ASSAL Hugo
BOURDON Marin
DELCOURT Mathias
GRANDJEAN Mathéo
KONÉ Ismaël

Rendu final SAE3.01
Développement d'applications

12/01/2024

Sommaire :

1. Introduction et rappel du sujet de la SAE
2. Liste des fonctionnalités réalisées et auteur de ces fonctionnalités
3. Modification comparé à l'étude préalable
4. Présentation des éléments qui nous rendent fiers par étudiant
5. Diagramme de classe final et diagrammes par patrons de conception
6. Conclusion de la SAE

1. Introduction et rappel du sujet de la SAE

Ce compte rendu final conclut les 6 itérations de la partie qualité de développement de la SAE 3.01. Pour rappeler le sujet, l'objectif de cette SAE était de "développer un logiciel permettant de gérer des tâches personnelles pouvant être dépendantes les unes des autres et de visualiser l'organisation de la progression de leur réalisation dans une interface graphique restant simple d'utilisation et ergonomique".

Le sujet était assez libre puisqu'il nous permettait d'implémenter les fonctionnalités que nous souhaitions dans la limite du réalisable, sans imposer de conditions particulières, si ce n'est l'implémentation de quelques fonctionnalités obligatoires :

- **créer des tâches** qui peuvent dépendre d'autres tâches, par exemple une tâche inclue dans une autre tâche ou encore une tâche ne pouvant démarrer que si une autre est terminée ;
- **archiver des tâches** une fois celles-ci réalisées ;
- **visualiser l'ensemble des tâches** sous forme d'un bureau avec des colonnes regroupant des tâches, une tâche pouvant être glissée déposée d'une colonne à une autre. La description d'une tâche (et de ses sous-tâches) pourra être visualisée après sélection avec la souris ;
- **visualiser l'ensemble des tâches** sous forme de listes et de sous-listes si le déploiement des sous-tâches est demandé ;
- **générer le diagramme de Gantt** d'une sélection de tâches afin de visualiser la chronologie de la réalisation de celles-ci.

Où encore l'utilisation de JavaFX et du patron MVC pour la réalisation de l'application.
Désormais , nous allons nous intéresser aux fonctionnalités que nous avons réalisées dans l'application.

2. Liste des fonctionnalités réalisées et auteurs de ces fonctionnalités

Dans cette partie du rapport, nous allons lister les fonctionnalités réalisées dans notre application :

1. **Créer des tâches**, dans l'application on doit pouvoir créer une tâche directement dans une liste. Cette tâche et la liste dans laquelle elle appartient s'ajoutent dans la BD.
   L'implémentation de cette fonctionnalité a été réalisée par Hugo Assal et l'insertion dans la base de données par Marin Bourdon et Mathias Delcourt.

2. **Modifier une tâche**, en faisant un clic droit sur une tâche, une boîte de dialogue s'ouvre et permet différentes options, tels que, la modification du titre, du contenu, de la date de début et de fin de la tâche, ajouter des tâches dépendantes parmi les autres tâches de la liste et enfin archiver ou non la tâche. Les modifications effectuées sur la tâche se mettent à jour dans la BD.
   L'implémentation de cette fonctionnalité a été réalisée par Mathias Delcourt et Hugo Assal et la modification dans la base de données par Marin Bourdon et Mathias Delcourt.

3. **Archiver des tâches**, en bas du menu de modification d'une tâche (lorsque l'on fait un clic droit sur une tâche) un bouton "Archiver" est présent, lorsqu'on clic dessus, la tâche est archivée et disparaît donc de l'affichage, on retrouve la tâche archivée dans les paramètres de l'application. Le champ correspondant à l'archivage d'une tâche se met à jour pour la tâche en question.
   L'implémentation de cette fonctionnalité a été réalisée par Hugo Assal et la modification de la BD par Mathias Delcourt.

4. **Renvoyer sur le tableau une tâche archivée**, lorsqu'on fait un clic droit sur une tâche archivée, une boîte de dialogue s'ouvre et un bouton "Renvoyer sur le tableau" permet de renvoyer la tâche dans le tableau et dans la liste à laquelle la tâche appartenait avant d'être archivée. Le champ correspondant à l'archivage d'une tâche se met à jour pour la tâche en question.
   L'implémentation de cette fonctionnalité a été réalisée par Hugo Assal.

5. **Ajouter des tâches dépendantes**, lorsqu'on fait un clic droit sur une tâche pour la modifier et qu'on se rend en bas de la boîte de la dialogue, une ComboBox composé des autres tâches de la même liste pour les transformer en tant que tâches dépendantes. Rendre une tâche dépendante d'une autre, modifie sa date de début à la fin de la tâche dont elle dépend + 1. La table *ESTSOUSTACHE* se met à jour, avec l'id des tâches dépendantes.
   L'implémentation de cette fonctionnalité a été réalisée par Hugo Assal et l'insertion dans la base de données par Marin Bourdon et Mathias Delcourt.

6. **Visualiser l'ensemble des tâches sous forme d'un bureau avec des colonnes** regroupant des tâches, une tâche pouvant être glissée déposée d'une colonne à une

autre. La description d'une tâche (et de ses sous-tâches) pourra être visualisée après clic droit avec la souris.
L'implémentation de cette fonctionnalité a été réalisé par Hugo Assal, Mathéo Grandjean, Marin Bourdon.

