Rapport final – Projet Application de Gestion de Tâches

Projet : Application de Gestion de Tâches

Groupe : ***Marchal Enzo, Laghezali Nacime, Cabot Matthieu, Vincent Julien***

Introduction

Ce rapport présente le bilan du projet de développement d'une application de gestion de projet réalisée en groupe. L'objectif était de concevoir une application JavaFX permettant de gérer des projets, des tâches et des sous-tâches à travers plusieurs vues (kanban, gantt, liste, etc.).

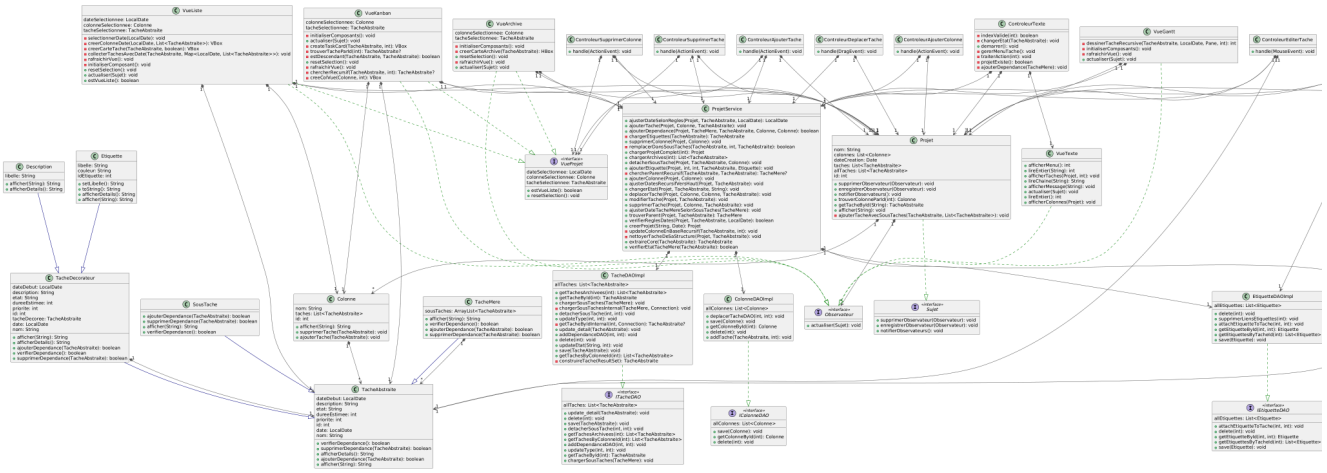
Le projet a été mené de manière itérative, avec plusieurs itérations successives permettant d'enrichir progressivement les fonctionnalités et d'améliorer l'architecture logicielle.

1. Liste des fonctionnalités réalisées

- Création, modification et suppression de projets
- Gestion des colonnes (type Kanban)
- Ajout, édition et suppression de tâches
- Gestion de sous-tâches
- Gestion des dates (début, fin, durée)
- Gestion des étiquettes
- Ajustement automatique des dates selon les sous-tâches
- Vue Kanban
- Vue Gantt
- Vue Liste
- Vue Dashboard
- Archivage de projets
- Persistance des données via DAO
- Mise en place de tests unitaires
- Navigation entre les différentes vues

2. Architecture et Conception

2.1 Diagramme de classe final



2.2 Patrons de conception

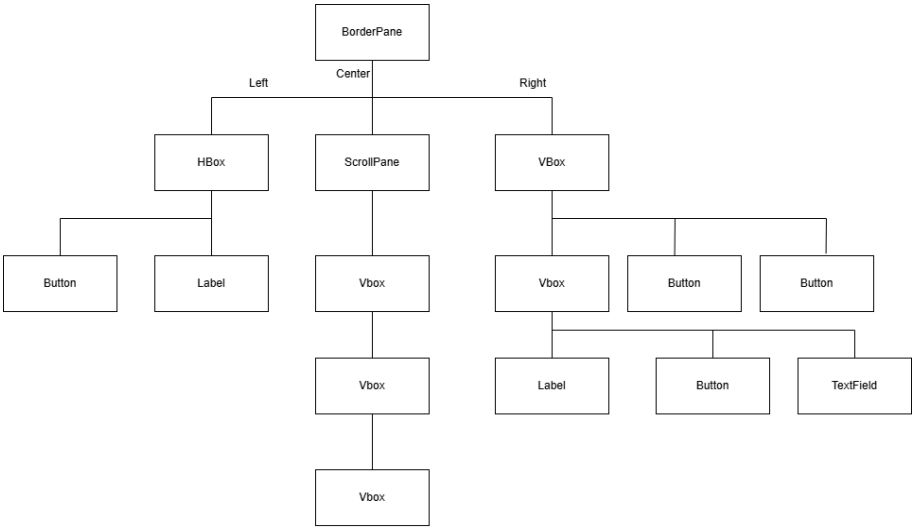
- **Composite** : pour la gestion des tâches sous-tâches avec les dépendances
- **Décorateur** : pour pouvoir ajouter des étiquettes au tâches (plus tard on peut ajouter des membres ou même des pièces jointe qui n'ont pas été traité ici)
- **Observateur** : pour la mise à jour des vues
- **Singleton** : pour pouvoir faire la connection avec la base de donnée
- **DAO** : pour la sauvegardes des données dans une base de donnée
- **Stratégie :** pour les vues kanban et liste
- **MVC** : Pour le patron d'architecture

