RAPPORT SAE 2.01 JAVA

Introduction :

Ce rapport porte sur la SAE 2.01, dont l'objectif est de concevoir une application Java dans le cadre d'une mise en situation proposée par les enseignants. Le scénario consiste à développer un système pour un réseau fictif de librairies nommé Livre Express. Cette application doit répondre à un cahier des charges précis, avec plusieurs attentes fonctionnelles et techniques.

L'un des objectifs principaux est de proposer aux clients des recommandations de livres basées sur leurs achats précédents. Il est également essentiel que l'interface de l'application soit structurée de manière hiérarchique, avec un menu principal et des sous-menus bien organisés, afin d'offrir une navigation fluide et intuitive.

L'application doit gérer trois types d'utilisateurs :

- Le client, qui peut consulter les livres disponibles, en commander un ou plusieurs, et choisir un mode de livraison (à domicile ou en magasin).

- Le vendeur, qui peut gérer le stock en ajoutant de nouveaux livres ou en modifiant leur quantité, transférer des livres d'une librairie à une autre, et visualiser les ventes.

- L'administrateur, qui peut créer de nouveaux comptes vendeurs, intégrer de nouvelles librairies dans le réseau, consulter les statistiques globales et gérer l'ensemble des librairies.

Cette application vise à simuler un environnement professionnel tout en mettant en pratique les compétences en programmation orientée objet, en gestion de projet, ainsi qu'en modélisation UML.

Dans ce rapport, nous commençons par présenter l'analyse de la base de données, suivie de celle des diagrammes UML. Ces deux étapes ont été réalisées en premier lieu, car elles sont essentielles pour bien structurer le projet et préparer efficacement la suite du développement. Ensuite, nous expliquons notre démarche pour l'implémentation de toutes les fonctionnalités. Dans un second temps, nous abordons la partie concernant l'interface Homme Machine dédiée à cette application. Par la suite, nous expliquons comment l'organisation du travail s'est déroulée dans notre groupe. Finalement, nous faisons le bilan de ce que nous avons réalisé dans ce projet informatique.

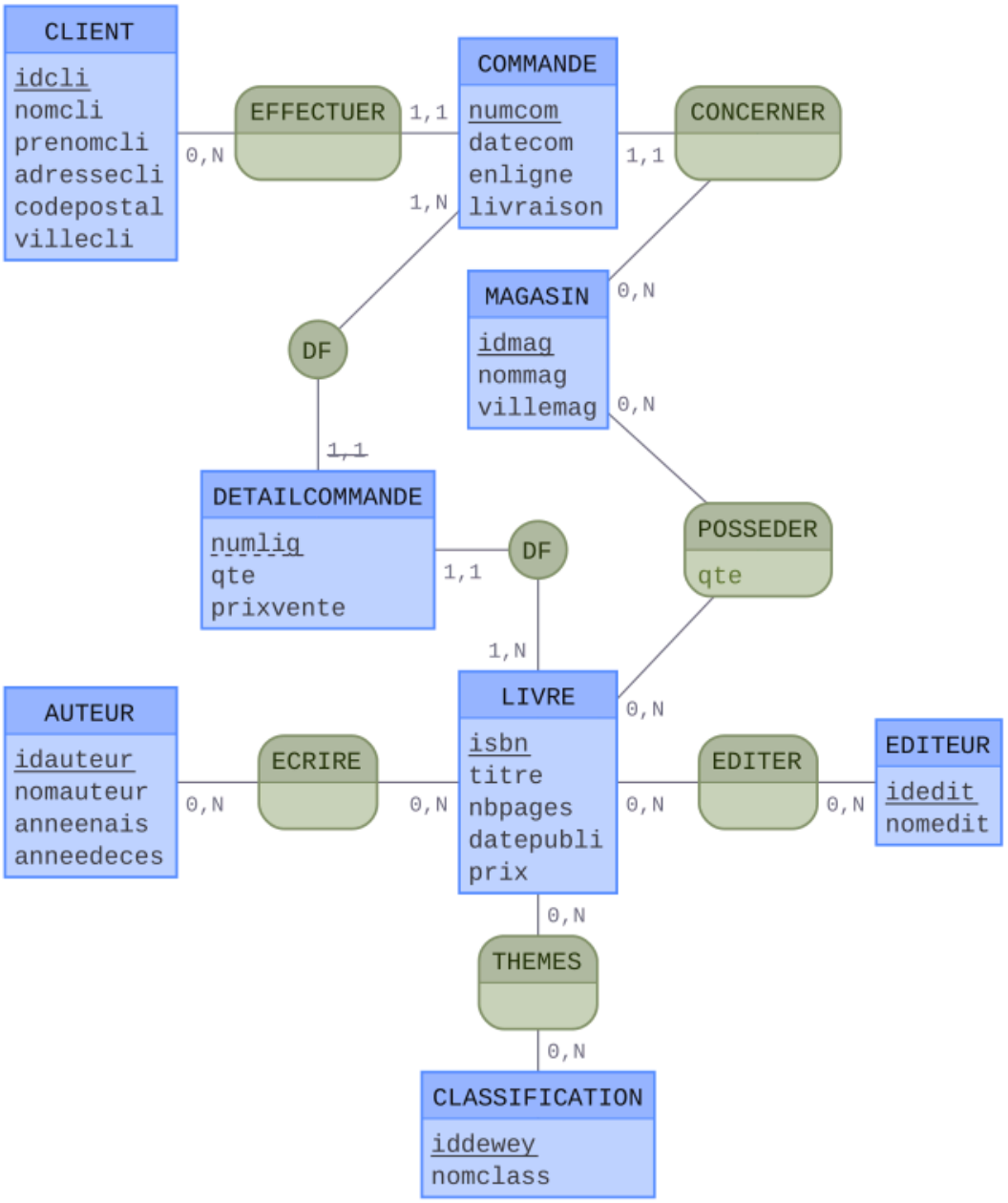
Analyse base de données :

