SAE31$_2$025

Chloé WIENCEK
Antoine DESESQUELLES

14 décembre 2025

Table des matières

1 Introduction

1.1 Présentation

Il s'agit d'un outil simple et ergonomique permettant à un utilisateur de gérer un ensemble de rappels : ajout, modification, suppression, description et catégorisation selon un thème ou un niveau d'importance. L'objectif principal était de concevoir une interface intuitive facilitant l'organisation personnelle, tout en respectant des exigences techniques comme la gestion d'une base de données, la validation des entrées, et l'intégration d'une interface graphique en Java. Ce projet mobilise plusieurs compétences fondamentales : conception logicielle, logique

de programmation, utilisation d'une base de données, et maîtrise de la bibliothèque Swing pour l'interface. Les objectifs fixés au début du projet étaient les suivants :

- Concevoir une interface simple, accessible et non intrusive.
- Permettre à l'utilisateur de créer des rappels avec un titre, une description, un thème et un ordre d'urgence.
- Imposer des règles de validation : titre limité à 50 caractères, description limitée à 4 lignes.
- Assurer une interaction fluide avec la base de données pour enregistrer, modifier et supprimer des rappels.
- Garantir une ergonomie optimale : la fenêtre principale apparaît en bas à droite de l'écran.

L'interface a été conçue pour être minimaliste et facilement manipulable. Elle inclut :
- une fenêtre principale affichant les rappels ;
- un formulaire d'ajout permettant de saisir titre, description, couleur ;
- des boutons d'action pour gérer les rappels.

Un soin particulier a été apporté à l'ergonomie. L'application se place automatiquement en bas à droite de l'écran, permettant d'être visible sans gêner l'utilisateur.

1.2 Aspect technique

L'application a été développée en Java en utilisant la bibliothèque Swing pour l'interface graphique. La persistance des données repose sur une base MariaDB. Un Makefile permet de compiler et d'exécuter le projet de manière automatisée.

