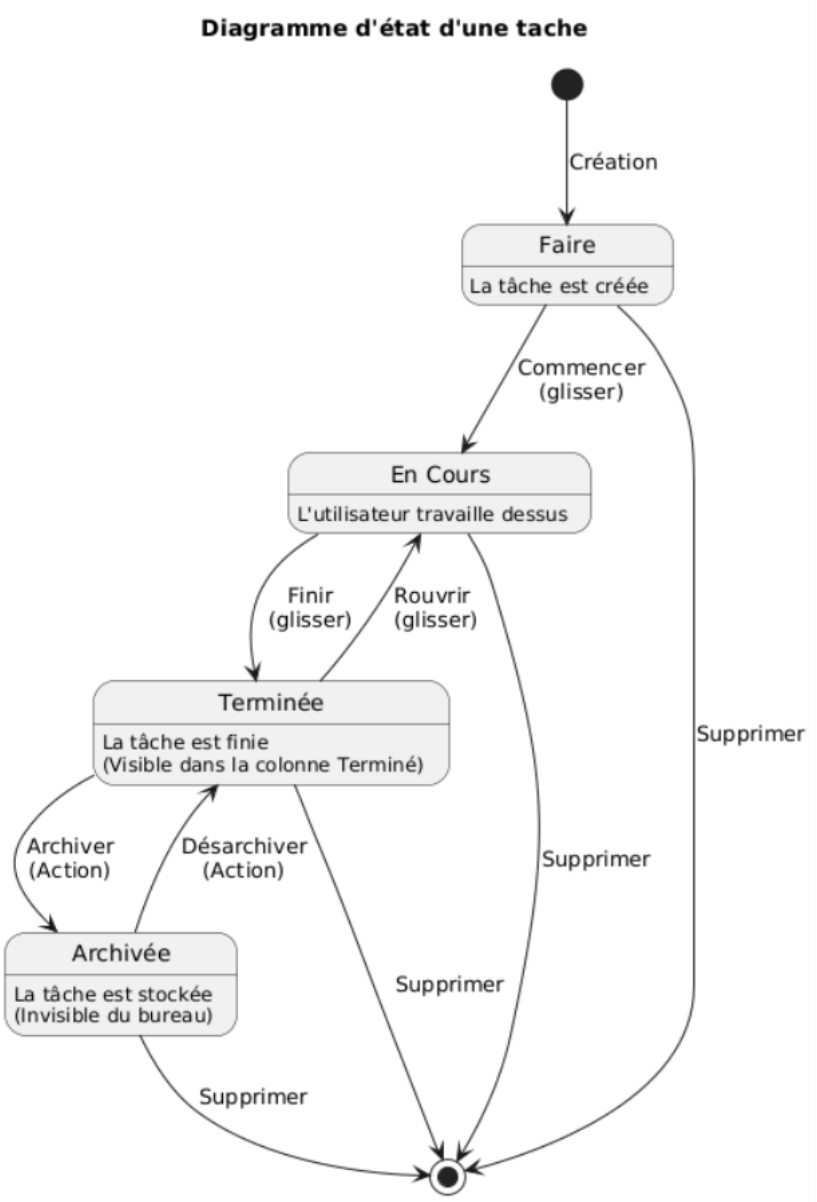
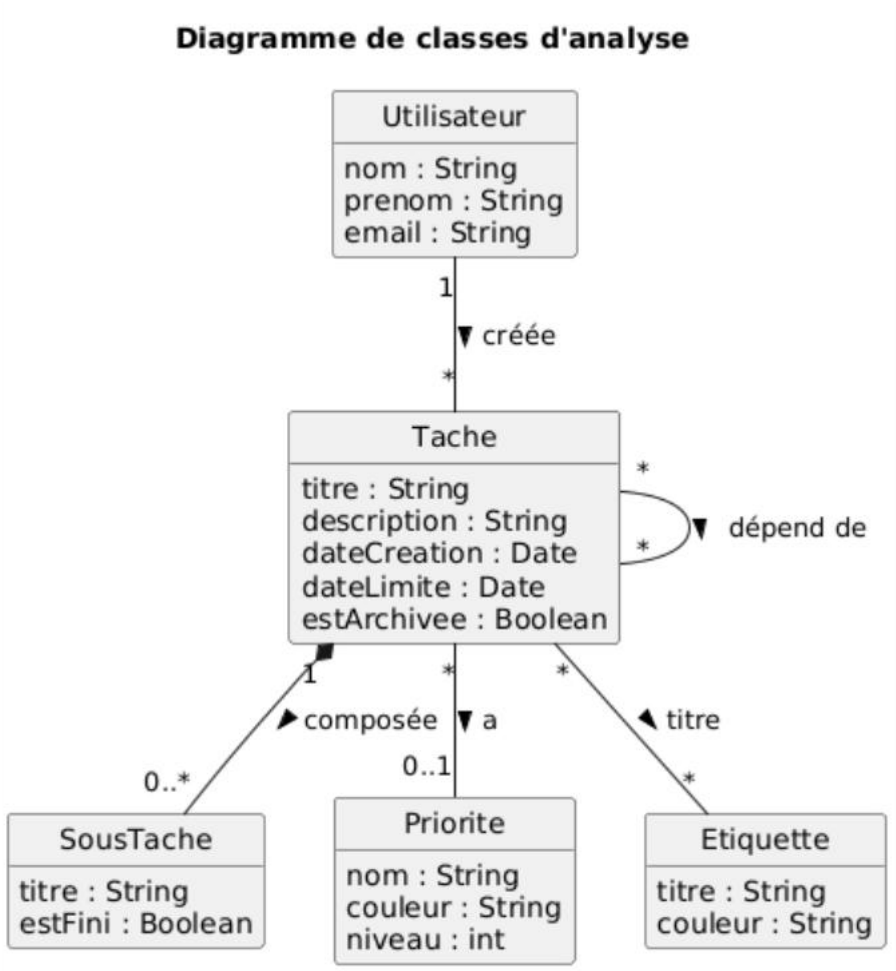


