2 mai 2025 En cours

RAPPORT SAE 2.01 - JAVA

Membres du groupe

- Nicolas Camera
- Quentin Chaufour
- Jolan Rolland
- River Huche

Introduction

Dans le cadre de la SAE 2.01, nos enseignants nous ont proposé une mise en situation de développement d'une application permettant de gérer et de vendre des livres pour le compte de l'entreprise fictive: "Livre-Express".
La première partie du projet consiste à analyser le besoin, et à développer l'application en Java, qui va se servir de notre précédente SAE Base de Données dans laquelle nous avons créé et utilisé une base de données pour cette même entreprise.
Dans la continuité, nous allons devoir imaginer puis créer une interface IHM qui va interagir avec notre application et notre base de données, le tout en prenant en compte les besoins des 3 types d'utilisateurs : Administrateur, Vendeur et Client.

Ce travail cible donc la compétence 1 : développer.

Dans ce rapport de SAE, nous allons vous présenter les outils que nous avons utilisés tel que la base de données, les différents schémas à savoir les diagrammes de classe et de séquence. Nous allons ensuite analyser le fonctionnement de l'application, puis la partie Interface homme machine, enfin nous aborderons l'organisation de notre groupe.

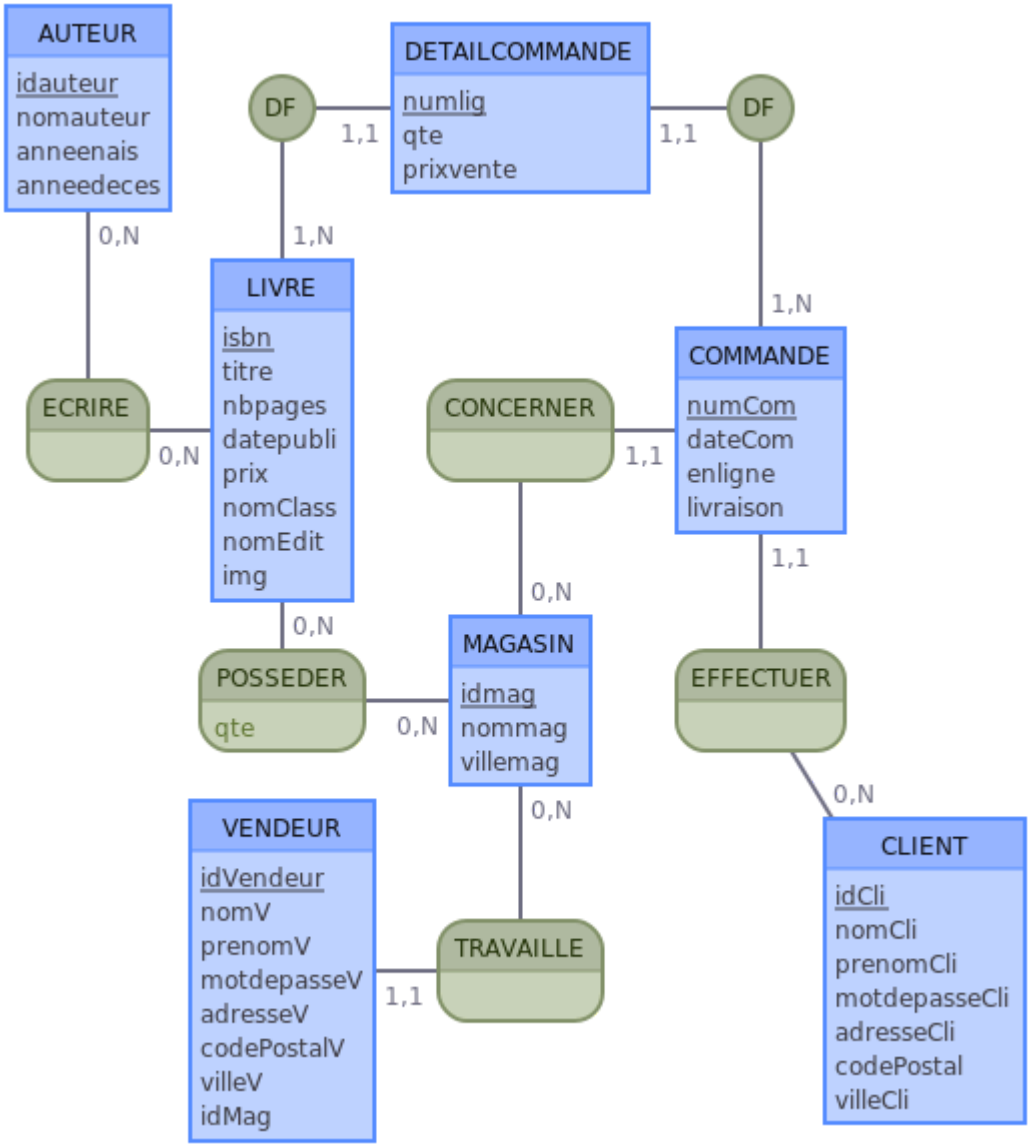
Nous conclurons, sur les fonctionnalités que nous avons réussi à implémenter et sur l'efficacité de l'organisation de notre groupe, ce qu'il nous a apporté sur ce projet et les compétences que nous avons pu développer avec ce travail.
Et enfin, notre retour personnel.