Base de données : Maria DB
- **Nom** = wiencek
- **MDP** = wiencek1234

Execution projet
Commande pour executer :
- make
- make run

2 Présentation du projet

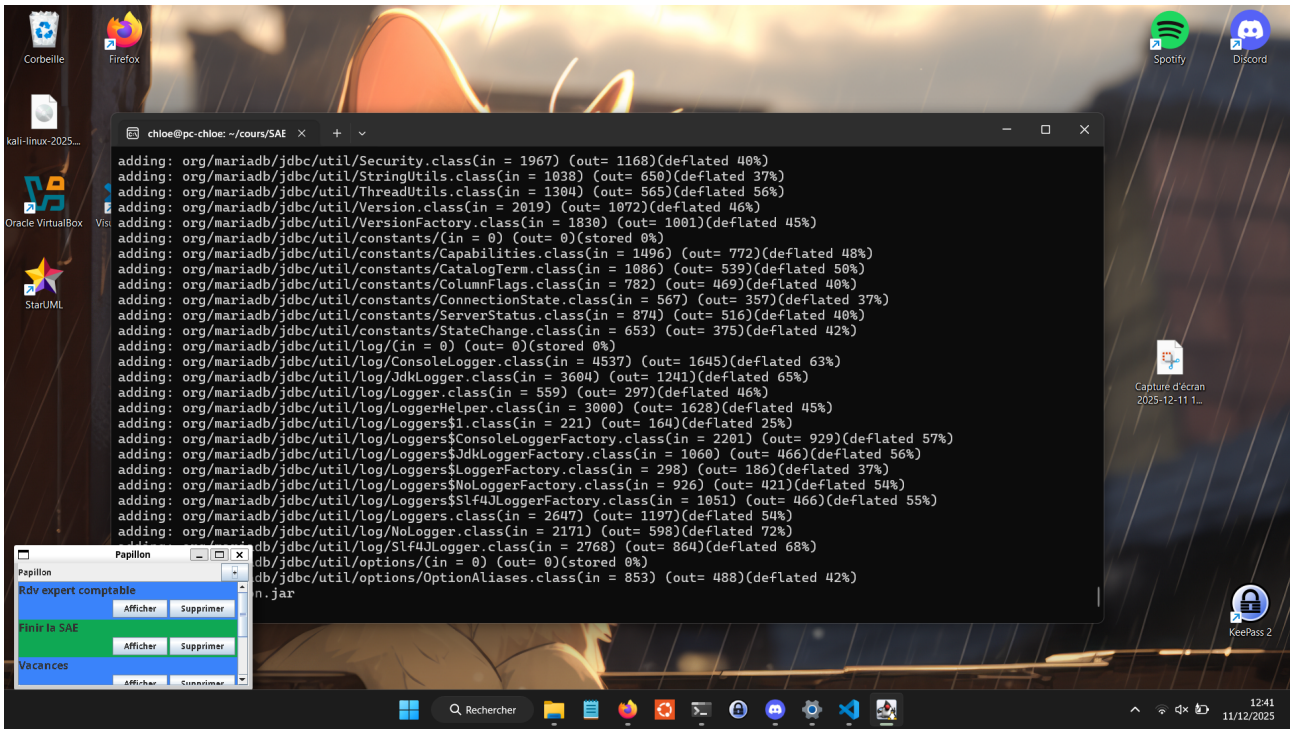
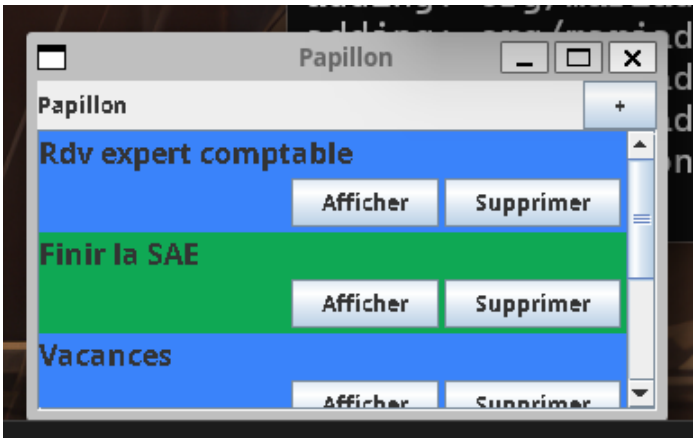


FIGURE 1 – Précédemment, nous parlions d'ergonomie. Voici l'apparence de l'application.



Le menu principal se divise en deux sections :

— La section en haut qui contient le titre de l'application et le bouton "+" pour ajouter un rappel

— Le reste de l'application est composé de panneaux : une boite par rappel.

Pour chaque "boite" de rappel, il y a deux boutons associés : Afficher supprimer

Pour comprendre le fil logique de l'application, voyons d'abord le bouton "+" (voir figure 2.1)

2.1 Ajouter un rappel

La fenêtre est aérée et simple. L'ajout d'un rappel comporte 4 catégories :
— Le titre est limité à 50 caractères maximum.
— La description est limitée à 4 lignes.
— Les thèmes sont choisis via un menu déroulant.
Ces contraintes sont vérifiées avant l'insertion dans la base de données. L'ensemble du fonctionnement repose sur une interaction fluide entre l'interface graphique et la BD.

2.2 Afficher un rappel

Cette fenêtre est celle qui s'affiche lorsque l'utilisateur appuie sur le bouton "Afficher" sur le menu principal. La page est simple elle comporte : titre, description et trois boutons. Le bouton permet d'envoyer l'utilisateur sur une une autre fenêtre ou il pourra modifier son rappel.