7. **Attribuer une date limite à une tâche**, en modifiant une tâche, on peut lui attribuer une date de début et une date de fin, utile notamment pour le diagramme de Gantt. Ces modifications sont aussi apportées dans la base de données.
   L'implémentation de cette fonctionnalité a été réalisée par Mathias Delcourt et Hugo Assal et l'insertion et la modification dans la base de données par Marin Bourdon et Mathias Delcourt.

8. **Créer une liste**, une fois le tableau créé, un bouton "Ajouter liste". Lorsqu'on clique dessus, un dialogue permettant de rentrer le titre de la liste s'ouvre, lorsqu'on l'a saisi et qu'on appuie sur entrée, alors la liste avec le titre choisi se crée dans le tableau. Les listes se créent les une à la suite des autres, il est donc possible de "scroller" pour visualiser l'ensemble des listes. La liste se rajoute dans la BD avec l'insertion des lignes dans les tables correspondantes.
   L'implémentation de cette fonctionnalité a été réalisée par Hugo Assal et Ismaël Koné, et l'insertion dans la base de données par Marin Bourdon et Mathias Delcourt.

9. **Modifier une liste**, effectuer un clic droit sur une liste ouvre une boîte de dialogue similaire à celle de la création avec en plus un bouton "Supprimer", et permet de mettre à jour son titre, à la fois dans l'affichage mais aussi dans la BD.
   L'implémentation de cette fonctionnalité a été réalisée par Marin Bourdon et la modification dans la base de données par Marin Bourdon et Mathias Delcourt.

10. **Supprimer une liste**, effectuer un clic droit sur une liste ouvre une boîte de dialogue avec un TextField pour modifier le titre de la liste mais aussi un bouton "Supprimer" qui permet de supprimer la liste du tableau mais aussi de la BD.
    L'implémentation de cette fonctionnalité a été réalisée par Marin Bourdon et Mathias Delcourt et la suppression dans la base de données par Marin Bourdon et Mathias Delcourt.

11. **Créer un tableau**, lorsqu'on ouvre l'application, sur le côté gauche de l'application, on retrouve l'indication "Tableau" avec à droite un bouton. Lorsqu'on clique dessus, un dialogue permettant de rentrer le titre du tableau s'ouvre, lorsqu'on l'a saisi et qu'on appuie sur entrée, alors le tableau avec le titre choisi s'ajoute en dessous du bouton de création de tableau. Le tableau se rajoute dans la BD avec l'insertion des lignes dans les tables correspondantes.
    L'implémentation de cette fonctionnalité a été réalisée par Mathéo Grandjean et l'insertion dans la base de données par Marin Bourdon et Mathias Delcourt.

12. **Modifier un tableau**, effectuer un clic droit sur le bouton d'un tableau ouvre une boîte de dialogue similaire à celle de la création avec en plus un bouton "Supprimer", et permet de mettre à jour son titre, à la fois dans l'affichage mais aussi dans la BD.

L'implémentation de cette fonctionnalité a été réalisée par Mathéo Grandjean et la mise à jour dans la BD par Mathias Delcourt.

13. **Supprimer un tableau**, effectuer un clic droit sur un tableau ouvre une boîte de dialogue avec un TextField pour modifier le titre du tableau mais aussi un bouton "Supprimer" qui permet de supprimer le tableau de l'application mais aussi de la BD.
    L'implémentation de cette fonctionnalité a été réalisée par Mathéo Grandjean et la suppression dans la BD par Mathias Delcourt.

14. **Sélectionner un tableau**, lorsqu'on clic sur un des boutons correspondant à un tableau dans le côté gauche de l'application alors on change l'application par le tableau choisi avec l'affichage des listes et des tâches du tableau correspondant.
    L'implémentation de cette fonctionnalité a été réalisée par Mathéo Grandjean.

15. **Générer le diagramme de Gantt**, lorsqu'on clic gauche sur plusieurs tâches, qu'on les rend dépendantes les unes des autres alors on peut cliquer sur le bouton Diagramme de Gantt, une nouvelle fenêtre non modifiable s'ouvre avec les tâches sélectionnées présenter sous forme de diagramme de Gantt.
    L'implémentation de cette fonctionnalité a été réalisée par Hugo Assal.

16. **Création utilisateur**, lorsqu'on lance l'application, un écran permettant la connexion et l'inscription se présente. Si on clique sur le bouton d'inscription, un formulaire s'ouvre, une fois le formulaire rempli et respectant les conditions pour s'inscrire (@ dans l'email, Mot de passe robuste, etc) alors on appuie sur le bouton valider, qui ouvre alors l'application avec le compte créé et enregistre le nouvel utilisateur (en hachant son MDP) dans la BD.
    L'interface graphique d'inscription a été réalisée par Mathéo Grandjean et l'insertion dans la base de données par Marin Bourdon.

17. **Se connecter**, lorsqu'on lance l'application, un écran permettant la connexion s'ouvre, si le compte existe, alors l'application s'ouvrira après avoir rempli les informations de connexion. Si les informations renseignées sont incorrectes alors un message d'erreur s'affiche.
    L'interface graphique a été réalisée par Mathéo Grandjean et l'insertion dans la base de données par Marin Bourdon.