3. Interface Graphique

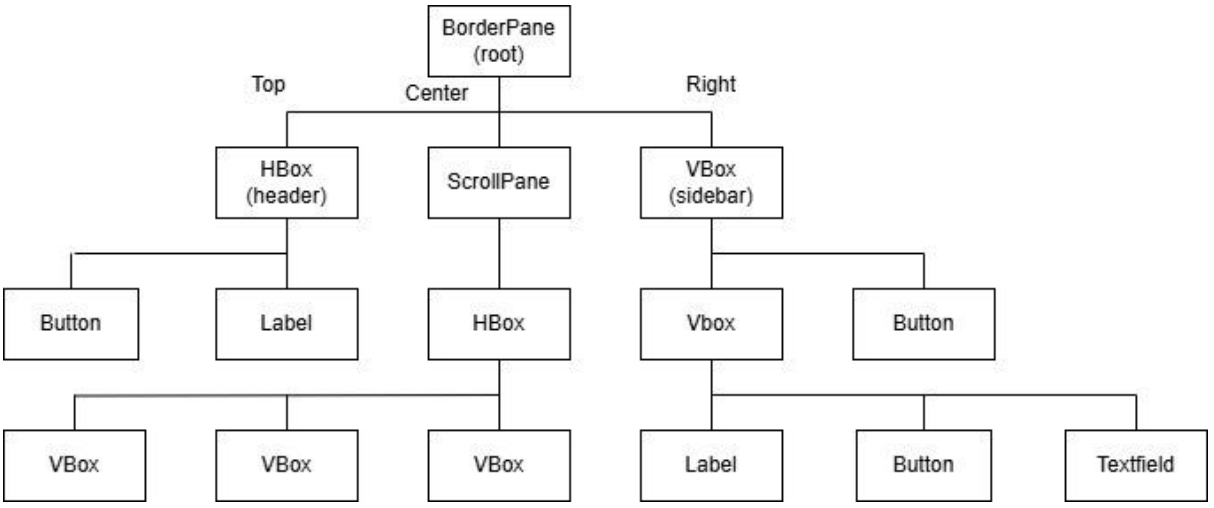
3.1 Graphe de scène

Organisation des vues :