Dans cette partie, c'est la vision orientée sur la base de données qui est mise en valeur. Une base de données est un regroupement organisé d'informations structuré.

1 Analyse du MCD et MLD

Le MCD (Modèle Conceptuel de Données) et le MLD (Modèle Logique de Données) sont des outils qui permettent de représenter une base de données. Le MCD est la première étape dans la conception d'une base de données, c'est un schéma constitué des différentes structures de données de manière simplifiée. Le MLD va se construire à partir du MCD est va être sous la forme de texte. Dans ce modèle on précise les liens entre les différentes tables du modèle en inscrivant les clés primaires et étrangères. La clé primaire d'une table est l'information qui représente la table, il s'agit souvent d'un ID ou d'un numéro. Une clef étrangère est une clé primaire qui se trouve être dans une autre table.

La Base de données du projet est composée de 14 tables, certaines contiennent des informations d'autres servent à relier une ou plusieurs tables entre elles. Par exemple, la table "MAGASIN" permet de stocker toutes les informations sur le Magasin comme son nom ou bien la ville où il se situe. On a aussi des tables comme la table "POSSEDER"qui fait le lien entre la table "MAGASIN" et "LIVRE". Cette table n'apporte pas d'information supplémentaire, elle ne contient que les clés primaires des deux tables qu'elle lie.



(figure 1: MCD fourni dans le sujet de la SAE2.04)

Ci-dessus nous pouvons retrouver le MCD correspondant à la SAE 2.04 et c'est à partir de cette base que la SAE 2.01 s'organise dans la création de son diagramme de classe. Le MLD correspondant à ce modèle et le suivant:

CLIENT [idcli, nomcli, prenomcli, adressecli, codepostal, villecli]
COMMANDE [numcom, datecom, enligne, livraison, idmag, idcli]
MAGASIN [idmag, nommag, villemag]
DETAILCOMMANDE [isbn, numcom, numlig, qte, prixvente]
POSSEDER [idmag, isbn, qte]
LIVRE [isbn, titre, nbpages, datepubli, prix]

ECRIRE[isbn, idauteur]
AUTEUR [idauteur, nomauteur, annenais, anneedeces]
EDITER [isbn, idedit]
EDITEUR [idedit, nomedit]
THEMES [isbn, iddewey]
CLASSIFICATION [iddewey, nomclass]

Avec ces deux schémas on peut voir que la base de donnée est organisée de manière à ce qu'un Client puisse commander plusieurs livres, les informations inscrites dans la commande sont segmentées en plusieurs lignes avec une ligne par livre. Chaque livre à un auteur, une classification et un éditeur. et un livre peut être dans un magasin en une quantité allant de 1 à n.

Pour répondre aux demandes de la SAE JAVA il nous aussi fallu recréer une base de données, nous somme parti d'une copie de la la base de données Librairie et nous avons ajouté plusieurs Tables comme les tables Vendeur et Administrateur.

```
CREATE TABLE VENDEUR(
  PRIMARY KEY (idven),
  idven     int NOT NULL,
  nomven    varchar(50),
  prenomven varchar(30),
  idmag     VARCHAR(42) NOT NULL
);

CREATE TABLE ADMINISTRATEUR(
  PRIMARY KEY (idadm),
  idadm     int NOT NULL,
  nomadm    varchar(50),
  prenomadm varchar(30)
)
```