18. **Mode sombre**, lorsqu'on rentre dans les paramètres de l'application. Un RadioButton avec écrit "Mode sombre" à sa gauche s'affiche. Lorsqu'on clique sur ce bouton alors l'application passe en mode sombre. Le mode sombre peut être retiré à la guise de l'utilisateur. Si on relance l'application après l'avoir arrêté en mode sombre, elle se rouvre alors en mode clair.
    La fonctionnalité a été implémentée par Mathéo Grandjean.

19. **Ajouter une section logs**, dans les paramètres, on retrouve les logs de créations, de modification et de suppression qui s'enregistrent au fur et à mesure dans l'application.

La fonctionnalité a été réalisée par Mathéo Grandjean.

20. **Choisir des templates**, il y a une comboBox qui est situé en haut à gauche de l'application à côté du logo, elle permet de choisir parmis une liste de modèles qui quand on clique sur l'un d'eux, un tableau est créé avec des listes et des tâches déjà placées inspiré de modèle existant et/ou utilisé en entreprises (exemple : modèle Kanban).
    La fonctionnalité a été réalisée par Mathéo Grandjean et Hugo Assal

21. **Déplacer une tâche en glisser-déposer**, on peut maintenir le clic sur une tâche et la déplacer d'une liste à une autre.
    La fonctionnalité à été réalisée par Ismaël Koné.

3. Modification apportée à l'étude préalable

Maintenant, nous allons nous intéresser à la partie étude préalable de cette SAE. Dans cette partie, nous chercherons à comparer le résultat du projet avec les diagrammes et l'étude préalable de la partie analyse de la SAE.
On observe ci-dessous le diagramme de classe réalisé lors de l'étude préalable :

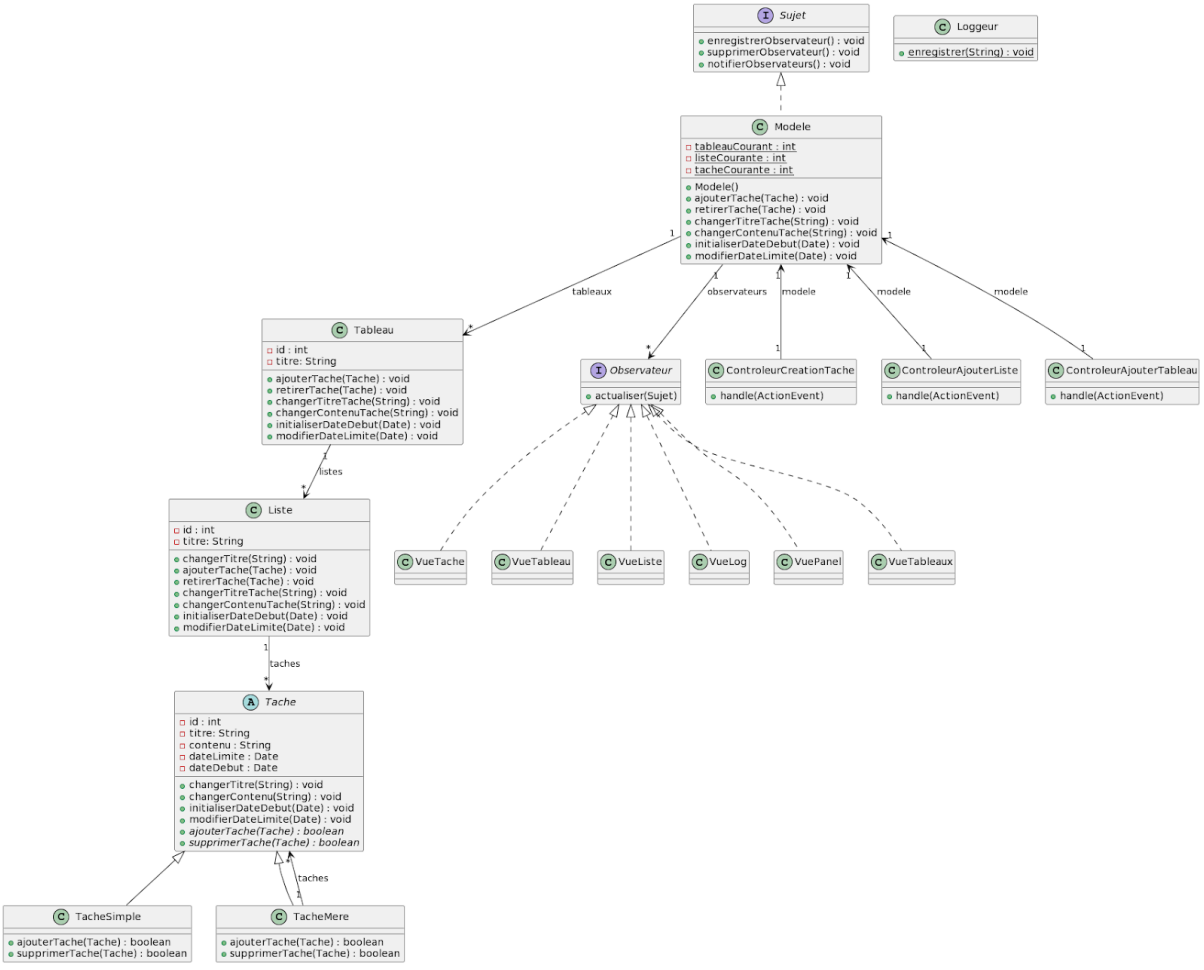

Image 1 : Diagramme de classe réalisé lors de l'étude préalable

Ce diagramme a majoritairement été respecté, la forme reste la même sur notre diagramme de classe final, à la simple différence que les vues et contrôleurs sont beaucoup plus variés qu'à l'origine, car on ne se rendait pas forcément compte de toutes les classes que ces actions allaient nous nécessiter.