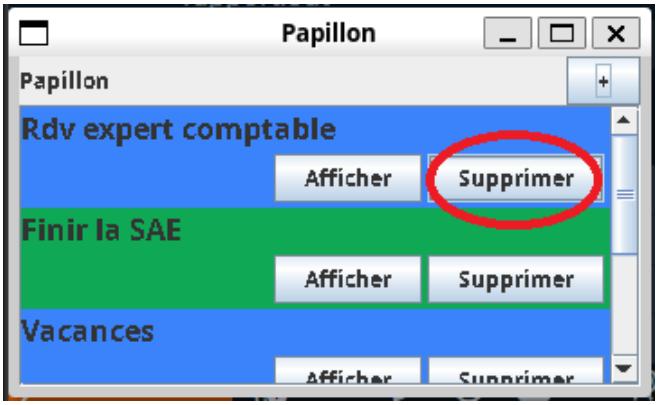
Nous avons également pris la décision de pouvoir régler le rang dans la fenêtre affichée pour ne pas surcharger la fenêtre principale d'informations. Nous pensons que c'est un choix optimisé qui permet à l'utilisateur de ne pas se perdre dans les fonctionalités. Il peut ainsi choisir de monter le rappel ou le descendre. Les extrémités (le rappel le plus haut et le plus bas) n'offrent bien évidemment qu'un seul choix. L'affichage se fait instantanément.

2.3 Modifier un rappel

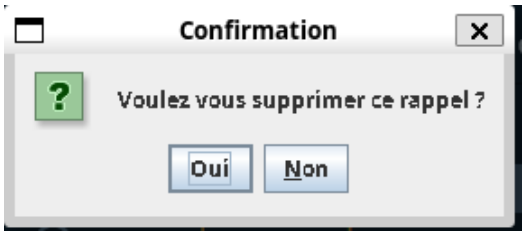
La page "modifier" permet à l'utilisateur de changer des informations qui sont enregistrées sur la base de données. Lorsque la fenêtre s'affiche, on a par défaut les informations qui ont été mises lors de l'ajout du rappel. On a donc la possibilité de modifier le titre, la description, le thème et le rang tout en respectant les contraintes déjà définies.

L'utilisateur appuie ensuite sur le bouton en bas pour enregistrer les modifications.

2.4 Supprimer un rappel



Message de confirmation en cas de suppression :



La suppression se fait directement via le menu. L'action est simple. L'appui sur le bouton et la validation du message de suppression suffisent à supprimer.

3 Une logique graphique

3.1 Loi de Hick

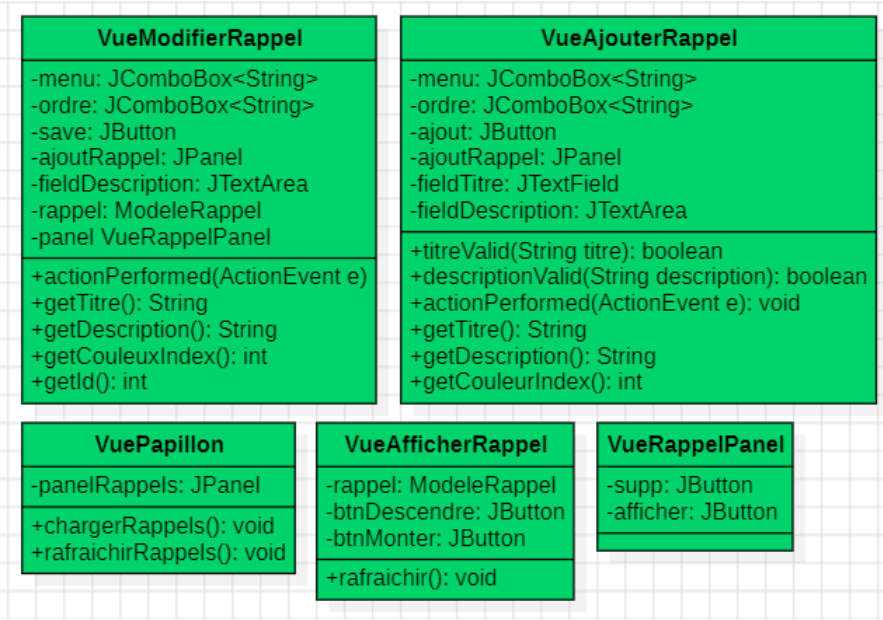
"Le temps pour prendre une décision augmente avec le nombre et la complexité des choix"
Nous avons appliqué cette logique en présentant le bouton "+" en haut à droite. Puis 2 autres boutons pour chaque rappel : Afficher et Supprimer. Ces boutons nous paraissent les deux plus pertinents à afficher sur le menu. Puis lorsque l'utilisateur appuie sur Afficher, un nouveau bouton Modifier apparaît.