I - Analyse de la base de donnée et des modifications

En amont de ce projet nous avons travaillé sur sa base de données dans le cadre d'une une SAE spécifique. Le travail consistait à créer des requêtes à l'aide d'une base de données qui nous été fournie avec la structure, le script de création et l' ajout de données en SQL.
Un MCD est un graphique qui permet de représenter une base de données, avec ses associations, les tables qui contient les attributs, les clés primaires qui permettent d'identifier les attributs de cette table, le script quant à lui nous permet d'ajouter les valeurs et les données à ces tables afin de voir si nos requêtes fonctionnent correctement et le script de création pour créer les tables du MCD. Pour ce projet il nous a fallu créer des requêtes, faire un insert et des requêtes qui vont se servir de Openoffice calc afin de réaliser des graphiques.
Ainsi, pour cette SAE java nous allons nous servir du MCD donné et du script de création. De plus, cette base de donnée a été modifiée pour ajouter les tables qui nous serviront pour les méthodes que nous souhaitons implémenter

1) Explication du MCD/MLD

Ainsi, d'après le MCD donné dans la SAE BD nous pouvons observer que la table *CLIENT*, peut posséder plusieurs commandes mais que chaque commande est reliée à un client unique, de plus la commande est également liée à un seul magasin et possiblement plusieurs détails de commande qui permettent d'avoir les détails de chaque livre commandé.
Un livre peut être lié à un ou plusieurs auteurs, éditeurs et classifications.
Chaque livre a un stock par rapport à un magasin avec l'association *POSSEDER*.
Il y a aussi des associations qui permettent de faire le lien entre les auteurs,éditeurs et classifications avec les livres par les tables *ÉCRIRE, ÉDITER* et *THEME*.



Nous avons donc modifié le MCD pour l'adapter à cette SAE avec de nouvelles tables.

2) Choix des insertions

Ensuite, pour les insertions de la base de données. La grande majorité de ces dernières nous étaient fournies. Ainsi, nous n'avons pas eu à ajouter de nombreuses insertions étant donné qu'elles nous étaient fournies. Nous avons juste réalisé l'insertion qui nous a été demandée dans le sujet, à savoir l'insertion du livre "La base de donnée pour les nuls". Malgré cela après nous allons détailler par la suite les modifications que nous avons faites notamment sur l'insertion

3) Explication des requêtes principales

Durant la SAE base de données, nous avons dû réaliser plusieurs requêtes tel que le calcul du chiffre d'affaires concernant une librairie et un classement des auteurs les plus populaires.
Nous avons également réalisé des graphiques à partir des résultats des requêtes élaborées précédemment, en utilisant LibreOffice Base et LibreOffice Calc.
Enfin, nous avons dû créer la requête palmarès avec des vues, ainsi que la requête a de facture dans laquelle on récupère toutes les commandes selon un certain mois et une certaine année.