Désormais, nous allons nous pencher sur la description textuelle réalisée durant la phase d'analyse, c'est-à-dire celle qui couvre le cas "Choix d'un template" :

Préconditions
Accès à l'application Tabl'o ; disponibilité des templates différents ; compte utilisateur correct ; connexion internet stable.
Postconditions
Si les templates sont disponibles, l'utilisateur à un accès total au template sélectionné qui devient un de ses tableaux personnels selon le modèle choisi.

Déroulement normal
(1) L'utilisateur accède à l'application Tabl'o, (2) il se connecte à son espace personnel en utilisant ses identifiants, (3) le système vérifie l'authenticité des informations d'identification. (4) Une fois connecté, l'utilisateur peut accéder à son espace de travail et aux tableaux sur son compte. (5) L'utilisateur sélectionne un
template qui correspond à ses besoins et ses préférences. (6) L'application actualise le profil de l'utilisateur avec le template ajouté comme un tableau à part entière de l'utilisateur.

Variantes
(A) Le template ne marche pas. L'utilisateur peut redémarrer l'application ou choisir un autre template.
(B) Si un template est déjà présent, l'application l'ajoute en nouveaux tableaux.

Contraintes non fonctionnelles
(A) Sécurité : Les informations personnelles doivent être sécurisées et protégées conformément aux normes de sécurité en ligne.

Image 2 : Description textuelle du choix d'un template

Pour cette description, on constate que le fonctionnement correspondant à cette fonctionnalité est exactement le même dans l'application, si ce n'est que la variante (B) n'est pas tout à fait juste, car si un template est déjà présent, on ne peut pas en créer un nouveau notamment car nous avons choisi d'implémenter Singleton et pour éviter tout doublon et erreur dans l'application.

Enfin, pour finir cette comparaison entre l'étude préalable et l'application terminée, nous allons comparer le diagramme d'état transition de l'objet Tâche et son fonctionnement dans l'application :

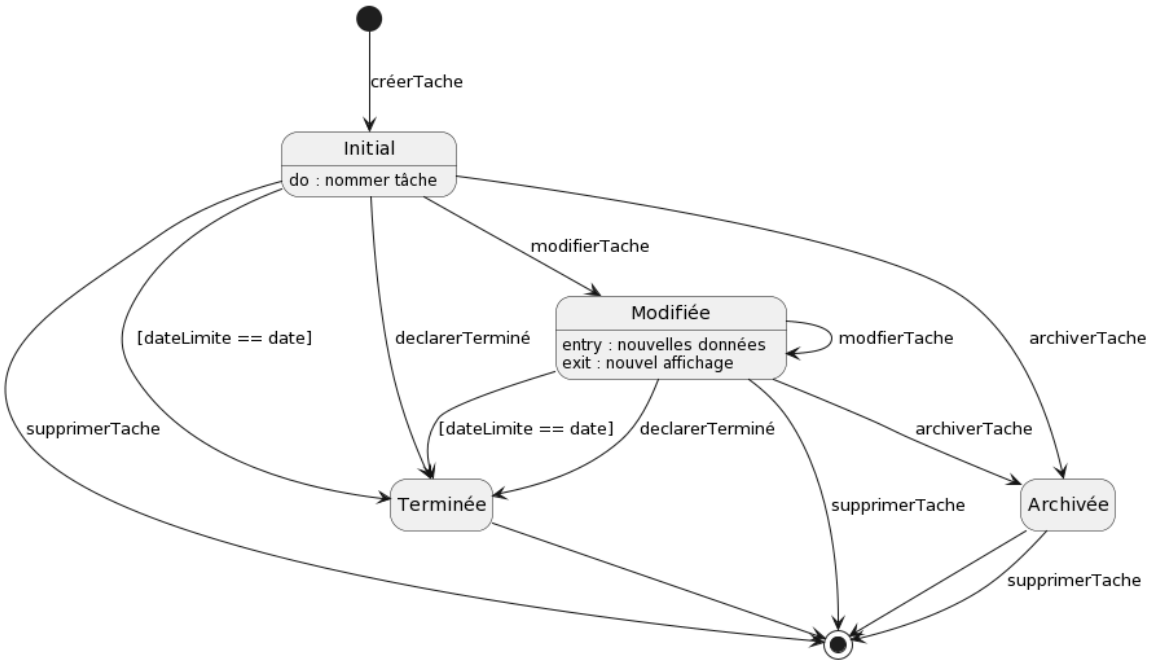

Image 3 : Diagramme d'état de l'objet central de l'application

L'application intègre fidèlement les fonctionnalités décrites dans le diagramme, à l'exception de la suppression de tâche qui n'a pas été implémentée sur l'interface graphique par souci. Toutefois, une méthode de suppression de tâche à été décrite, celle-ci est utilisée pour la fonctionnalité de déplacement de tâche.