3.2 Loi de Fitts

"Le temps pour atteindre une cible est une fonction de la distance à la cible et de la taille de la cible"
Il n'y a pas de dark patterns. Toutes les cibles (boutons dans notre cas) sont tous accessibles et intuitifs.

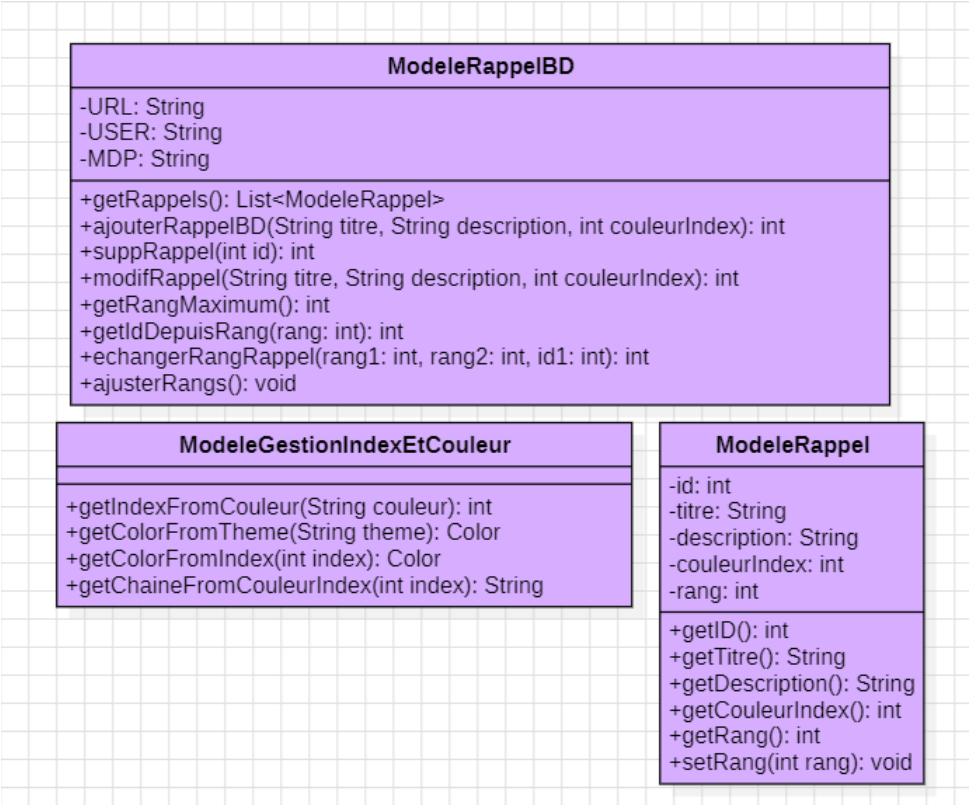
4 Les Diagrammes

Dans un premier temps, il est intéressant de comprendre la logique que nous avons appliquée pour l'organisation des classes :
Nous avons appliqué le modèle MVC.
Les vues :