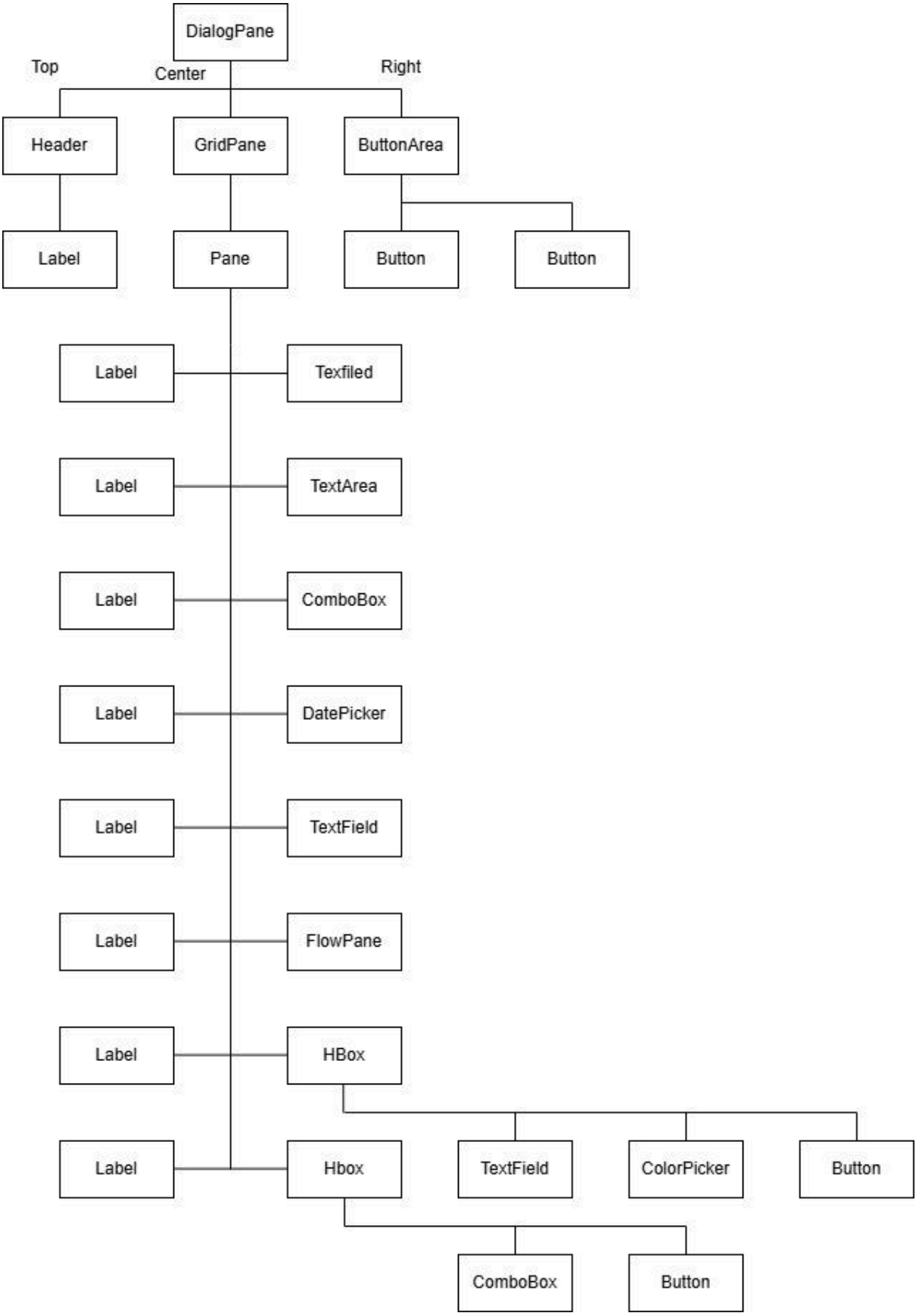
- VueListe



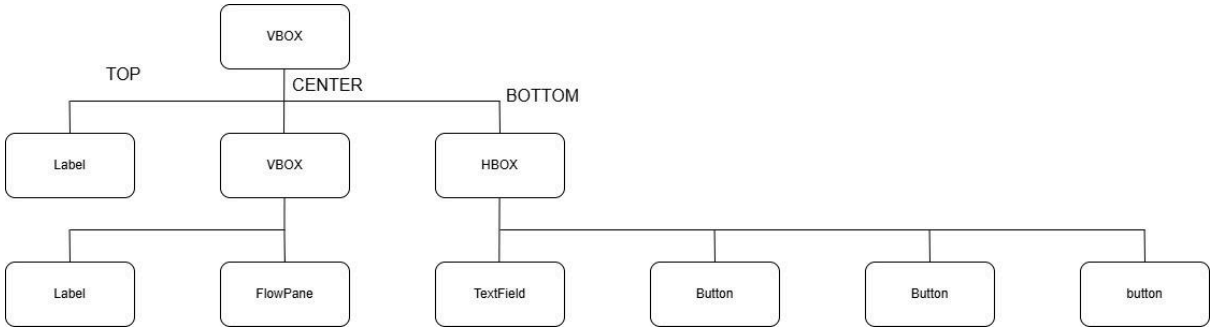
- Vue Kanban



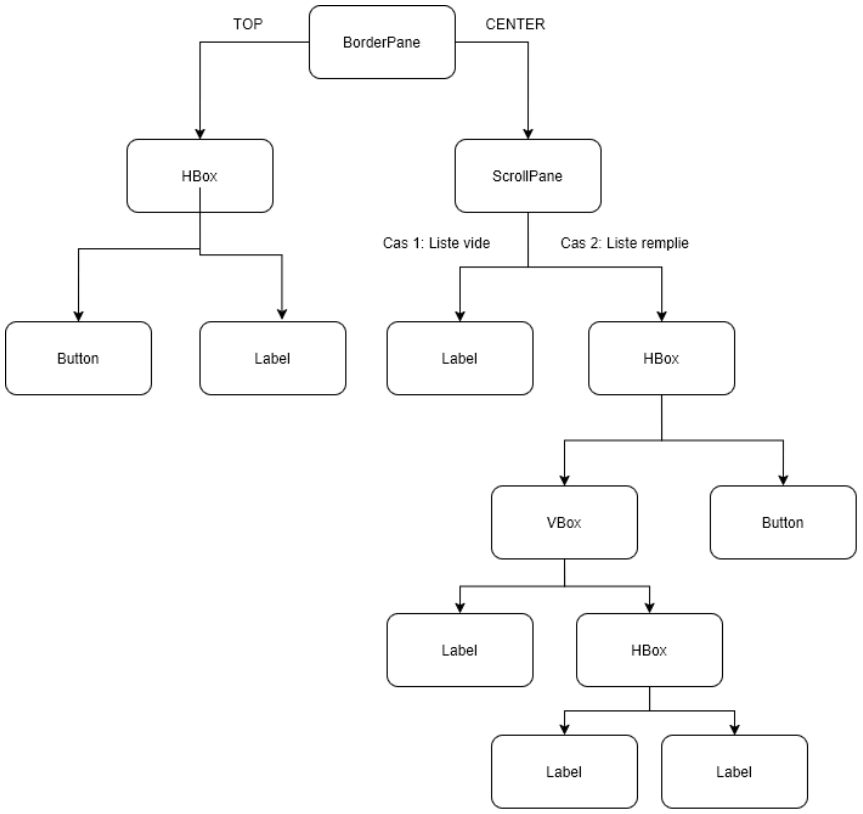
- Vue Description



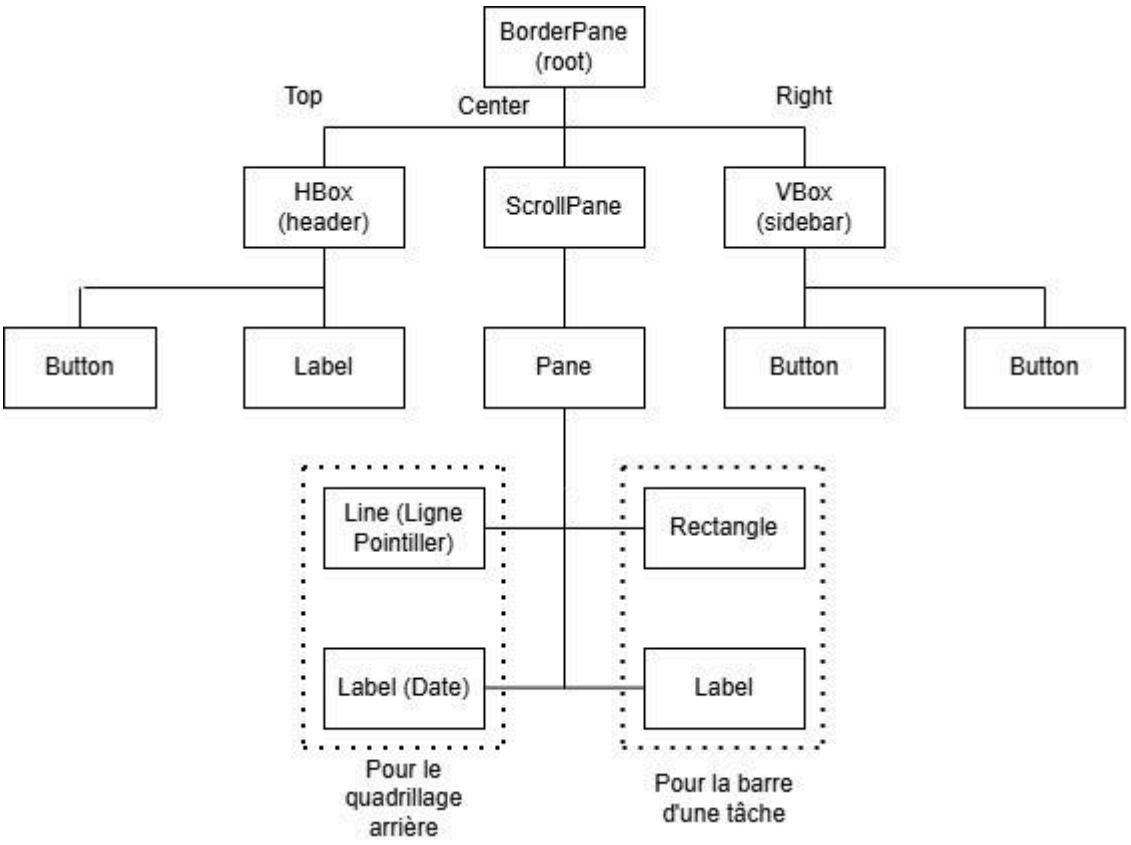
- Vue Dashboard



- Vue Archive



- Vue Gantt



3.2 Navigation et changement de vue

Le changement de vue dans l'application est géré par la classe MainApp, qui joue le rôle de contrôleur principal de navigation. Chaque vue (Dashboard, Kanban, Liste, Gantt, Archives) est affichée grâce à une méthode dédiée (afficherDashboard, afficherKanban, afficherListe, etc.). Lorsqu'un utilisateur clique sur un bouton de navigation, un événement est déclenché et appelle directement la méthode correspondante de MainApp.

Concrètement, une nouvelle instance de la vue demandée est créée (par exemple new VueKanban(...)) puis intégrée dans une nouvelle Scene. Cette scène est ensuite appliquée à la fenêtre principale grâce à primaryStage.setScene(scene). Cela permet de remplacer entièrement l'interface affichée sans relancer l'application.

Les boutons présents dans les barres supérieures et latérales sont configurés dynamiquement : MainApp parcourt les composants graphiques (HBox, VBox) pour identifier les boutons et leur associer des EventHandler. Par exemple, un bouton "Vue Liste" appelle la méthode afficherListe(...), tandis que le bouton retour utilise le contrôleur ControleurRetourDashboard.

Avant chaque changement de vue, le projet est rechargé depuis la base de données via projetService.chargerProjetComplet(...), ce qui garantit que les données affichées sont toujours à jour.

4. Organisation du projet

4.1 Répartition du travail

Tableau récapitulatif de la charge de travail par itération.

Itération	Tâches principales	Étudiant responsable
Itération 1	Lors de la première itération, je me suis principalement occupé de la conception globale du projet. J'ai participé à la réflexion sur l'architecture générale de l'application ainsi que sur l'organisation des différentes parties du programme. J'ai également contribué à l'élaboration d'un main temporaire permettant de tester l'application durant les premières phases de développement. Enfin, j'ai travaillé sur la gestion des dépendances et sur la définition des états d'une tâche (à faire, en cours, terminée, etc.).	Enzo
	Lors de la première itération, j'ai créé les bases de la classe Tache, avec l'interface TacheAbstraite et TacheMere et SousTache. J'ai également élaboré la base de données qui sera utilisée comme moyen de sauvegarde pour l'application.	Julien
	Lors de la première itération, j'ai créé les bases de la classe Projet, J'ai également contribué avec enzo sur la première version de la vue texte temporaire permettant d'avoir une base dès le début afin de pouvoir tester le projet.	Nacime