4. Présentation des éléments dont les membres du groupe sont fières

Dans cette partie du compte rendu, chaque membre du groupe va présenter personnellement, un élément du projet, qui le rend fier et expliquer pourquoi ce choix :

Hugo : L'élément original dont je suis fier est l'utilisation du patron Stratégie pour les templates. Cet élément a été rajouté lors de la fin de la SAE l'avant-dernier jour, car je me suis rendu compte que le code de Mathéo ajoutait beaucoup trop de code dans la classe Modèle qui était déjà suffisamment lourde comme ça. En plus de cela, les principes SOLID n'étaient pas respectés, certes avec MVC on ne peut pas réellement respecter les principes SOLID, mais ajouter la gestion des templates aurait encore plus trahit le Single Responsability Principle. Contrairement à maintenant car le Modèle n'est pas directement modifié mais il est étendu par l'attribut de type TemplateStrategy. Cette fonctionnalité m'a rendu particulièrement fière car je pense que c'est ce genre de conception qui était attendu dans ce projet.

Ismaël : Je suis particulièrement fier de la fonctionnalité de déplacement de tâches par glisser-déposer dans mon logiciel de gestion des tâches. Bien qu'elle puisse être améliorée, elle demeure un élément clé. La principale difficulté est qu'il m'a fallu revoir la méthode d'ajout de tâche et créer une méthode de suppression de tâche d'une liste donnée afin de réaliser au mieux cette fonctionnalité. De plus, accomplir cette tâche a impliqué la mise en place d'un système de sélection d'une tâche lorsque celle-ci est attrapée à la souris par l'utilisateur. Ce processus a demandé une réflexion approfondie pour assurer une manipulation fluide et intuitive des tâches dans l'application.

Marin : La fonctionnalité que j'ai le plus aimé développer est l'implémentation d'une base de données pour sauvegarder toutes les données de l'application afin de pouvoir sauver son travail et le modifier d'une connexion à une autre. Je trouve que cette fonctionnalité donne du sens à l'application, dans le sens ou cela permet réellement de suivre l'organisation d'un projet par exemple, avec l'ensemble des tableaux, listes et tâches qui seront sauvegardés dans la bd. L'utilisateur aura juste à se reconnecter avec ses identifiants déterminés lors de la création de son compte, pour pouvoir récupérer son travail et le modifier.

Mathéo : L'élément que j'ai trouvé le plus original à faire, était la fonctionnalité de choix du mode sombre car ceci était purement visuel. Son implémentation n'était pas la plus complexe à réaliser mais le mode est quelque chose que nous avons l'habitude de voir sur des applications que nous utilisons tous les jours et qui à mon sens apporte une réel plus value au quotidien. Mais les fonctionnalités les plus marquantes, selon moi, quant à la réalisation était la modification et la suppression d'un tableau même si elles ne sont pas forcément des plus « originales » mais plutôt obligatoires. Au début de leurs conception, je n'avais pas utilisé la bonne technique d'implémentation comme on peut le voir avec mon premier diagramme de séquence ci-dessous :

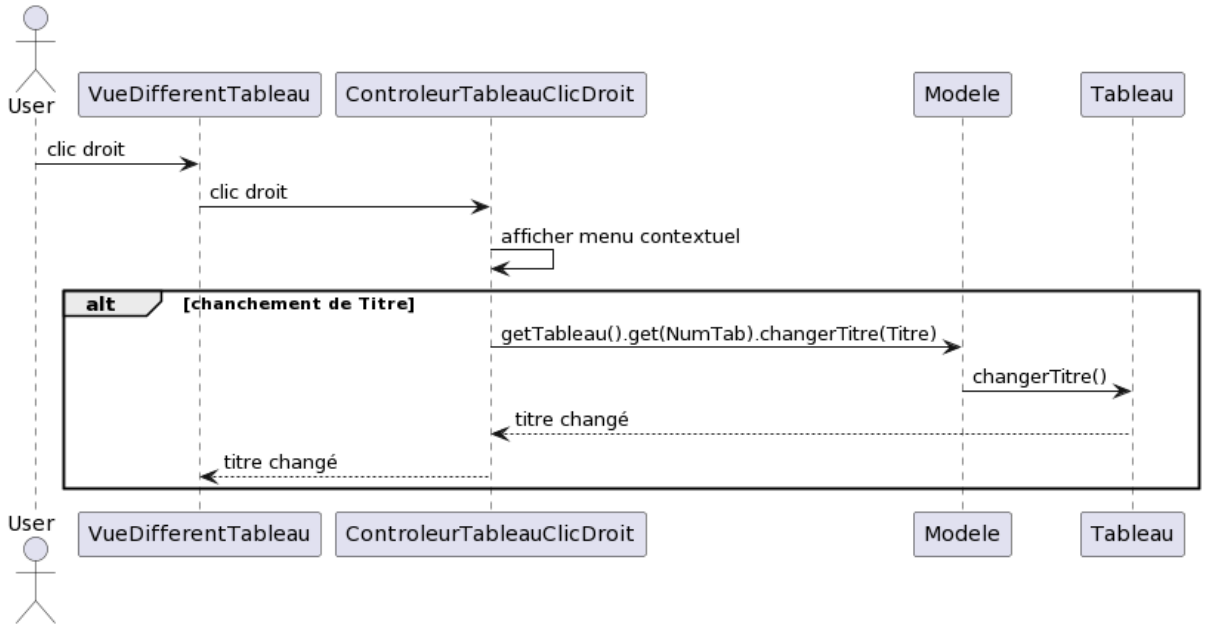

Image 4 : Diagramme de séquence de la fonctionnalité modifier un tableau version 1