4) Explication modification BD

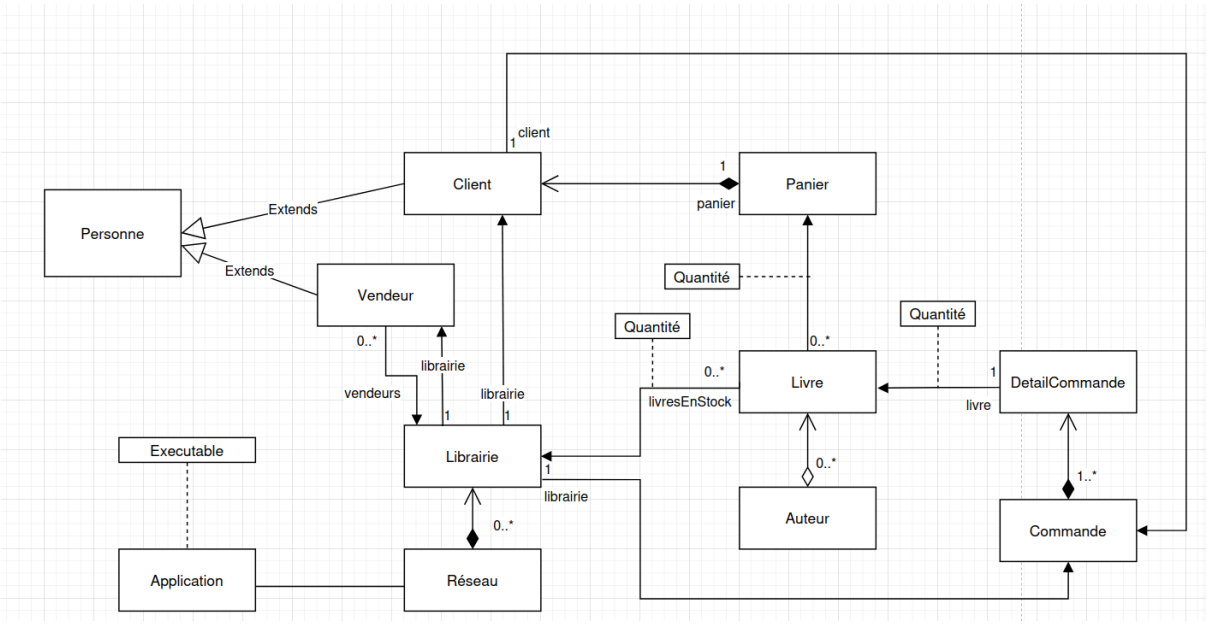
Pour cette SAE, nous avons modifié la base de données afin de l'adapter au contexte de ce projet. En effet nous avons notamment ajouté la table *VENDEUR*, cependant nous avons enlevé la *CLASSIFICATION* et *EDITEUR* que nous avons directement intégré à la table *LIVRE* afin que cela soit plus facile à manipuler, mais aussi car nous considérons qu'un "même" livre ayant différents éditeurs sont des livres à part entière ne possédant qu'une classification principale.
Enfin, nous avons ajouté un mot de passe a la table *CLIENT* et *VENDEUR* permettant une identification simple d'un utilisateur et d'un vendeur. Ainsi puisque ces modifications ont été faites nous avons dû modifier les insertions qui nous avait été données, ainsi il a fallu intégrer la classification et les éditeurs aux livres et insérer de nouveaux vendeurs.

II - UML

1 - Structure de l'application

Durant la phase de planification du projet, nous avons établi les différents cas d'utilisation de notre application, ainsi que la structure de l'application sous forme de diagramme UML (Unified Modeling Language).

Pour cela, nous avons créé un diagramme de classe représentant la structure de l'application, dont voici une version simplifié décrivant les relations entres les différentes classes Java :



Ce diagramme se compose de multiples éléments représentant les classes Java utiles à la création de notre Application.

Nous avons décidé de généraliser les Clients et les vendeurs utilisant une classe parent : *Personne*.

Un *Client* est en possession d'un unique *Panier* pouvant contenir des *Livres* en la quantité voulu provenant d'une *Librairie*.

De plus, nous considérons qu'un *Client* est en relation avec une seule et unique *Librairie* a la fois, de même pour un *Vendeur* qui ne peut travailler dans plusieurs *Librairies*.

La *Librairie* est en connaissance de l'ensemble des *Vendeurs* travaillant pour celle-ci.

Notre *Réseau* est composé de toutes les *Librairies* de l'entreprise.

Nous avons ajouté un panier qui appartient au client et qui contient des livres en une certaine quantité.

Ces *Livres*, sont présents en une quantité donnée dans les stocks d'une librairie, et possèdent 1, plusieurs ou aucun *Auteur*.

Enfin, une *Commande* est affiliée à une unique *Librairie* et un unique client .
Elle est constituée d'au minimum un *DetailCommande* représentant un livre de la commande et sa quantité commandée.

La classe *Application* permet la gestion des interactions entre l'utilisateur et le reste de l'application.

Une version plus complète et détaillée de ce diagramme est disponible en annexe.

2 - La commande d'un client

Lorsqu'un client passe commande de son panier, l'application doit pouvoir prendre en compte la validité d'une commande et les modifications induites dans les stocks.