2.Choix des insertions

Cette partie porte sur l'analyse de la conception de cette base de données et plus précisément sur la composition des tables. Pour toutes les tables, la base de données ne stock que des informations utiles. De plus, il ne faut pas que les informations soient redondantes entre les tables. L'organisation de la base se fait lors des insertions après avoir créé les tables. Plus la base de données est conséquente plus le choix des insertions est important.

```sql
-- Les livres
insert into LIVRE(isbn, titre,nbpages,datepubli,prix) values
    ('9782205054750', 'Les cavernes', 48, 2003, 8.81),
    ('9782012101425', 'Astérix légionnaire', 48, 2008, 7.85),
    ('9782205054996', 'Aberrations', 48, 2005, 29.2),
    ('9782205062403', 'Le visage de la vengeance', 48, 2009, 5.0),
    ('9782070577101', 'Le tour du monde en quatre-vingts jours', 348, 2007, 21.7),
    ('9782205084672', 'L''homme qui tua Chris Kyle', 162, 2020, 15.19),
    ('9782226175113', 'Vous n''aurez pas le dernier mot!', 232, 2006, 11.9),
    ('9782205055252', 'Escale dans le passe', 47, 2005, 11.76),
    ('9782205064223', 'Au poil', 48, 2010, 12.2),
    ('9782205062335', 'Les dragons de Pékin', 48, 2009, 6.14),
```

(figure 2 : capture d'écran des insertions dans la table Livre)

Par exemple, dans la figure 2 ci-dessus, on peut voir que les isbn des livres sont très similaires entre eux. La moindre erreur rendrait la base de données inutilisable car deux livres différents ne peuvent pas avoir le même identifiant; Il est donc important de bien choisir les informations à insérer dans la base de données. Ensuite il faut notifier que chaque information doit avoir le bon type et au bon endroit. Dans sa création une table indique de quelles informations elle a besoin, dans quel ordre et de quel type.

Comme dit dans la partie précédente nous avions créer de nouvelles tables il nous a fallu insérer de nouvelle valeurs pour faire nos tests. En de ces insertions d'autres ont été faites avec l'aide méthode que nous avons implémenté comme ajouterVendeur de Magasin.

```sql
INSERT INTO VENDEUR (idven, nomven, prenomven, idmag) VALUES
  (1, 'Dupont', 'Marie', 1),
  (2, 'Martin', 'Jean', 2),
  (3, 'Durand', 'Sophie', 3);

INSERT INTO ADMINISTRATEUR (idadm, nomadm, prenomadm) VALUES
  (1, 'Lefevre', 'Paul'),
  (2, 'Bernard', 'Lucie');
```

3.Explication des principales requêtes

Pour accéder à certains éléments de la table, il est nécessaire de faire des requêtes bien spécifiques.

```
WITH FACTURE as (select nommag,nomcli,prenomcli,adressecli,codepostal,villecli,numcom, datecom, isbn, titre, qte, prix, SUM(qte*prix) as total
FROM CLIENT natural join COMMANDE natural  join DETAILCOMMANDE natural join LIVRE natural join MAGASIN
where MONTH(datecom) = '02' and YEAR(datecom) = '2020'
GROUP BY nomcli,prenomcli,adressecli,numcom, datecom, isbn, titre,qte, prix),

total_facture as (select numcom,nomcli,prenomcli,adressecli, datecom,SUM(total) as totalfac
FROM FACTURE
Group by numcom,nomcli,prenomcli,adressecli, datecom)


select F.nommag,F.villecli,F.codepostal,F.numcom, F.nomcli,F.prenomcli, F.adressecli, F.datecom,F.isbn, F.titre, F.qte, F.prix, F.total, T.totalfac
FROM FACTURE F
Join total_facture T on F.numcom = T.numcom
Order by nommag, numcom;
```

(figure 3 : capture d'écran de la requête Facture)

Par exemple, la figure 3 est une requête qui dans la première partie crée une vue, qui récupère toutes les informations nécessaires, pour le mois de février 2020 et fait le total de chaque ligne séparément.Ensuite la requête viens faire le total de la facture dans la deuxième partie avec le totalfac. Enfin les parties sont réunies dans la troisième partie pour faire l'affichage global. Pour cette requête je fais une vue temporaire avec le "WITH" pour pouvoir après parcourir cet ensemble de données bien précis. Cette requête a pour but de créer une facture regroupant toutes les ventes d'un magasin. Cette requête est couplée à un programme python qui gère l'affichage et une partie des calculs de totaux.

Une autre requête qui a de l'importance dans la base de données est la requête Palmarès, cette dernière renvoie les auteurs ayant vendu le plus de livres de 2020 à 2024. La requête renvoie l'année, le nom de l'auteur ainsi que le total de ses ventes.

```
WITH Palmarès as (SELECT idauteur, YEAR(datecom) as annee, sum(qte) as total
FROM AUTEUR natural join ECRIRE natural join LIVRE natural join DETAILCOMMANDE natural join COMMANDE
WHERE YEAR(datecom) != '2025'
GROUP BY idauteur, annee
ORDER BY total desc)
SELECT annee, nomauteur, max(total)
FROM AUTEUR
natural right join Palmarès
group by annee;
```