Ainsi j'ai dû recommencer ces deux fonctionnalités et j'ai pu comprendre mes erreurs et réparer ce problème, pour finalement obtenir un diagramme de séquence comme ci-dessous :

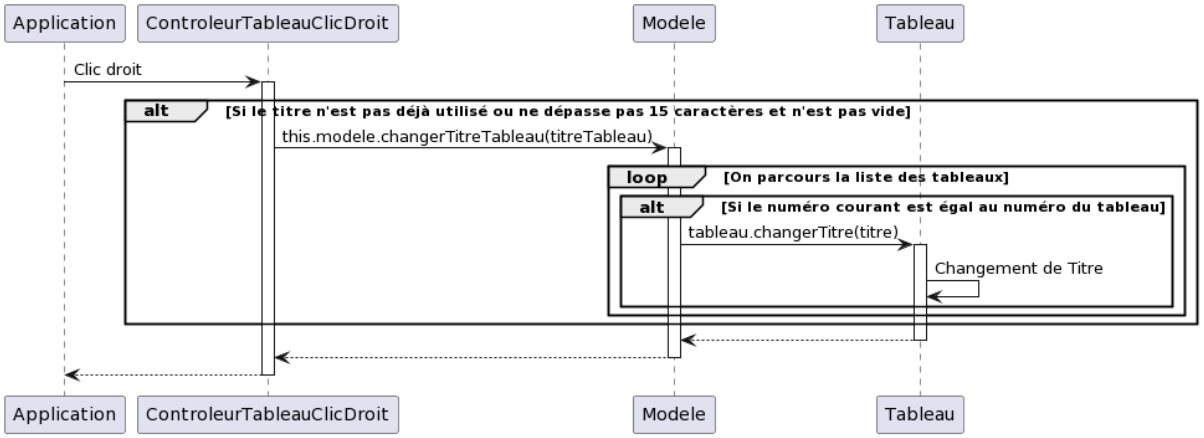

Image 5 : Diagramme de séquence de la fonctionnalité modifier un tableau version 2

Maintenant avec ces modifications, les fonctionnalités supprimer et modifier sans aucun problème le tableau courant.

Mathias : L'élément qui personnellement me rend fier et l'implémentation de tout le côté base de données de l'application. Ce travail n'est pas une simple implémentation du patron ActiveRecord, le travail était bien plus profond, il fallait penser à l'ensemble du travail. Cet ensemble correspond à : l'initialisation de la base de données, l'initialisation des tables utiles au bon fonctionnement notamment avec l'utilisation d'une deuxième classe Main, l'implémentation du patron ActiveRecord pour les classes adéquates et la phase de tests. Cependant pour les tables d'associations, des questions de conceptions se posaient, notamment la création de nouvelle classe pour les tables associatives ? l'écriture directement dans le code des requêtes SQL ? Par soucis de temps, d'un point de vue itération et conception, nous avons finalement opté pour l'écriture dans le code des requêtes SQL. La réalisation d'une base de données et la programmation JAVA étant 2 choses que j'apprécie personnellement, travailler sur cette fonctionnalité où plutôt sur cet élément était un réel plaisir malgré les prises de têtes surtout de penser que c'est une réelle plus value sur notre projet puisque la grande majorité des autres groupes ne propose pas cette fonction dans leur projet.