	Pour cette première itération, je me suis occupé dans un premier temps de la création de la classe concrète Colonne avec la gestion d'une colonne et ses méthodes de base. Nous avons de manière générale discuté de la conception globale du projet (utilisation d'une base de données par exemple), avec l'ensemble des fonctionnalités que nous voulions réaliser ainsi que quelle partie chacun maîtrisait le mieux. J'ai ainsi choisi de me diriger vers le développement de la partie graphique de l'application. Pour la suite de cette itération, j'ai effectué une première esquisse en JavaFX de la vue Dashboard pour la gestion des projets, car nous savions que nous partirions sur une architecture MVC. Cela se résume par un code en dur qui montre graphiquement ce à quoi pourrait ressembler cette vue, ainsi qu'un graphe de scène pour montrer les éléments graphiques JavaFX utilisés.J'ai terminé en effectuant des tests Unitaires sur les premières classes concrètes que nous avions développées.	Matthieu
Itération 2	Durant cette itération, je me suis occupé du développement d'une véritable vueTexte fonctionnelle. Cette vue permettait de tester le projet sans interface graphique complète. En parallèle, j'ai réfléchi à la suite du projet et préparé les prochaines itérations à l'aide de diagrammes afin d'anticiper l'évolution de l'architecture.	Enzo
	Lors de la deuxième itération, je me suis occupé de mettre en place le patron DAO afin de relié la base de donnée à l'application. Le patron DAO agira comme un intermédiaire entre le code et la base de donnée.	Julien
	Lors de la deuxième itération, je me suis occupé de l'implémentation du patron observateur avec MVC, afin de faciliter par la suite l'ajout de nouvelle vue.	Nacime
	Pour cette deuxième itération, j'ai continué le développement de l'interface graphique avec les bases de la vue Kanban, son graphe de scène, ainsi que les premières fonctionnalités de cette vue. Cela inclut l'ajout	Matthieu

	et la suppression d'une tâche grâce à des contrôleurs (ControleurAjouterTache, ControleurSupprimerTache), et l'affichage des colonnes.	
Itération 3	Lors de cette phase, je me suis occupé de faire le lien entre le projet et la base de données. J'ai implémenté la logique de sauvegarde des projets et veillé à ce que les données soient correctement enregistrées et rechargées. Mon objectif était d'assurer un fonctionnement fiable du système de persistance.	Enzo
	Lors de la troisième itération, j'ai implémenté la gestion des étiquettes avec mon camarade Nacime. Je me suis également occupé d'intégrer les méthodes du DAO dans les classes concrètes, permettant d'utiliser la base de donnée dans les différentes vues.	Julien
	Lors de la troisième itération, je me suis occupé de l'implémentation du patron décorateur pour gérer les étiquettes sur les tâches, avec la possibilité d'ajouter des étiquettes a des taches. J'ai également travaillé sur la vue détaillée d'une tâche pour la version textuelle	Nacime
	Pour cette troisième itération, j'ai finalisé les fonctionnalités de base de la vue Kanban avec la sélection d'une colonne pour pouvoir y ajouter ou supprimer une tâche. J'ai aussi ajouté la possibilité de créer/supprimer une colonne ainsi que le drag and drop d'une tâche entre les différentes colonnes en m'inspirant de celui qui nous avait été donné. J'ai ensuite modifié certains de mes contrôleurs pour implémenter l'utilisation du DAO sur toute la vue Kanban. Pour finir, j'ai créé une méthode de chargement pour que la vue Kanban puisse charger un projet depuis la base de données (avec ses colonnes et tâches), ainsi qu'une vue détaillée qui permet de visualiser le nom et la description d'une tâche.	Matthieu