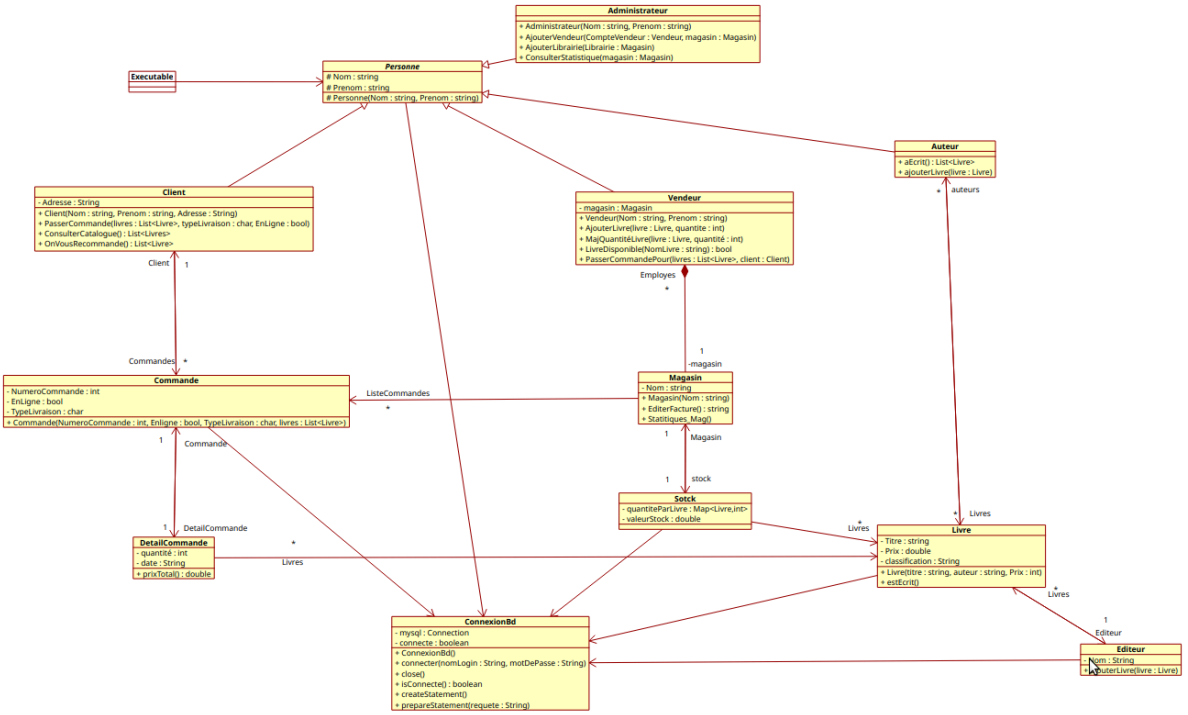
(figure 4 : capture d'écran de la requête Palmarès)

Cette requête est construite en plusieurs parties. La première, est une vue temporaire, elle se charge de regrouper tous les auteurs ayant vendu des livres avant 2025. La vue temporaire s'occupe aussi de calculer le total des ventes de chaque auteur. Ensuite dans la deuxième partie de la requête s'occupe de sélectionner tous les auteurs de la base puis fais une jointure entre cette sélection et la vue temporaire pour ne sélectionner que le meilleur auteur de chaque année.

Ces requêtes permettent d'avoir accès à des informations précises et ordonnées de la manière dont on veut. Ce qui permet d'avoir rapidement accès à des informations sans perdre de temps. Les requêtes sont une partie importante d'une base de données car elles permettent l'utilisation et l'accès aux données de cette dernière.

Analyse UML de l’application :