Pour permettre une implémentation simplifiée, nous avons élaboré un diagramme de séquence décrivant le fonctionnement de la méthode commander() symbolisant l'action de l'utilisateur.

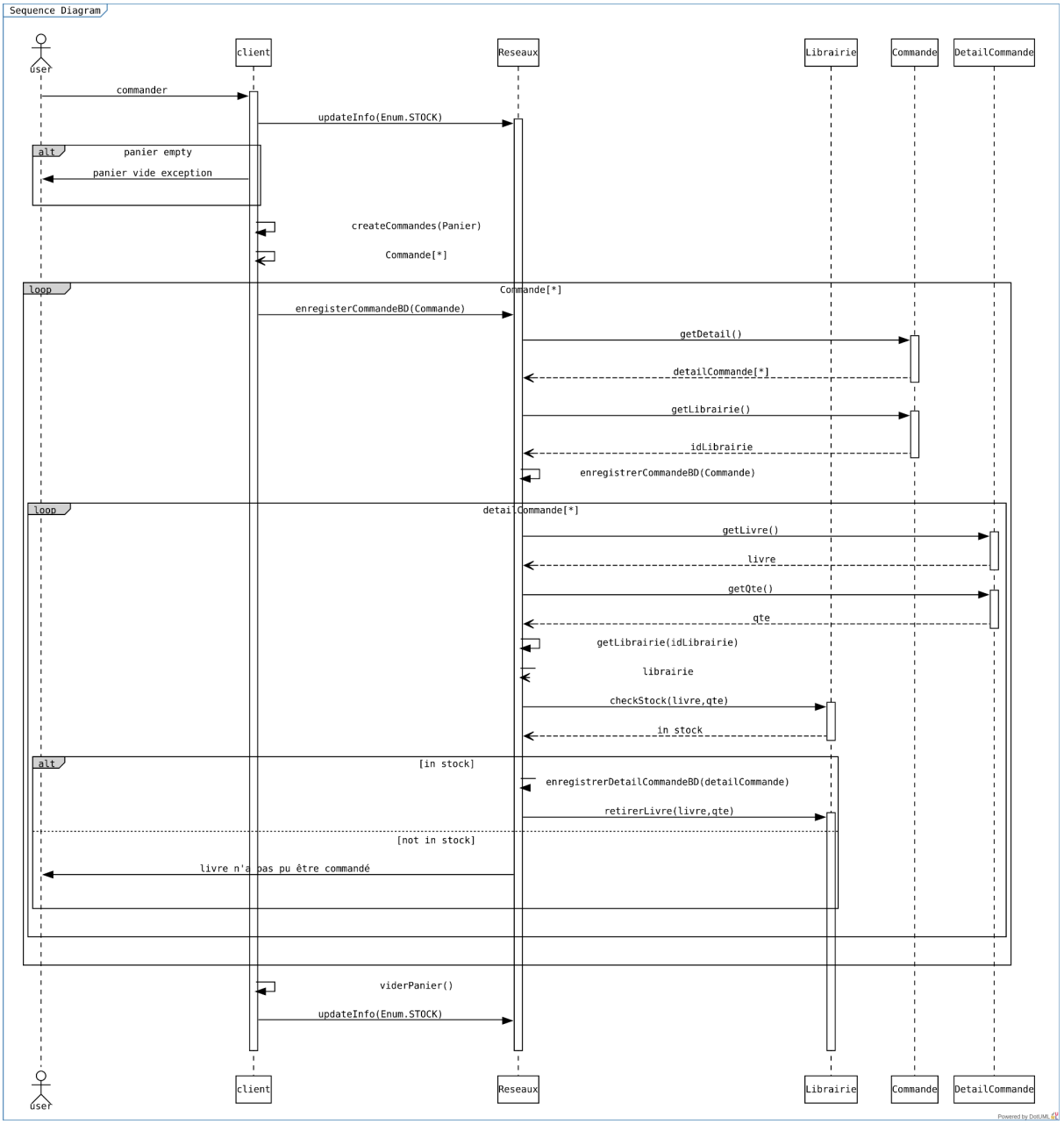


Diagramme de séquence commander()

Pour débuter, une mise à jour des stocks de chaque librairie du réseau est effectuée pour assurer la validité des commandes du client.

Dans le cas où le panier du client est vide, aucune commande ne peut être passée donc on informe le client que son panier est vide, la méthode s’arrête.

Autrement, les commandes sont créées conformément aux contenus du panier du client.