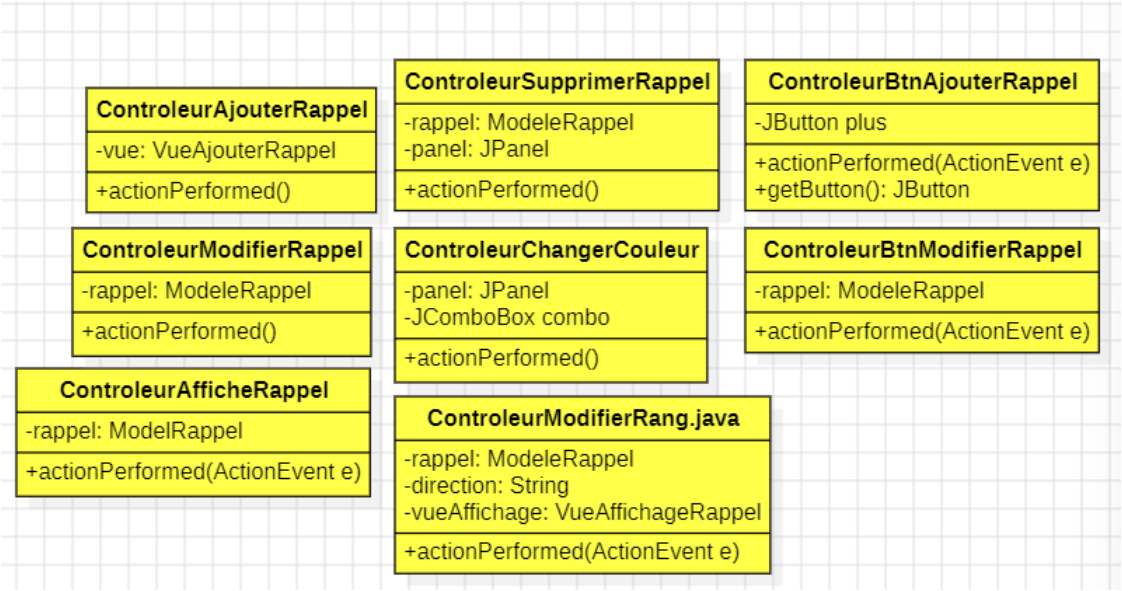
Représente l'interface utilisateur, avec des classes telles que VuePapillon, VueAjouterRappel, VueModifierRappel, VueRappelPanel, et VueAfficherRappel. Ces classes contiennent des composants Swing (JPanel, JButton, JComboBox, etc.) et des méthodes pour récupérer les données saisies ou rafraîchir l'affichage.

Les modèles :



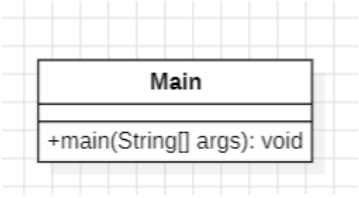
Représente les données métier. ModeleRappel encapsule les propriétés d'un rappel (titre, description, couleur, rang, etc.), tandis que ModeleRappelBD gère la persistance en base de données (ajout, suppression, modification, gestion des rangs). La classe IndexEtCouleur fournit des utilitaires pour la gestion des couleurs et des index d'importance.

Les controleurs :