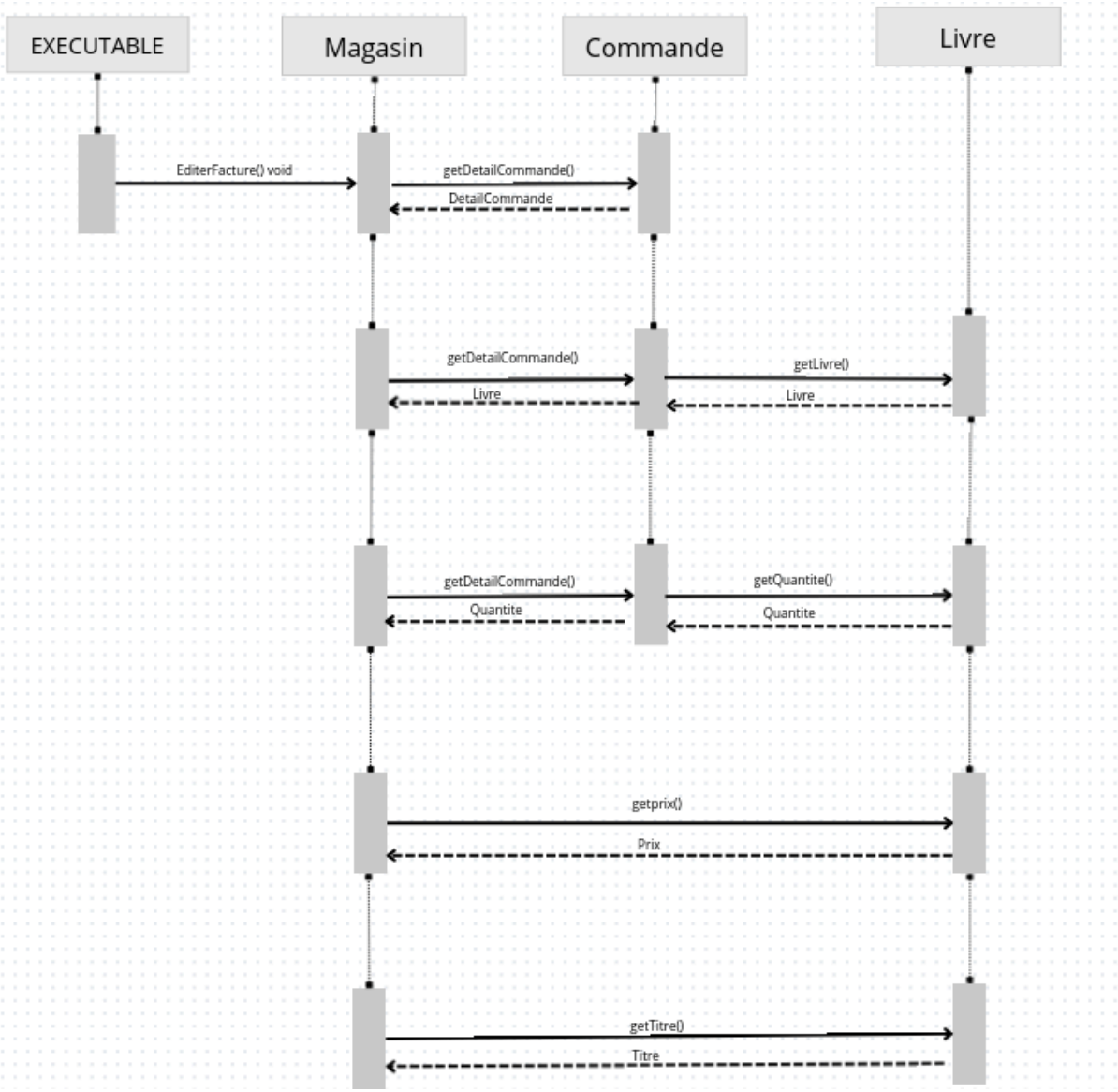
Une analyse UML de l’application permet de faire un point sur le travail à réaliser. Cela passe par deux étapes, la première la réalisation du diagramme de classe qui permet de voir les différentes classes que nous devons implémenter, leurs attributs et méthodes mais aussi les différents liens entre elles. Ensuite, le diagramme de séquence pour les grosses méthodes, permet de réfléchir sur l’acheminement des méthodes et voir quelles classes doivent interagir.



(figure 5 : capture d’écran du diagramme de classes)

Nous avons choisi de ne pas modifier la structure de la base de données initiale. Ce choix s'explique par notre volonté de respecter les contraintes existantes et de gagner du temps en nous appuyant sur un modèle de données déjà fonctionnel. Pour faciliter l’interaction entre l’application Java et la base de données, nous avons représenté les tables sous forme de classes.
Au départ, nous n'avions pas prévu de modéliser chaque table par une classe dans notre diagramme. Cependant, cette décision a été remise en question après les retours des enseignants, qui nous ont souligné l'importance de la correspondance entre les entités de la base et celles du modèle objet pour assurer la cohérence du projet. Nous avons donc adapté notre diagramme en conséquence.
Afin d’éviter les redondances entre les classes Client, Vendeur, Administrateur et Auteur, nous avons introduit une classe abstraite Personne contenant les attributs communs tels que le nom et le prénom. Ce choix nous a permis de centraliser certaines fonctionnalités et de simplifier la maintenance du code.
Enfin, une classe ConnexionMySQL a été développée pour gérer la connexion entre Java et la base de données, en utilisant l’API JDBC. Cela nous a permis d'encapsuler les aspects

techniques de la communication avec la base dans un composant dédié, ce qui renforce la modularité de notre application.



Implémentation

Dans le classe Client :
La méthode consulterCatalogue() permet de récupérer la liste de tous les livres présents dans la base de données. Elle exécute une requête SQL qui sélectionne tous les ISBN de la table LIVRE, puis pour chaque ISBN trouvé, elle crée un objet Livre correspondant et l'ajoute à une liste. Cette liste de livres est ensuite retournée, ce qui permet à un client de parcourir l'ensemble du catalogue disponible.