Itération 4	Cette itération a été consacrée au développement complet de la VueListe. J'ai conçu l'interface, implémenté l'affichage des tâches et assuré la mise à jour dynamique en fonction des modifications du projet. Cette vue constitue l'un des éléments centraux de l'application.	Enzo
	Lors de cette itération, j'ai commencé à m'occuper de la gestion des sous-tâches. Je me suis occupé de la création des sous-tâches par Drag and Drop, de la gestion graphique et dans la base de donnée. Les sous-tâches sont un des éléments les plus techniques et importants de l'application, j'ai dut donc faire attention à gérer tout les cas.	Julien
	Lors de la quatrième itération, je me suis occupé d'implémenter la gestion des étiquettes dans vue détaillée la vue graphique, donc avec la possibilité d'ajouter des étiquettes et de les visualiser dans la vue détaillée mais aussi dans la vue kanban.	Nacime
	Lors de la quatrième itération, j'ai repris mes travaux de la première itération pour implémenter la VueDashboard afin qu'elle soit fonctionnelle et permette de gérer nos projets. Cela est passé par la création de contrôleurs (ControleurDashboardAjouter, ControleurDashboardSupprimer, ControleurDashboardOuvrir) ainsi que des méthodes afficherDashboard() et afficherKanban() qui me permettaient de naviguer d'une vue à l'autre. Pour la suite de cette itération, je me suis occupé de finir la vue détaillée d'une tâche en y affichant l'ensemble de ses informations ainsi que la possibilité de les modifier, notamment par le contrôleur ControleurEditerTache().	Matthieu
Itération 5	Je me suis occupé de la mise en place des règles de gestion des tâches, notamment sur les dates et leur réajustement automatique car nous avons décidé que les dépendances soient des dépendances strictes, c'est-à-dire que les tâches mères ne peuvent pas commencer tant que toutes ses sous-tâches ne soient	Enzo

	pas terminées. J'ai également travaillé sur la gestion des états : par exemple, une tâche mère ne peut pas être marquée comme terminée tant que toutes ses sous-tâches ne le sont pas. Cette itération a renforcé la cohérence fonctionnelle du projet.	
	Lors de cette itération, j'ai finalisé la gestion des sous-tâches, notamment sur le détachement d'une sous-tâche et sa réatribution. J'ai également travaillé sur la vue Archive, permettant d'archiver des tâches et de les restaurer.	Julien
	Lors de la cinquième itération, je me suis occupé de finir complètement la gestion des étiquettes dans le projet, avec tout d'abord une résolution du bug d'affichage des étiquettes dans la vue liste et avec l'ajout de la possibilité d'attribuer des étiquettes déjà existante dans le projet sans les recréer.	Nacime
	Pour cette cinquième itération, je me suis occupé de la conception et de la réalisation de la vue Gantt, ce qui m'a occupé toute la durée de l'itération. Cela est passé par la réalisation du graphe de scène pour imaginer le résultat visuel, ainsi que par le développement des différentes méthodes de parcours de l'ensemble des tâches d'un projet. C'est cette fonctionnalité que j'ai décidé de développer dans la suite de ce compte-rendu.	Matthieu
Itération 6	Cette dernière itération a été principalement dédiée à la correction de bugs, aux tests finaux et à la vérification du bon fonctionnement global de l'application. J'ai participé à la stabilisation du projet avant le rendu final.	Enzo
	Lors de cette dernière itération, j'ai corrigé différent bug qui, notamment lié à la gestion du chargement des sous-tâches. J'ai également testé les différentes fonctionnalités du projet pour vérifier son bon fonctionnement.	Julien

	Lors de cette sixième itération, je me suis occupé de préparer le rapport final ainsi que la préparation de la soutenance, avec la préparation du diaporama ainsi que plusieurs diagrammes de séquence et de classe.	Nacime
	Cette sixième itération avait pour thème, comme pour mes camarades, la finalisation du projet, la correction de bugs (notamment sur la mise à jour d'informations d'une tâche qui ne se faisait pas correctement dans certains cas), ainsi que la préparation du compte-rendu final, de la présentation et du DS de fin de SAE. J'ai tout de même ajouté une finition au niveau de la vue Gantt pour mieux visualiser l'arborescence des tâches, avec un changement de couleurs selon le niveau de profondeur d'une tâche (tâche mère, sous-tâche, etc.).	Matthieu

5. Contributions Personnelles

5.1 Contribution de Enzo Marchal

- **L'élément réalisé :** VueListe

  L'élément du projet dont je suis le plus fier est la VueListe, qui permet d'afficher les tâches sous forme de liste chronologique organisée par dates. Chaque tâche possède une date de début, une durée et une date de fin calculée automatiquement, offrant ainsi une lecture claire du planning. L'aspect le plus intéressant de cette vue réside dans la gestion intelligente des dates : lorsqu'une sous-tâche est ajoutée, modifiée, la tâche mère est automatiquement recalculée. Sa date de début correspond à celle de la première sous-tâche, avec un ajustement automatique si une sous-tâche dépasse la durée prévue et fonctionnelle aussi avec le drag and drop. J'ai également mis en place des règles métier, par exemple une tâche mère ne peut pas être considérée comme terminée tant que toutes ses sous-tâches ne le sont pas, garantissant ainsi la cohérence du projet. Techniquement, la VueListe repose sur JavaFX, des VBox dynamiques, un ScrollPane et le pattern Observateur, permettant une actualisation automatique grâce aux méthodes rafraichirVue() et actualiser(Sujet s). À chaque modification du projet, l'interface se met à jour sans action de l'utilisateur. Sur cette partie, j'ai conçu l'interface, développé toute la logique métier liée aux dates, implémenté les contrôleurs associés et optimisé l'ergonomie ou je me suis inspiré de la vueKanban de Matthieu en grande partie. Cette vue représente pour moi l'élément le plus abouti du projet que j'ai fait, car elle m'a demandé une importante réflexion algorithmique et constitue une véritable