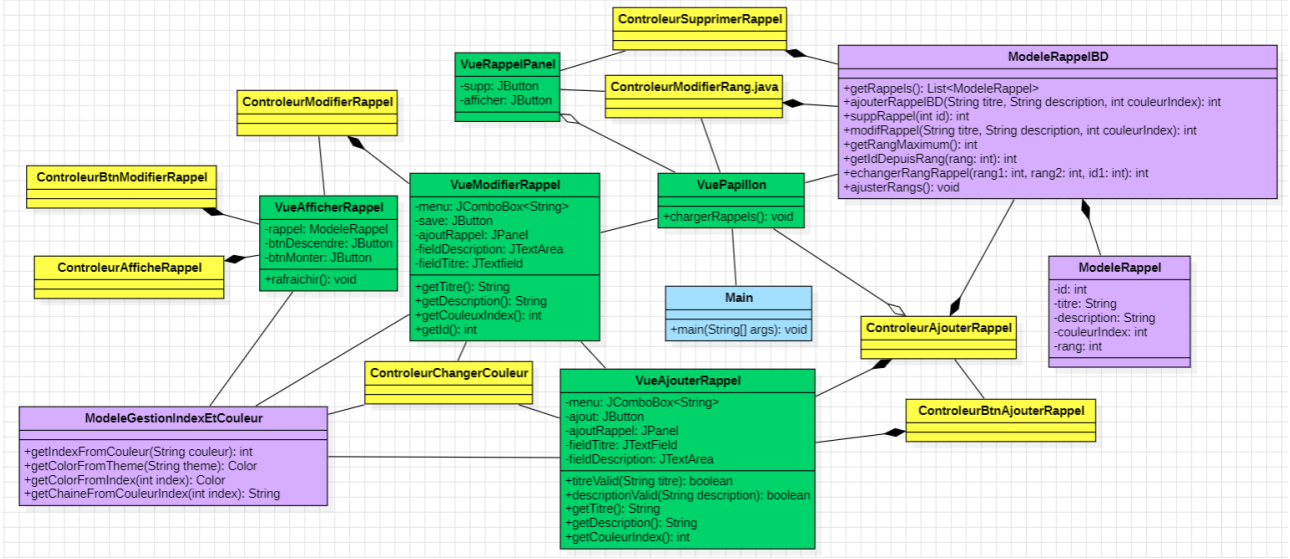


Gère les actions de l'utilisateur et la logique de traitement. Les classes comme ControleurAjouterRappel, ControleurModifierRappel, ControleurSupprimerRappel, ControleurChangerCouleur, ou ControleurModifierRang interceptent les événements (ActionEvent) et interagissent avec les modèles pour modifier l'état de l'application.

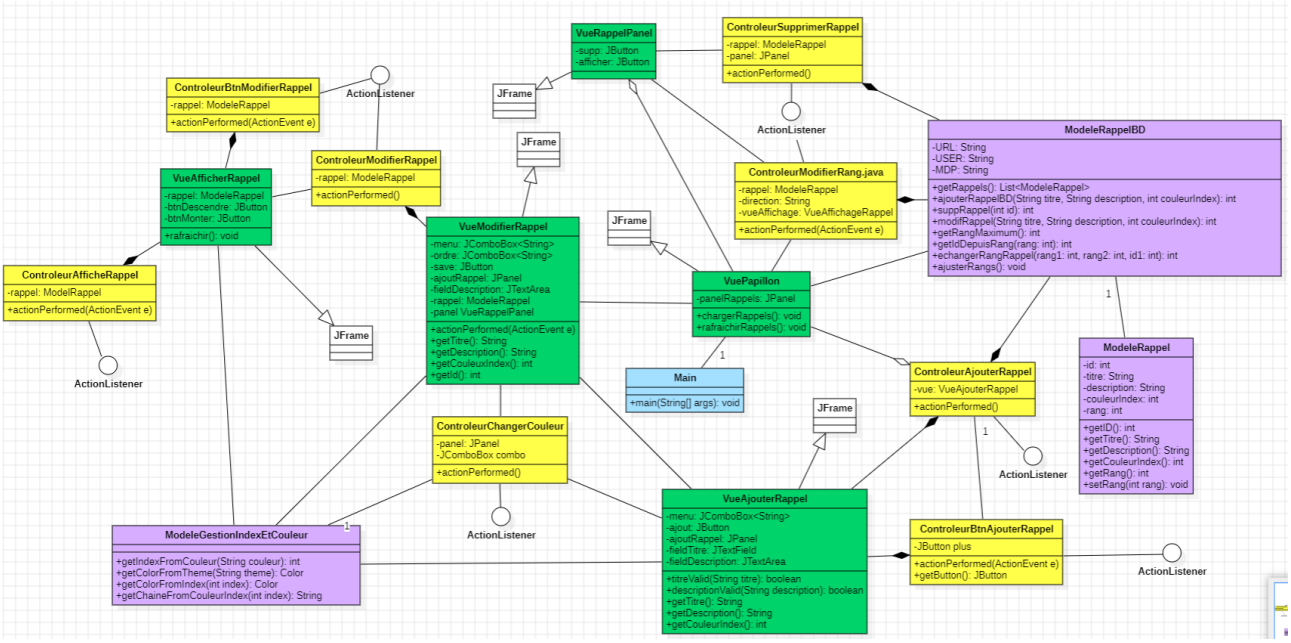
Le main :