La méthode onVousRecommande() propose des recommandations personnalisées à un client en se basant sur les achats des autres clients ayant des goûts similaires. Elle commence par récupérer les livres déjà achetés par le client courant. Ensuite, elle identifie les autres clients ayant au moins trois livres en commun avec lui. Parmi les livres achetés par ces clients similaires, elle sélectionne ceux que le client courant n'a pas encore achetés. Enfin, elle récupère les titres de ces livres pour les proposer en recommandation, offrant ainsi une suggestion pertinente basée sur les habitudes d'achat des clients ayant des profils proches.

La méthode ajouterLivre() permet au vendeur d'ajouter un livre dans le stock de son magasin. Elle vérifie d'abord si le livre (identifié par son ISBN) existe déjà dans le stock (POSSEDER). Si oui, elle met simplement à jour la quantité en ajoutant la quantité fournie à la quantité existante. Si le livre n'est pas encore présent dans le stock du magasin, elle crée une nouvelle entrée dans la table POSSEDER avec la quantité indiquée.

La méthode majQuantiteLivre() sert à modifier la quantité d'un livre déjà présent dans le stock du magasin du vendeur. Elle effectue une mise à jour de la quantité (qte) dans la table POSSEDER pour le livre et le magasin concernés, en ajoutant la quantité passée en paramètre (qui peut être positive ou négative). Si le livre n'est pas trouvé dans le stock, un message d'erreur est affiché.

La méthode livreDisponible() permet de vérifier si un livre, identifié par son titre, est disponible dans le stock du magasin du vendeur. Elle effectue une requête qui joint les tables POSSEDER et LIVRE pour retrouver la quantité disponible du livre dans le magasin. Si la quantité est supérieure à zéro, la méthode retourne true, indiquant que le livre est disponible, sinon elle retourne false.

La méthode ajouterLibrairie() permet à l'administrateur d'ajouter une nouvelle librairie (magasin) dans la base de données. Elle commence par générer un nouvel identifiant unique pour la librairie en prenant le maximum existant dans la table MAGASIN et en ajoutant 1. Ensuite, elle insère les informations du nouveau magasin (nom et ville) dans la table MAGASIN. Si l'opération réussit, un message de confirmation est affiché, sinon une erreur est signalée.

La méthode consulterStatisques() affiche des statistiques globales sur l'activité de la librairie. Elle commence par afficher le chiffre d'affaires de chaque magasin ainsi que le chiffre d'affaires total, puis elle affiche le livre le plus vendu (bestseller) toutes boutiques confondues. Cette méthode se contente d'appeler successivement les méthodes afficherChiffreAffaires() et bestseller(), et structure l'affichage avec un titre pour plus de clarté.

La méthode afficherChiffreAffaires() affiche le chiffre d'affaires réalisé par chaque magasin ainsi que le chiffre d'affaires total. Elle effectue une requête SQL qui agrège, pour chaque magasin, le montant total des ventes en multipliant la quantité vendue par le prix de vente pour chaque ligne de commande. Les résultats sont affichés magasin par magasin, puis le total général est calculé et affiché à la fin.

La méthode bestseller() affiche le livre le plus vendu toutes boutiques confondues. Elle exécute une requête SQL qui regroupe les ventes par livre (en utilisant l'ISBN et le titre), additionne les quantités vendues, puis trie les résultats pour ne garder que le livre ayant le plus grand nombre d'exemplaires vendus. Le titre du livre et la quantité vendue sont affichés, ou un message indique qu'aucun livre n'a été vendu si c'est le cas.

Travail de groupe :

Organisation du travail

Dès le début de la SAE, nous avons entamé une réflexion collective autour du diagramme de classes. Une fois ce dernier suffisamment détaillé, nous avons structuré notre travail en plusieurs étapes : la finalisation du diagramme de classes, l'élaboration des diagrammes de séquence, puis l'implémentation en Java des différentes classes et méthodes. Afin d'optimiser notre efficacité, nous avons réparti les classes entre les membres de l'équipe. Chacun a pu développer les méthodes et les tester, puis nous avons mis en commun nos avancements pour les analyser ensemble et ajuster si besoin.

Communication et collaboration

Pour assurer une bonne coordination au sein de l'équipe, nous avons mis en place plusieurs outils de communication. Un groupe Discord nous a permis d'échanger facilement par messages et d'organiser des visioconférences lorsque des points techniques ou des décisions collectives devaient être abordés. Parallèlement, GitHub a été utilisé pour le partage du code et la gestion des versions, facilitant ainsi le suivi des modifications et la collaboration à distance.