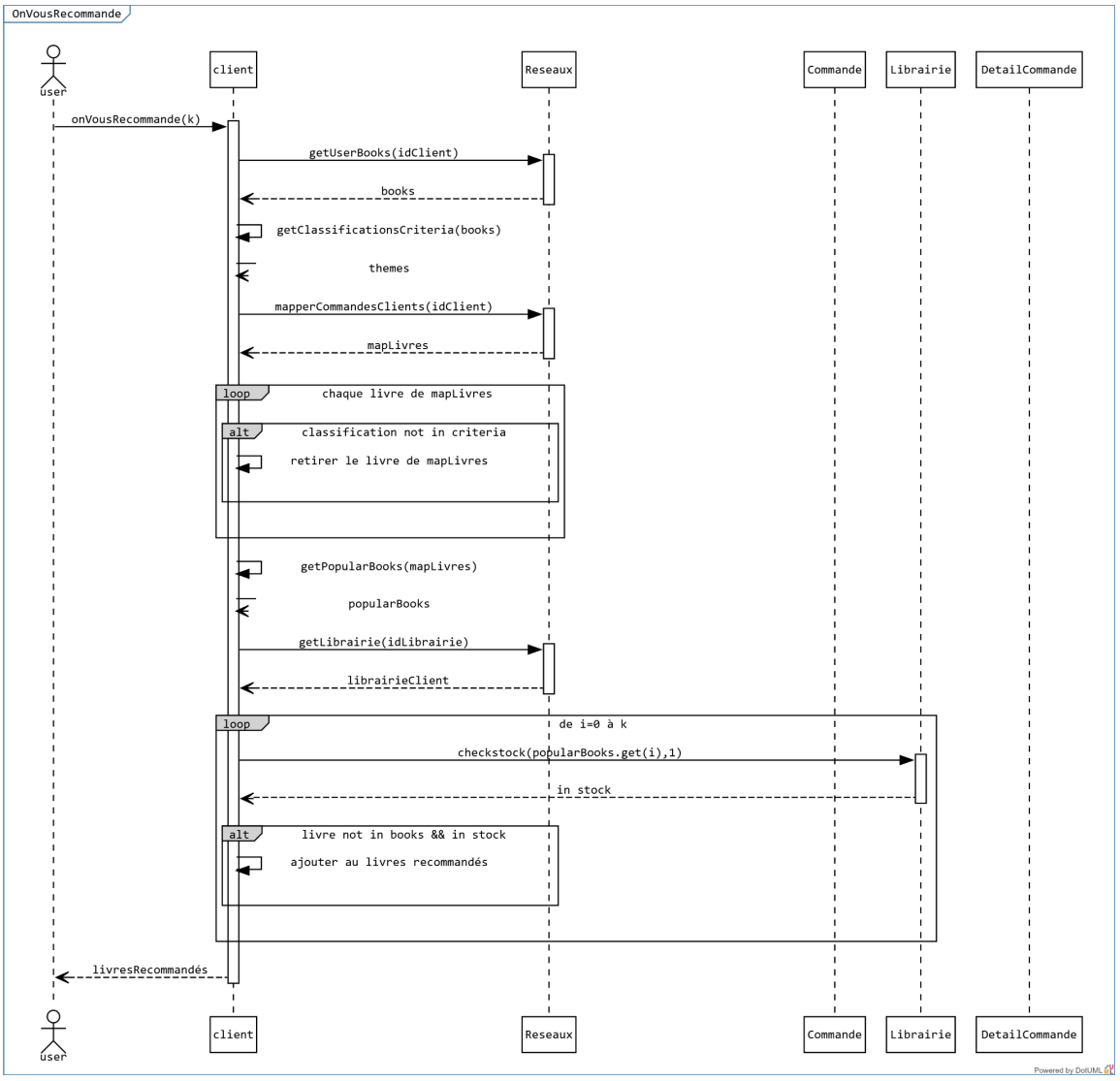
Ensuite, pour chaque commande, on l'enregistre en base de données par l'intermédiaire du *reseaux*, vérifiera la validité de chaque détail de la commande en comparant la quantité commandée avec la quantité actuellement en stock.

Une fois contrôlé, chaque détail est enregistré sur la base de données, et le stock du livre commandé dans la librairie est réduit ou supprimé si le stock tombe a 0.

Enfin, l'application rend compte à l'utilisateur des livres qui n'ont pas pû être commandés et vide le panier du client.

3 - La recommandation de livre à un client

L'application doit aussi permettre à un utilisateur de se faire recommander des livres en fonction des achats précédents du client concerné.



En voici le fonctionnement :

Nous considérons la classification (thème) du livre comme critère pour recommander des livres.