réussite car tout fonctionne comme je le voulais et c'est très satisfaisant d'avoir réussi cette fonctionnalité et d'avoir su gérer les problèmes ou bug. Elle répond pleinement aux besoins fonctionnels en proposant une gestion automatisée, fiable et cohérente des dates, ce qui réduit fortement les erreurs humaines. Elle améliore également l'expérience utilisateur grâce à une interface claire, intuitive et toujours à jour. Enfin, la stabilité du système, validée par de nombreux tests, ainsi que l'intégration fluide avec les autres vues démontrent la qualité de la conception.

5.2 Contribution de Nacime Laghezali

- **L'élément réalisé :** Gestion complète des étiquettes

L'élément du projet dont je suis le plus fier est la mise en place du système d'Étiquettes, qui permet d'enrichir les tâches avec des informations visuelles supplémentaires, comme des catégories ou des indicateurs avec la couleur. Chaque tâche peut ainsi être personnalisée dynamiquement sans modifier sa structure interne, offrant une grande flexibilité à l'utilisateur pour organiser son travail. L'aspect le plus intéressant de cette fonctionnalité réside dans sa modularité : une même tâche peut cumuler plusieurs étiquettes (par exemple "nacime" et "enzo") qui s'empilent. Pour améliorer le système, j'ai rajouté le fait de permettre l'attribution d'étiquettes existantes : l'utilisateur peut piocher directement dans une liste déroulante d'étiquettes du projet (ComboBox) sans avoir à les recréer à chaque fois. Techniquement, cette partie repose sur le patron de conception Décorateur. J'ai implémenté une classe abstraite TacheDecorateur qui enveloppe une TacheAbstraite. Ce choix architectural est crucial : il évite de modifier la classe de base pour chaque nouveau type d'ajout. Sur cette partie, j'ai développé toute la logique d'encapsulation et assuré l'intégration des étiquettes, elles sont désormais enregistrées en base de données et manipulables de manière synchronisée à travers toutes les Vues du projet (Vue Kanban, Vue Liste et Vue Détaillée).