Rédaction du rapport

Enfin, la rédaction de ce rapport a été réalisée de manière collaborative grâce à Google Docs. Cet outil nous a permis de travailler simultanément sur le même document, de proposer des modifications en temps réel et de maintenir une cohérence globale dans la présentation du projet.
Diagramme de Gantt

SAE 2.01 JAVA

Livre Express

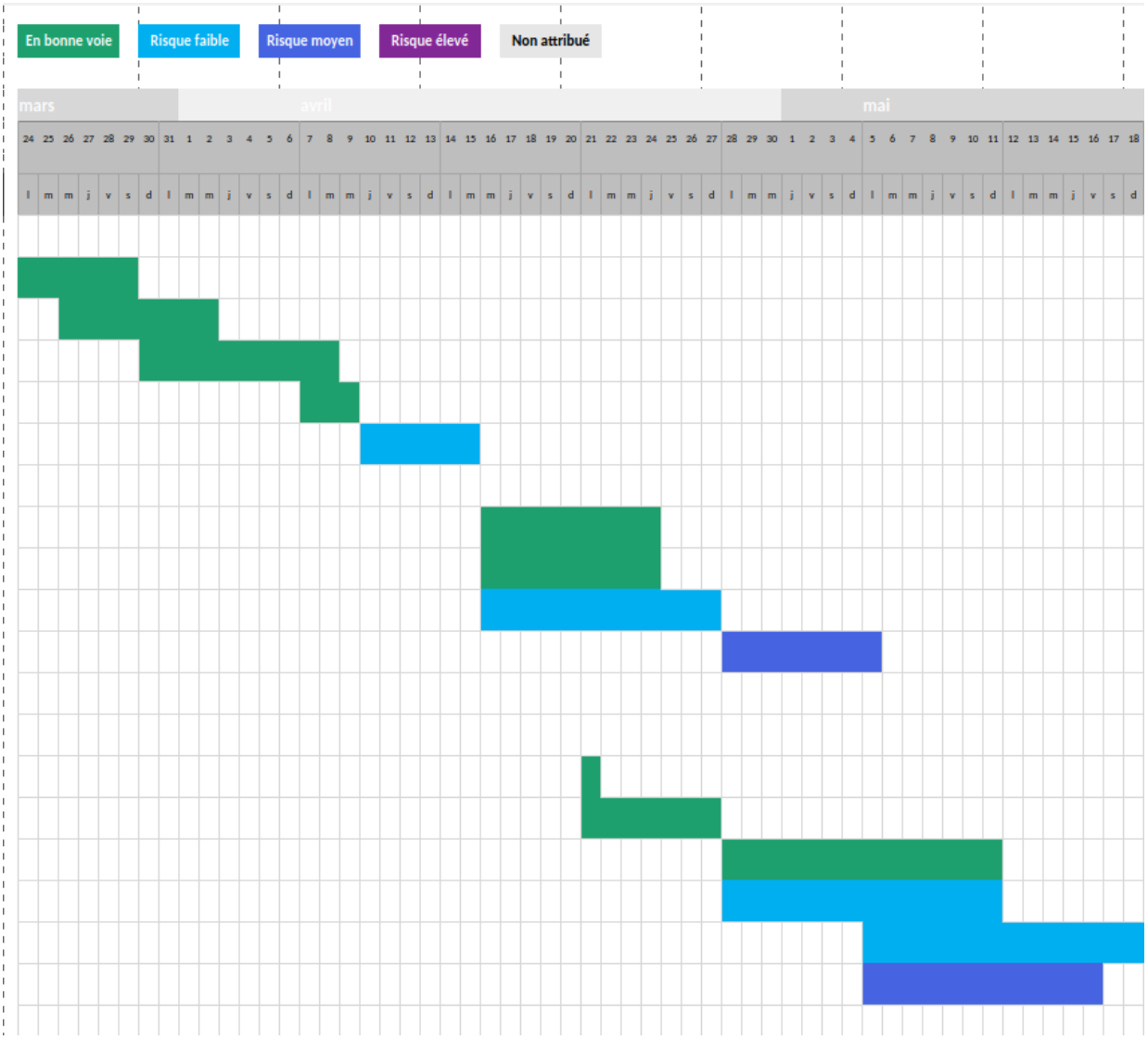
Légende :