Pour cela, nous commençons par déterminer l'ensemble des genres de livres commandés par le client.

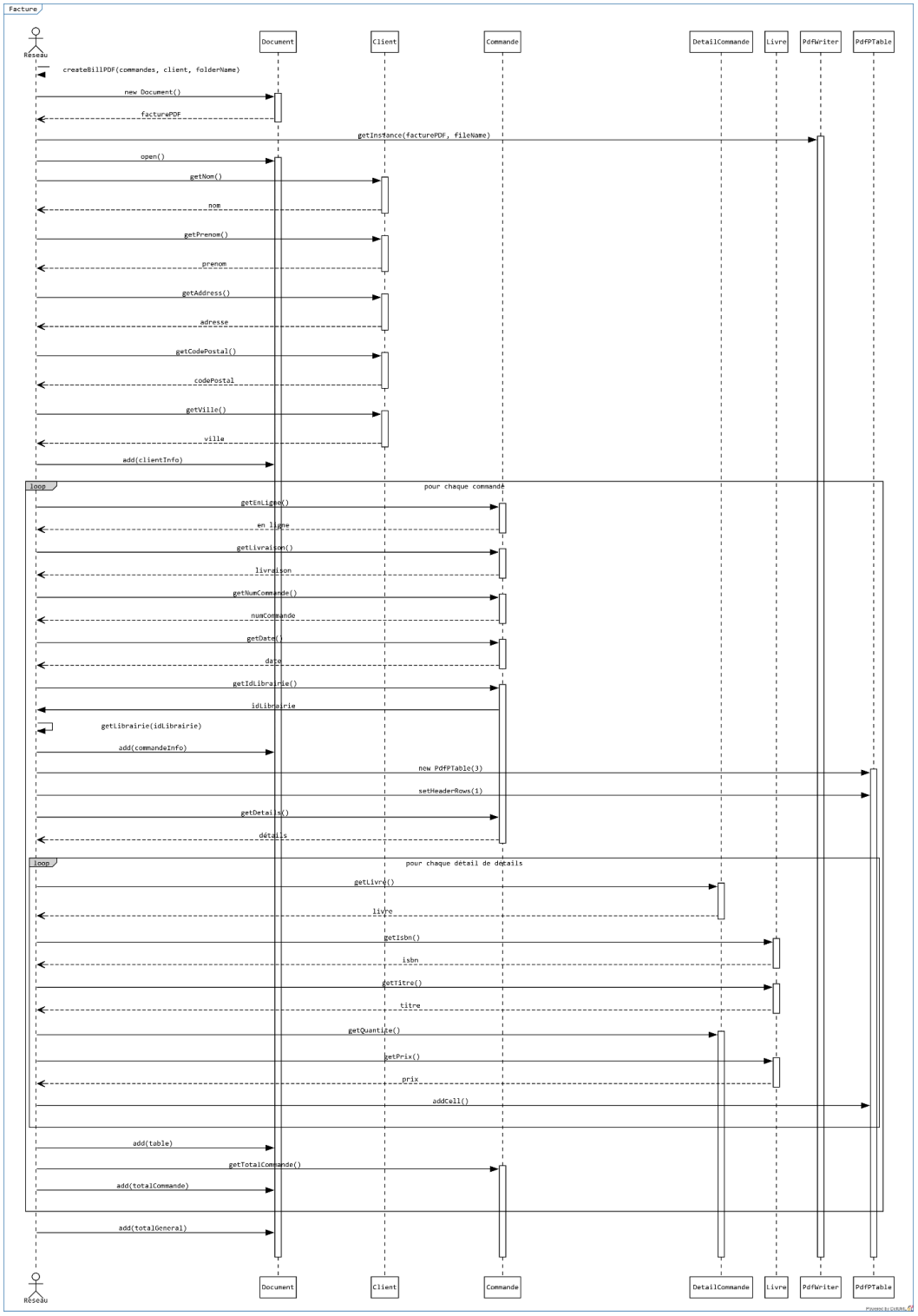
Puis, nous faisons une cartographie de l'ensemble des clients ayant commandé un même livre (lui-même exclu), de laquelle nous supprimons les livres de classification qui ne sont pas dans les genres de livres commandés par le client déterminé auparavant et les livres déjà commandés par le client.

Nous organisons ensuite cette cartographie selon le nombre de clients ayant acheté un livre, nous obtenons alors une carte des livres ordonnée par popularité.