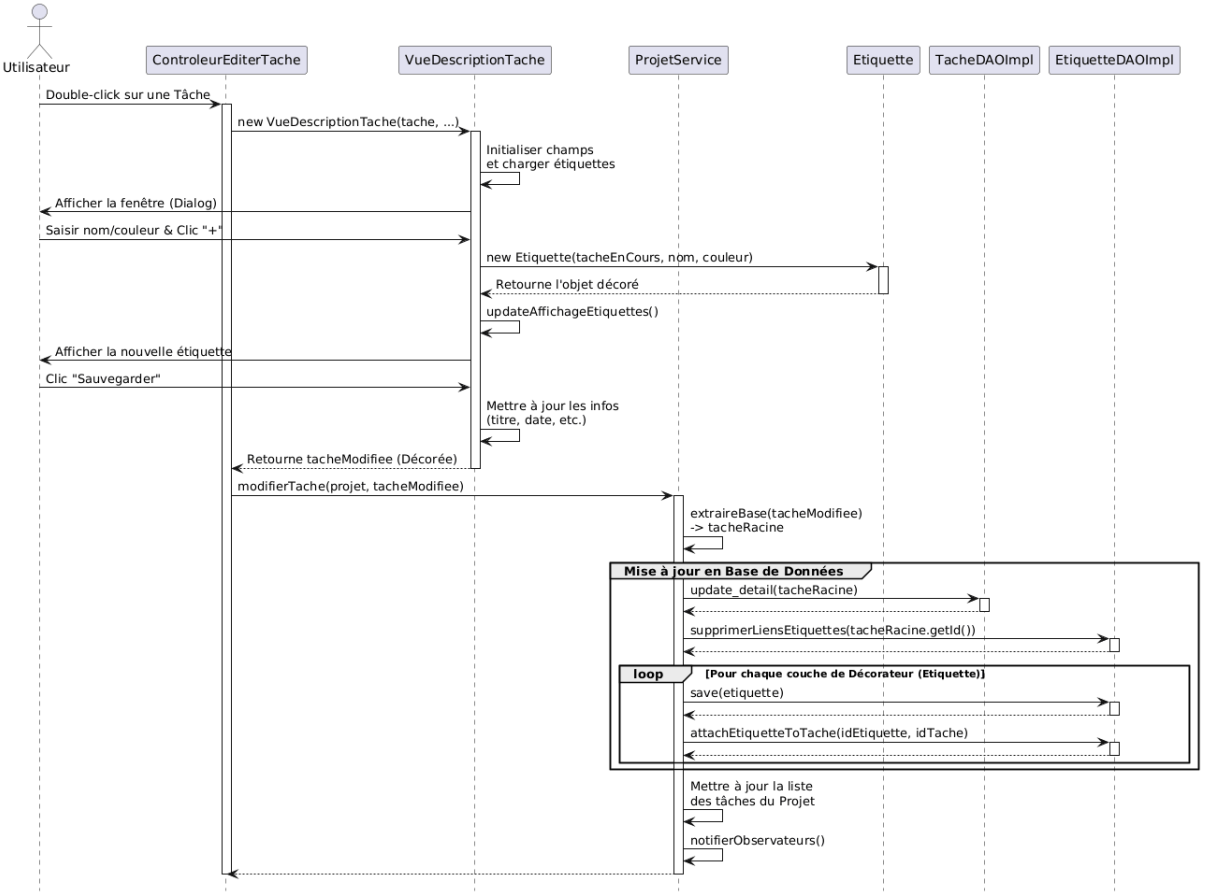


*Diagramme de Séquence de l'ajout d'étiquettes

5.3 Contribution de Julien Vincent

- **L'élément réalisé :** Gestion complète des sous-tâches

  L'élément du projet dont je suis le plus fier est l'implémentation des sous-tâches, qui permet de structurer hiérarchiquement le projet en décomposant des tâches complexes en unités plus petites et gérables. Chaque tâche peut devenir une tâche mère en accueillant des sous-tâches, créant ainsi une arborescence claire et détaillée du travail à accomplir. L'aspect le plus intéressant de cette fonctionnalité réside dans l'utilisation du patron de conception Composite : il permet de traiter de manière uniforme les tâches simples et les tâches composées, simplifiant grandement le code client. Techniquement, cette structure repose sur une modélisation objet avec une classe abstraite TacheAbstraite et des classes concrètes TacheMere et SousTache, intégrées via le pattern Observateur pour propager instantanément les changements dans toutes les vues, du Kanban au Gantt. Sur cette partie, j'ai conçu l'architecture des classes, développé les algorithmes de récursion pour le parcours de l'arbre, implémenté la persistance en base de données avec gestion des clés étrangères et optimisé les interactions utilisateur pour permettre la création intuitive de dépendances. Cette fonctionnalité représente pour moi l'élément le plus abouti du projet que j'ai réalisé, car elle m'a demandé une importante réflexion sur la structure de données et constitue une véritable réussite car la hiérarchie se manipule avec une grande fluidité Elle améliore l'expérience

utilisateur en rendant l'organisation logique et visuelle, tout en garantissant l'intégrité des données

5.4 Contribution de Matthieu Cabot

- **L'élément réalisé :** Vue Gantt

J’ai choisi de détailler la conception et le développement de la Vue Gantt, car c'est la fonctionnalité qui m'a le plus intéressé durant cette SAE. Pour rappel, l’objectif de cette vue était de représenter visuellement l’ensemble des tâches sous la forme d’un diagramme de Gantt, positionnées selon leur date de début et leur durée.

C’est cette fonctionnalité qui m'a posé le plus de défis conceptuels, l'idée de développer un tel diagramme en JavaFX m'effrayait un peu au départ. Ma première interrogation portait sur le choix du composant graphique de base. J’ai initialement opté pour un GridPane, qui me semblait logique pour une structure en grille, mais après quelques tests, je me suis retrouvé limité par son manque de liberté. J’ai donc décidé de basculer sur un Pane simple, ce qui m’a permis d’utiliser un positionnement absolu des éléments dans un repère X, Y. Le tout est encapsulé dans un ScrollPane pour simplifier la navigation, quel que soit le nombre ou la durée des tâches.

La seconde étape majeure concernait l'algorithme d'affichage (dessinerTacheRecursive) pour gérer la hiérarchie des tâches. Mon algorithme procède ainsi :

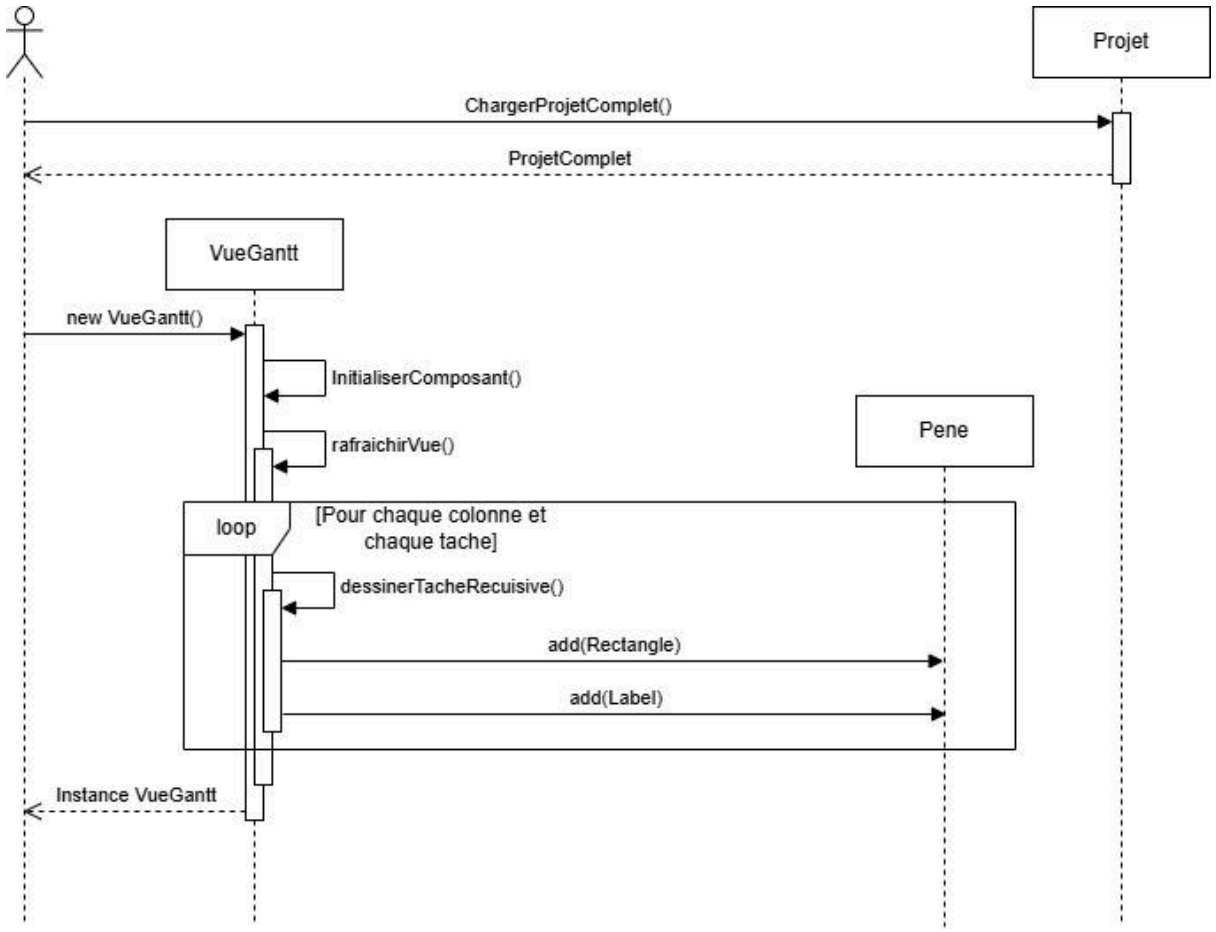
Je dessine le rectangle de la tâche courante à la ligne actuelle (indexLigne, initialisé à 1). La position X est calculée via la différence entre la date de début du projet (dateZero) et celle de la tâche, et la largeur correspond à sa dureeEstimee.