5. Diagramme de classe final et diagrammes par patrons de conception

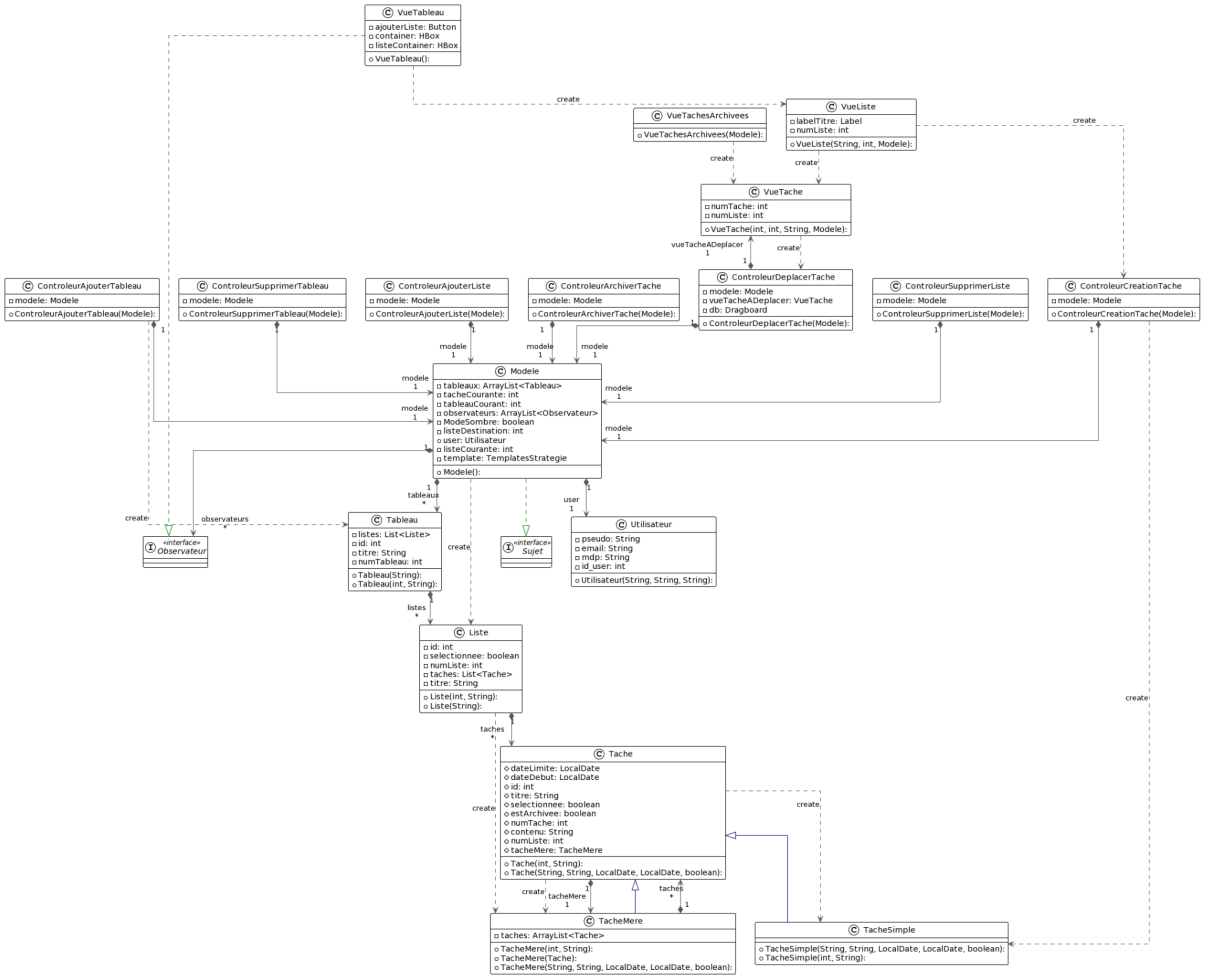

Image 6 : Diagramme de classe final de l'architecture globale de l'application

Description : Ce diagramme représente l'architecture globale de l'application. En effet, ce diagramme est fidèle à celui réalisé lors de l'étude préalable. Les patrons de conception figurants sont : Observateur; MVC; et Composite pour la représentation des dépendances entre les tâches.

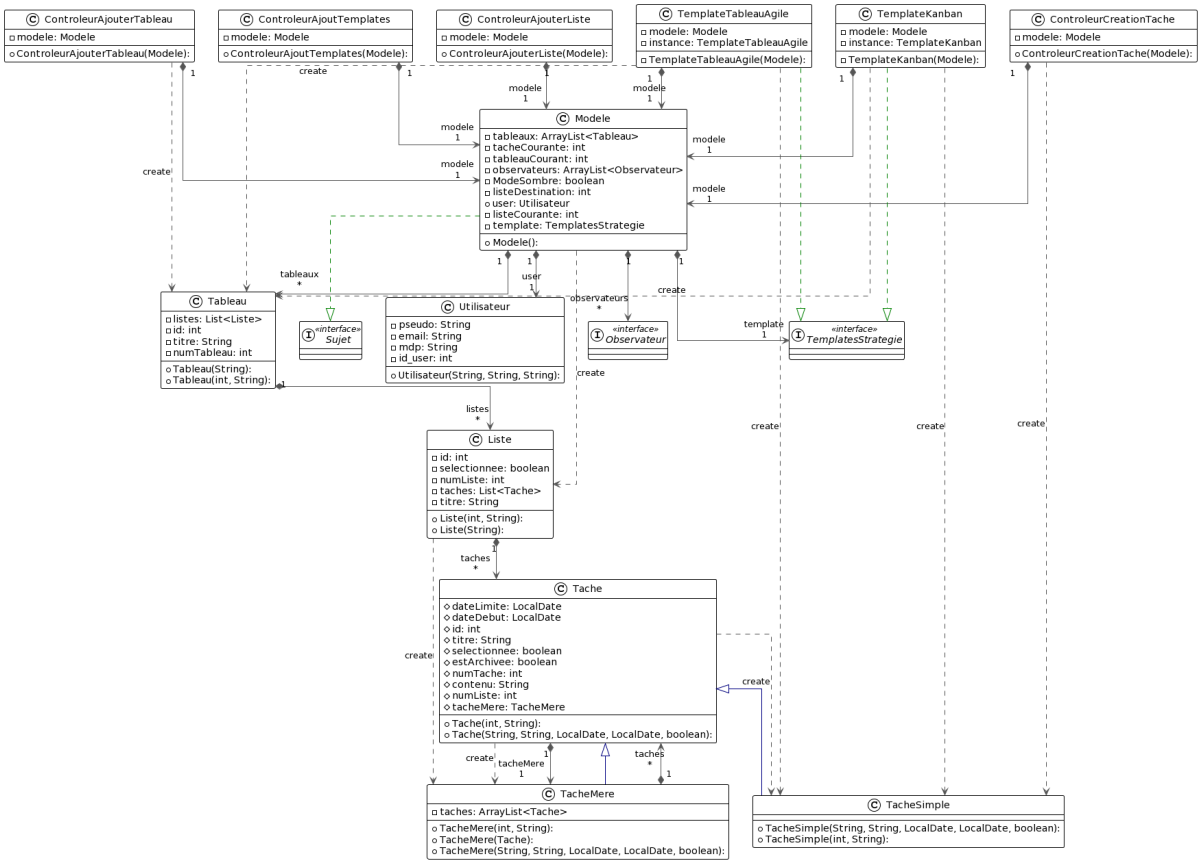

Image 7 : Diagramme de classe représentant l'implémentation de templates

Description : Les templates sont des tableaux avec des listes et des tâches prédéfinies. En effet, le patron de conception utilisé pour la représentation des templates est Stratégie, il nous a notamment permis d'éviter d'ajouter un grand nombre de lignes dans le modèle.