Diagramme de classe (Analyse) :



Patrons de conception utilisés dans notre application

Dans le cadre de la réalisation de notre organisateur de tâches en JavaFX, plusieurs patrons de conception seront utilisés afin d'assurer une architecture conforme aux exigences de la SAE. Les choix suivants sont cohérents avec l'architecture MVC et avec les fonctionnalités de gestion de tâches, sous-tâches, dépendances et visualisations (Kanban, listes, Gantt).

MVC (Patron d'architecture)

L'architecture MVC est imposée pour ce projet. Le modèle contient les données de l'application (tâches et sous-tâches), les vues correspondent à l'affichage JavaFX (colonnes du Kanban, listes, etc.), et les contrôleurs gèrent les

interactions de l'utilisateur. Ce modèle permet de séparer clairement les responsabilités et de maintenir un code plus lisible.

Observer

Le patron Observer permet de synchroniser automatiquement l'interface graphique avec les données du modèle. Lorsque l'état d'une tâche change, les vues associées sont notifiées et se mettent à jour. JavaFX utilisant déjà des listes et propriétés observables, ce patron s'intègre naturellement et simplifie la mise à jour de l'affichage.

Composite

Les tâches peuvent contenir des sous-tâches, ce qui crée une structure hiérarchique. Le patron Composite répond directement à ce besoin en permettant de traiter de manière uniforme une tâche simple et une tâche composée. Grâce à ce patron, l'application gère facilement des structures imbriquées et cohérentes.

Singleton

Un Singleton est utilisé pour centraliser certains éléments accessibles globalement dans l'application. Par exemple, il peut servir à gérer l'accès au projet courant, aux paramètres ou aux données partagées entre plusieurs contrôleurs. Cela garantit qu'il n'existe qu'une seule instance de cet objet dans toute l'application.