Ce tri nous permet de sélectionner le nombre de livres demandés par le client

4 - La facture

Lorsqu'un client passe une commande de son panier, l'application doit offrir la possibilité de produire une facture des commandes faites.
La facture est produite de la manière suivante :



La facture sera décomposée en multiples parties, la première partie étant les informations du client ayant passé commande en en-tête.

Chaque commande effectuée forme un bloc comprenant les données utiles lié à la commande (date de commande, la librairie associée, type de livraison etc …), un tableau décrivant le contenu de la commande soit le livre, la quantité commandée et le prix total pour ce livre.
Ce bloc se clos sur le montant total de la commande.
Le montant total de la facture sera indiqué après la description de l'ensemble des commandes passées.

III - Implementation

Lors de la phase d'implémentation, nous avons combiné les analyses précédentes de base de données et UML.

Nous avons calqué les tables présentes en base de données tel que le client, les livres et auteurs etc …
A cela, nous avons décidé de centraliser la majorité des méthodes impliquant une interaction avec la base de données dans une bibliothèque nommée *Réseau.*

Nous avons commencé par les implémentations simples et basiques de chaque classe présentes en BD puis nous avons implémenté les méthodes du réseau en utilisant les librairies mariadb.jdbc pour les échanges avec la base de données et itextpdf permettant de créer des pdf.

Pour permettre les tests des méthodes de la classe réseau, nous avons utilisé un serveur mariaDB local en utilisant Docker

Enfin, nous avons développé une interface terminal de notre application avec des menus pour chaque utilisateur et les actions possibles avec traitement des entrées des utilisateurs.

Pour tester nos méthodes nous avons utilisé JUnit, et avons créé une base de données spécialement pour les tests copiant à l'identique la base de données principale nous permettant d'insérer les données nécessaire et assurant le déterminisme des résultats obtenus .

Nous avons utilisé de multiples systèmes de données simples tels que les *ArrayList,* et d'autres plus complexes tels que les *HashSet* et *HashMap.*
Nous avons pu aussi découvrir une nouvelle implémentation de Map avec LinkedHashMap nous permettant dans le cas de la réalisation d'un palmarès, de

garder l'ordre d'entrée des éléments, parfait pour un classement, tout en permettant de garder des informations importantes en tant que valeur et en simplifiant l'implémentation.

L'exemple de : *onVousRecommande()*

La méthode *onVousRecommande()* a été implémentée conformément au diagramme de séquence.

En effet, on récupère l'ensemble des livres déjà commandés par le client pour, à la fois déterminer un ensemble de critère (thème du livre ici) et permettre d'exclure ces livres des possibilités de recommandation.

Puis on récupère un dictionnaire (Map) ayant pour clé les livres commandés et en valeur l'ensemble des ID client ayant commandé ce livre, tout en excluant la client a recommander.

Ce dictionnaire est ensuite traité pour créer une liste des livres triés selon le nombre de personnes ayant commandé le livre.

Enfin, selon le nombre de livres à recommander demandé, on crée une liste à partir de la liste des livres populaires tout en vérifiant que le dit livre n'a jamais été commandé par le client, et qu'il est disponible dans les stocks de la librairie courante du client.

Nous avons également utilisé de nombreuses exceptions afin de rendre notre application plus fiable, notamment lorsque l'on crée une commande si la quantité de livre n'est pas valide alors la méthode renverra une exception avec un message d'erreur associé.

Organisation du travail

Comment vous êtes-vous organisés tout au long de cette SAE ?

Pour ce projet notre groupe est composé de 4 personnes. Nous avons organisé ce travail à l'aide de l'outil de gestion et d'organisation Trello. En effet, nous avons choisi ce dernier puisque la majorité d'entre nous avons déjà utilisé cet outil pour les précédentes SAE.

En parallèle, nous avons réalisé un diagramme de Gantt afin d'organiser le partage des tâches (diagramme disponible en annexe)

En ce qui concerne le rapport nous nous sommes servis de Google Docs afin que ces derniers soient accessibles en ligne par tous les membres du groupe.
De plus, nous nous sommes servis de DotUML pour la réalisation des diagrammes en ligne.

Présentez vos outils de gestion de projet

Pour ce projet, nous avons utilisé différents outils, notamment Trello qui a été la pierre angulaire de notre projet.
Cet outil nous a permis de stocker tous les fichiers qui ne correspondent pas à du code. Que ce soit les schémas, les différents textes, le diagramme de Gantt. Nous avons également créé une section contenant une checklist pour chaque classe de notre application afin de suivre plus facilement l'état de l'avancement du projet et des tâches à réaliser.