Je calcule la couleur de fond via la méthode genererCouleurParProfondeur. Celle-ci utilise la fonction interpolate() pour mélanger le bleu original avec du blanc selon un facteur croissant, ce qui permet d'éclaircir la teinte en fonction de la profondeur de la tâche (Tâche Mère > Sous-tâche > etc.).

J’incrémente indexLigne pour préparer l'affichage suivant.

Enfin, par récursivité, si la tâche est une tâche mère, je rappelle la méthode pour chacune de ses sous-tâches avant de passer à la suite. Cela me permet d’afficher la tâche mère et ses sous-tâches comme un même bloc qui s'enchaîne et ainsi faciliter la compréhension du diagramme.

Ce diagramme de Séquence représente comment j’ai imaginé le chargement de la vue Gantt durant l’itération 5 :



* Pane et non Pene

6. Manuel d'utilisation (IntelliJ)

Cette section détaille la procédure technique pour configurer l'environnement de développement, installer les dépendances et lancer l'application sur IntelliJ IDEA.

6.1 Pré Requis Techniques (Versions)

Pour garantir le bon fonctionnement de l'application, l'environnement de développement doit respecter les versions suivantes, définies dans notre fichier de configuration `pom.xml` :

- **IDE recommandé :** IntelliJ IDEA (Community ou Ultimate, version 2023.3 ou supérieure recommandée pour le support Java 21).
- **Java Development Kit (JDK) : Version 21** (LTS). Le projet est configuré avec `<source>21</source>` et `<target>21</target>`.
- **Gestionnaire de projet :** Maven (intégré à IntelliJ).
- **Base de Données :** MySQL Server (version 8.0 ou supérieure).

6.2 Gestion des Dépendances

Le projet utilise Maven pour la gestion automatique des librairies. Les dépendances principales incluses dans le fichier `pom.xml` sont :

- **Interface Graphique (JavaFX) :**
  - `javafx-controls` (21.0.6) : Composants graphiques de base.
  - `javafx-fxml` (21.0.6) : Chargement des vues FXML.
  - `controlsfx` (11.2.1) et `bootstrapfx-core` (0.4.0) : Composants UI avancés et stylisation.
  - `formsfx-core` (11.6.0) : Gestion simplifiée des formulaires.
- **Base de Données :**
  - `mysql-connector-j` (9.3.0) : Driver JDBC pour MySQL.
  - `HikariCP` (5.1.0) : Pool de connexions pour optimiser les accès à la base.
- **Tests Unitaires :** `junit-jupiter` (5.12.1).

6.3 Installation et Configuration de la Base de Données

Avant de lancer l'application, la base de données doit être accessible localement.

1. **Lancer WAMP/XAMPP/MAMP** ou votre service MySQL local.
2. **Configuration par défaut :**
   - L'application est configurée pour se connecter avec les identifiants suivants (visibles dans `src/main/java/application/DAO/DBConnection.java`) :
     - **Host :** `127.0.0.1` (Localhost)
     - **Port :** `3306`
     - **Base de données :** `sae_taches`
     - **Utilisateur :** `root`
     - **Mot de passe :** `""` (vide)
3. **Création de la BDD :** Exécutez le script SQL fourni (situé à la racine) pour créer la base `sae_taches` et les tables nécessaires.
   - *Note : Si votre configuration MySQL diffère (mot de passe non vide), modifiez les constantes `userName` et `password` dans la classe `DBConnection.java`.*

6.4 Procédure de Lancement sur IntelliJ

1. **Exécution :**
   - Dans l'arborescence, naviguez vers `src/main/java/application/MainApp.java`.
   - Faites un clic droit sur le fichier et sélectionnez `Run 'MainApp.main()'`.

6.5 Guide d'Accès aux Fonctionnalités

Dans la BD fourni dans le dépôt, il y a déjà des projet de test que vous pouvez utiliser (Projet Test et Projet Exemple Arche)

Une fois l'application lancée, la navigation s'effectue comme suit (basé sur la logique définie dans `MainApp.java`) :

1. **Tableau de Bord (Dashboard) :** L'application s'ouvre sur la `VueDashboard`. C'est le point central pour :
    - **Créer** un nouveau projet (bouton "Ajouter").
    - **Ouvrir** un projet existant (sélectionner un projet puis cliquer sur "Ouvrir").
    - **Supprimer** un projet.
2. **Vue Kanban :** À l'ouverture d'un projet, vous arrivez sur la vue par défaut (`VueKanban`).
    - Gérez vos tâches par colonnes.
    - Utilisez le menu latéral (Sidebar) pour naviguer vers les autres vues.
3. **Changement de Vues :** Depuis le Kanban, vous pouvez accéder aux autres modes de visualisation via les boutons dédiés :
    - **"Vue Liste"** : Affichage chronologique ou tabulaire.
    - **"Vue Gantt"** : Visualisation temporelle des tâches.
    - **"Voir Archives"** : Consultation des éléments archivés.
4. **Navigation :** Un bouton `<- Dashboard` présent dans l'en-tête de chaque vue permet de revenir à tout moment à l'accueil pour changer de projet.