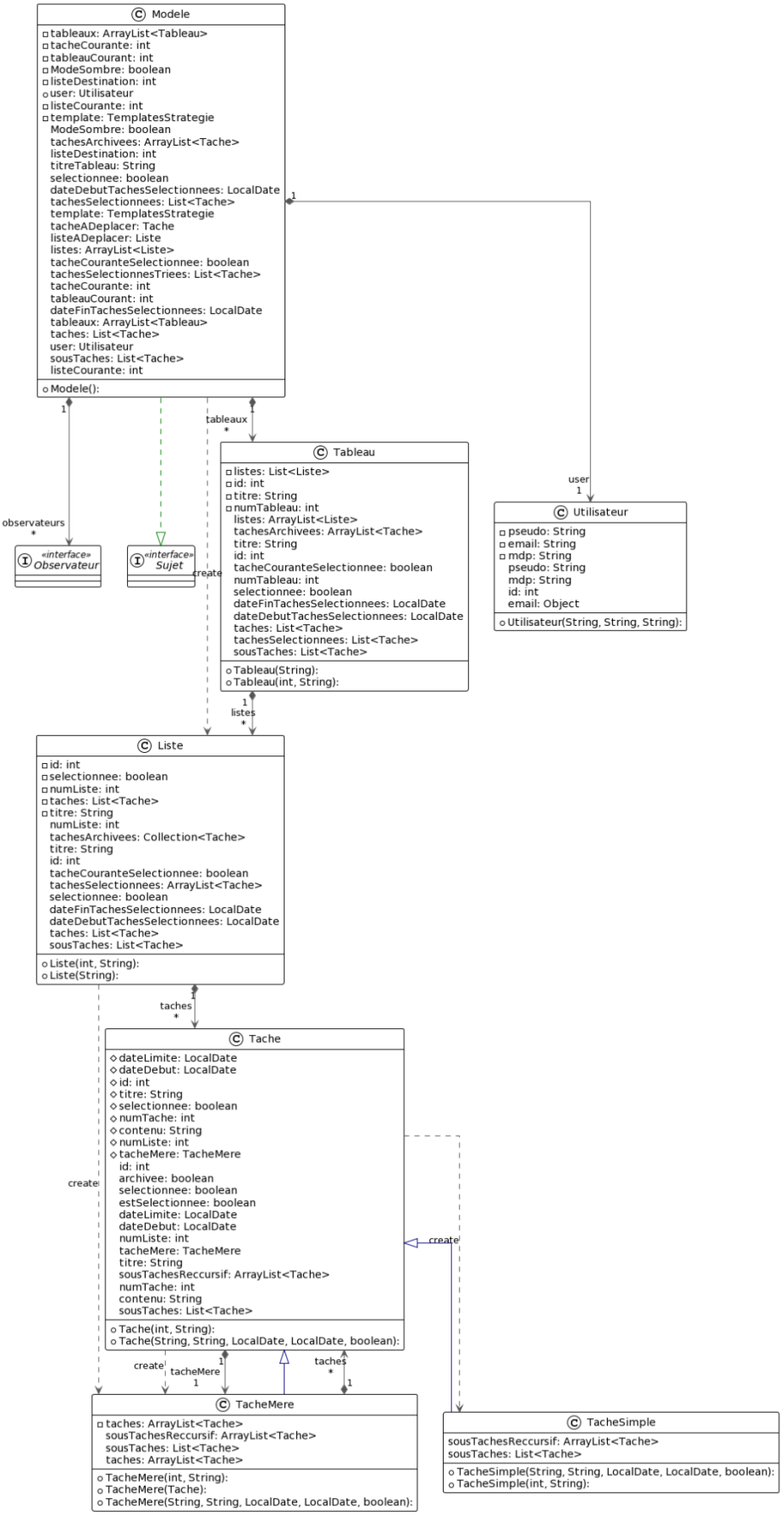

Image 8 : Diagramme représentant l'implémentation de la base de données

Description : Ce diagramme représente les classes comportant les méthodes nécessaire à l'implémentation du patron de conception ActiveRecord

6. Conclusion sur la SAE

En conclusion, le parcours que nous avons suivi lors du développement de l'application dans le cadre de SAE 3.01 a vraiment été formateur pour notre équipe. Bien que nous ayons réussi à créer un logiciel de gestion des tâches conforme à nos objectifs initiaux, nous avons réalisé rétrospectivement qu'une attention accrue aux phases de conception et de test aurait pu aider à atténuer certaines des complications rencontrées en cours de route. Ces défis, bien que formateurs, ont souligné l'importance d'une approche méthodique dès les phases préliminaires du projet. Cette expérience renforce notre engagement envers l'amélioration continue et l'application des meilleures pratiques dans nos futurs projets de développement de logiciels.