Pour simplifier la gestion du code du projet, des tests et de l'utilisation des différentes dépendances nécessaires, nous avons construit notre application sous forme d'un projet Maven.
Maven permet non seulement de gérer les cycles de vie du projet, les tests, la compilation, l'exécution et la transformation en une archive Jar, mais aussi nous évite les difficultés liées aux changements de plateforme (OS) entre les machines de l'IUT (Linux) et nos machines personnelles (Windows).

Enfin, nous avons utilisé Docker afin de créer une base de données locale, nous ayant permit de réaliser les tests liés à la base de données sur notre temps privé.

Présentation de Gitlab

Nous avons utilisé Git associé à Github afin de sauvegarder notre code, d'assembler les différents travaux des membres, et le rendre disponible en ligne, mais également tracer un historique de ce qui à été réalisé.
Ainsi, nous avons créé un repository sur github en ligne, puis nous avons copié ce repo sur nos machines afin que l'on puisse accéder à ce répertoire.
De plus, chaque membre du groupe à codé sur sa propre branche, et les a ensuite envoyée sur github (git push), à partir duquel notre chef de groupe a pu intégrer ces modifications de manière propre sur la branche principale a l'aide des pull requests.

Bilan du groupe :

Ainsi, au cours de ce projet, nous avons travaillé sur l'implémentation des différentes demandes pour l'application à savoir une application servant à la fois d'outil pour les vendeurs et les administrateurs afin de gérer les librairies ainsi que leurs stocks mais également une application servant aux utilisateurs comme plateforme d'achat de livres en ligne.
Nous avons donc implémenté plusieurs classes à savoir une application permettant l'affichage de l'application dans le terminal, un vendeur permettant à un vendeur réaliser toutes les actions qu'il peut réaliser durant sa journée en tant que vendeur, un panier représentant le panier d'un client en ligne ou dans un magasin afin de se servir de ce dernier dans d'autres méthodes d'autres classes afin de simplifier les choses.
Un client permettant de réaliser toutes les actions qu'un client aurait besoin. Le principe est le même pour la suite des classes.
Toutes nos classes fonctionnent sauf la classe Vendeur qui est codé mais dont les tests ne passent pas.
En ce qui concerne l'organisation comme précisé dans les parties précédentes nous nous sommes organisés principalement grâce à trello, des groupes de discussion et Git.

Annexe

Diagramme de Gantt

Sae Java

Nom de la société
Chef de projet

DIAGRAMME DE GANTT SIMPLE par Vertex42.com
https://www.vertex42.com/ExcelTemplates/simple-gantt-chart.html

Début du projet : Mon, 3/24/2025
Semaine d'affichage : 1

TÂCHE	ATTRIBUÉE À	AVANCEMENT	Durée	DÉBUT	FIN
SetUp Trello et énumération des fonctionnalité	Groupe	0%	2	24/3/25	26/3/25
Création du diagramme de classe	Quentin/Nicolas	0%	25	26/3/25	20/4/25
Création du diagramme de séquence Commander	Quentin	0%	37	4/4/25	11/5/25
Création du diagramme de séquence onVousRecommande	Quentin	0%	37	4/4/25	11/5/25
Création du diagramme de séquence Faire Facture	Quentin	0%	37	4/4/25	11/5/25
Analyse de la Base de Données et modifications	Nicolas	0%	28	4/4/25	2/5/25
Réalisation Diagramme de Gantt	Quentin	0%	6	21/4/25	27/4/25
Tâche 5		0%	0	27/4/25	27/4/25
Developpement					
Créer la base de classes Client,Livre,Librairie,Panier	Quentin	0%	6	28/4/25	4/5/25
Créer les classes Commande et DetailCommande	River	0%	3	28/4/25	1/5/25
Créer la classe vendeur	Nicolas	0%	6	28/4/25	4/5/25
Créer les fonctions de base du réseau	Jolan	0%	14	5/5/25	19/5/25
Créer les fonctions du réseau connecté a la BD	Quentin	0%	7	20/5/25	27/5/25
Créer les menus et l'IU terminal	Quentin/Nicolas	0%	14	5/5/25	19/5/25
Vérification et rapport					
Relecture et vérification du code et de la documentation	Quentin/Nicolas		2	28/5/25	30/5/25
Finalisation du rapport	Quentin/Nicolas		5	31/5/25	5/6/25
Ecrire le manuel d'utilisation	River/Jolan		8	31/5/25	8/6/25