Quentin Moreau, Lucas Desgranges, Enzo Charrier, Clément Schmit

Date de début du projet : 23/03/2025

Incrément de défilement : 1

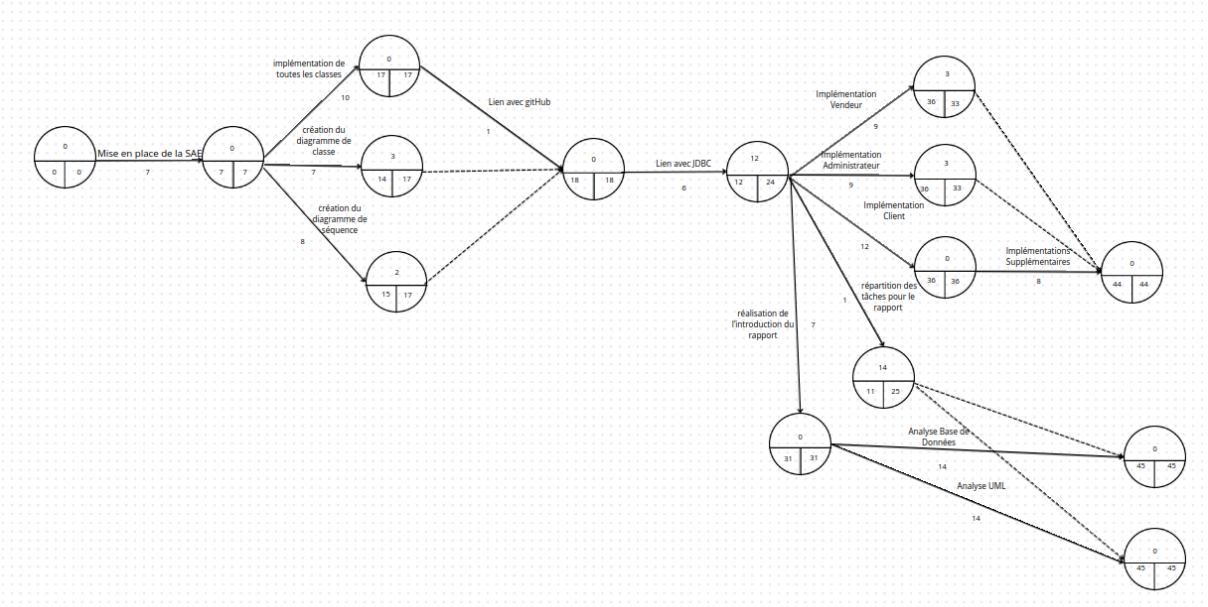
Description du jalon	Catégorie	Attribué à	Progression	Début	Jours
Mise en place de la Sae					
création du diagramme de classe	En bonne voie	Nom	80 %	23/03/2025	7
création du diagramme de séquence	En bonne voie		50 %	26/03/2025	8
implémentation du toutes les classes	En bonne voie		80 %	30/03/2025	10
lien avec GitHub	En bonne voie		100 %	07/04/2025	3
lien JDBC	Risque faible		100 %	10/04/2025	6
Réalisation de la Sae					
Fonctionnalité à implémenter : Administrateur	En bonne voie		20 %	16/04/2025	9
Fonctionnalité à implémenter : Vendeur	En bonne voie		50 %	16/04/2025	9
Fonctionnalité à implémenter : Client	Risque faible		33 %	16/04/2025	12
Contraintes : fonctionnalités supplémentaires	Risque moyen		10 %	28/04/2025	8
Elaboration du rapport					
Répartition des tâches	En bonne voie		100 %	21/04/2025	1
Introduction	En bonne voie		100 %	21/04/2025	7
Analyse Base de données	En bonne voie		100 %	28/04/2025	14
Analyse UML	Risque faible		50 %	28/04/2025	14
Travail de groupe	Risque faible		50 %	05/05/2025	14
Bilan	Risque moyen		20 %	05/05/2025	12


En bonne voie
Risque faible
Risque moyen
Risque élevé
Non attribué
mars
avril
mai
24 25 26 27 28 29 30 31 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
l m m j v s d l m m j v s d l m m j v s d l m m j v s d l m m j v s d l m m j v s d l m m j v s d l m m j v s d l m m j v s d

matrice Raci :

Service/Organisation		BUT informatique			
R : Réalisateur A : Approbateur C : Consultant I : Informé		Chef de Projet	Co-gérent	Manager	Rédacteur
ID	Livrable ou tâche	Lucas	Quentin	Enzo	Clément
A	création du diagramme de classe	C	C	C	R
B	création du diagramme de séquence	R	R	C	C
C	implémentation du toutes les classes	R	R	R	R
D	lien avec GitHub	R	R	I	I
E	lien JDBC	C	R	C	C
F	Fonctionnalité à implémenter : Administrateur	R	R	R	R
G	Fonctionnalité à implémenter : Vendeur	R	R	R	R
H	Fonctionnalité à implémenter : Client	R	C	C	C
I	Contraintes : fonctionnalités supplémentaires	R	R	R	R
J	Répartition des tâches Rapport	R	R	R	R
K	Introduction Rapport	I	R	I	R
L	Analyse Base de données Rapport	I	I	I	R
M	Analyse UML	I	R	I	I
N	Travail de groupe Rapport	R	R	I	I
O	Bilan Rapport	R	I	I	I

Diagramme de Pert



Bilan du groupe

Cette SAE nous a beaucoup apporté, notamment sur le plan professionnel. Elle nous a permis de renforcer nos compétences en développement Java. Elle a également participé au développement de nos compétences en travail d'équipe et en communication, notamment grâce à l'utilisation d'un groupe Discord pour la coordination et d'un dépôt GitHub pour la gestion du projet.