4.1 Diagramme de classes simple



4.2 Diagramme de classes complets

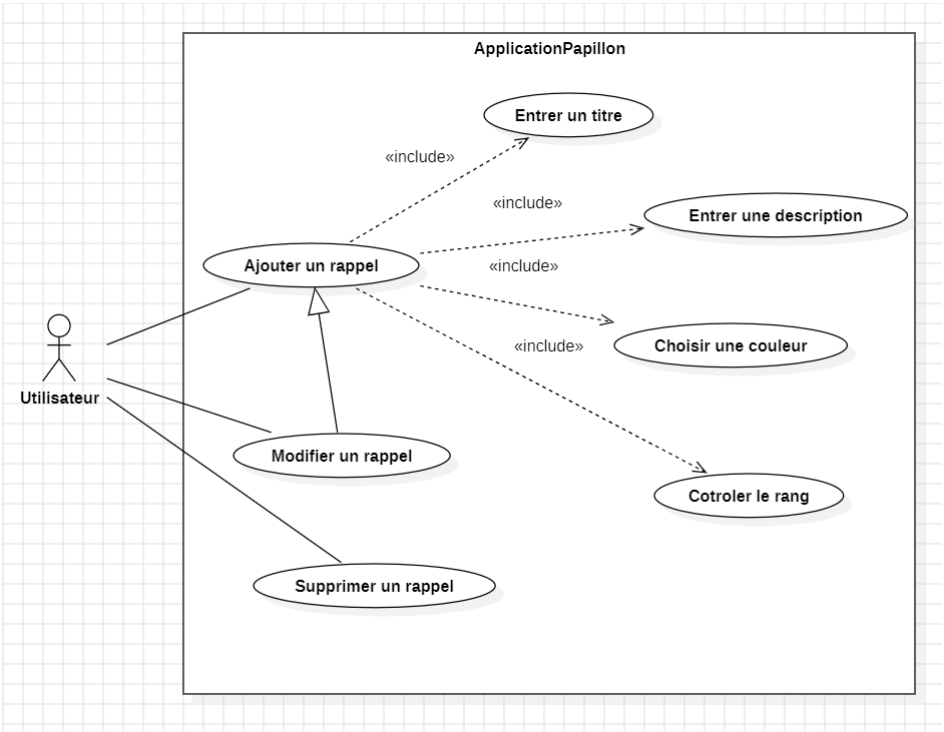


Le diagramme UML présenté modélise l'architecture d'un système de gestion de rappels en Java, structuré selon le paradigme MVC (Modèle-Vue-Contrôleur). Il illustre les relations entre les différentes classes de l'application, leurs responsabilités, ainsi que les interactions entre les composants graphiques, les contrôleurs et les modèles de données.

VuePapillon → VueRappelPanel = Un panneau peut afficher plusieurs rappels
ControleurAjouterRappel → VueAjouterRappel = Un contrôleur agit sur une seule vue
VueAjouterRappel → ModeleRappel = La vue peut créer plusieurs rappels
VueModifierRappel → ModeleRappel = Modifie un rappel donné
ModeleRappel → VueAfficherRappel = Un rappel peut être affiché dans plusieurs vues
VuePapillon → ModeleRappel = La vue principale peut afficher plusieurs rappels
VueRappelPanel → ControleurSupprimerRappel = Un bouton de suppr. est lié à un contrôleur
VueRappelPanel → ControleurAfficheRappel = Chaque bouton est lié à un contrôleur
ControleurBtnAjouterRappel → ControleurAjouterRappel = Le bouton déclenche un contrôleur
ControleurAjouterRappel → ModeleRappel = Ajoute un rappel à un seul modèle à la fois
VueAjouterRappel → ModeleGestionIndexEtCouleur = Utilise une seule vue pour les couleurs
VueModifierRappel → ControleurModifierRappel = La vue de modif. est liée à un contrôleur
VueAfficherRappel → ModeleRappel = Affiche un seul rappel à la fois

4.3 Diagramme de cas d'usage

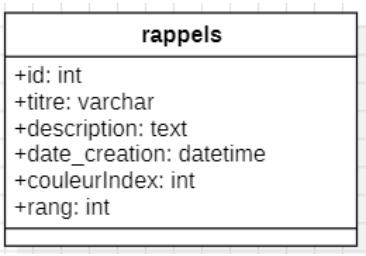
Nous pensons qu'il est pertinent d'imager notre projet avec un diagramme de cas d'usage :



Nous constatons que l'utilisateur peut :
— Ajouter un rappel mais cette nécessite qu'il choisisse d'abord un titre, une description et un thème.
— Supprimer un rappel
— Modifier un rappel qui applique les mêmes propriétés que l'ajout de rappel

C'est un diagramme simple qui regroupe principalement les actions que l'utilisateur peut effectuer.

4.4 Diagramme de la Base de données



La BD n'est composé que d'une table : Rappel.
— id : La clé primaire
— Titre
— Description
— DateCreation : prend automatique la date de création
— CouleurIndex : fait ensuite réference à la couleur
— rang : incrémenté ou décrémenté selon le choix de l'utilisateur

5 Point de vue de chacun

Pour la réalisation de ce projet, nous avons pris la décision de ne pas nous attribuer des parties distinctes du sujet : l'un fait le contrôleur, l'autre fait la vue etc. Nous avons choisi d'avancer ensemble dans le projet. Chacun d'entre nous a travaillé sur toutes les parties. Cela nous a permis de travailler sur tous les points. De plus, cette polyvalence a permis d'avoir une meilleure communication sur ce que nous voulions faire. En effet, en travaillant ensemble nous avons pu communiquer régulièrement et avancer ensemble.

5.1 Antoine DESESQUELLES

Ce projet a été très intéressant. J'ai pu tester ma polyvalence et mettre en pratique les différentes compétences acquises au cours de l'année. Travailler sur un projet concret m'a permis de consolider mes connaissances du modèle MVC, en passant de la théorie à la pratique.

De plus, cela m'a permis de m'entraîner à développer mon esprit critique : repérer les erreurs, les points à améliorer et réfléchir à des solutions adaptées. Ce processus m'a appris à analyser mon travail de manière plus objective et à anticiper les problèmes.

Enfin ce projet m'a donné l'opportunité de me familiariser avec de nouvelles notions que je n'avais pas eu le temps d'approfondir en cours : comme la gestion de fichiers .jar et d'autres outils techniques. Même si certaines étapes ont été un vrai défi, cela a rendu l'expérience d'autant plus formatrice et interessante.

En résumé, ce projet a été un excellent moyen de mettre en pratique mes compétences, d'en acquérir de nouvelles et développer une approche critique de mon travail.

5.2 Chloé WIENCEK

J'ai beaucoup apprécié travailler sur ce projet. D'apparence, il paraissait plutôt simple mais en réalité des challenges se sont révélés au fur et à mesure. La partie la plus intuitive a été la mise en place de la base de données. En effet cela ressemblait à ce que nous avions vu en cours. C'est pareil pour l'interface graphique. Globalement, il n'y a pas eu de grandes difficultés sur ce point-là.

Je n'avais pas beaucoup d'expérience pour faire un Makefile donc ce fut un obstacle de taille. Après beaucoup d'échanges avec d'autres élèves et de lecture de sites internet à ce sujet j'ai enfin pu faire fonctionner le makefile en gérant les dépendances. Désormais, sa réalisation me semble plus intuitive et facile à l'avenir. J'ai également apprécié faire une application utile que l'on peut utiliser tous les jours. Cela donne plus d'idées pour sa réalisation et donne une motivation supplémentaire.

Enfin, en plus de la création directe du projet, j'ai décidé de faire ce rapport en LaTeX que je n'avais jamais utilisé avant.

Ainsi, ce projet m'a permis de me rendre polyvalente : gérer une BD, créer une interface graphique, faire un MakeFile, le rapport en LaTeX