Clément Moisan

Tifany Meunier

Flavio Bouton

Nolhan Dolou

Rapport Collectif SAE JAVA

1) Introduction :

Le projet Livre Express consiste à développer une application informatique pour une entreprise fictive, le groupe de librairies Livre Express. Cette entreprise regroupe 7 librairies en France et permet à ses clients d'acheter des livres en magasin ou en ligne, avec des options de retrait en boutique ou de livraison à domicile. L'objectif est de concevoir, coder, tester et intégrer une solution complète qui répondra aux besoins des clients, des vendeurs et des administrateurs.

L'application permettra aux clients de passer des commandes, de choisir leur mode de réception (retrait ou livraison), et de consulter le catalogue des livres disponibles. Les vendeurs pourront gérer les stocks, ajouter des livres, mettre à jour les quantités, et vérifier la disponibilité des livres dans leur librairie ou ailleurs. Ils pourront aussi passer des commandes pour les clients en magasin et transférer des livres d'une autre librairie si nécessaire. Les administrateurs, quant à eux, pourront créer des comptes pour les vendeurs, ajouter de nouvelles librairies au réseau, gérer les stocks globaux et consulter les statistiques de vente.

L'application inclura également des fonctionnalités supplémentaires, comme des recommandations personnalisées pour les clients, basées sur leurs achats précédents. L'interface en ligne sera organisée avec des menus et sous-menus clairs, et l'application gérera les erreurs de saisie pour offrir une expérience utilisateur fluide.

Dans ce rapport de projet, on commencera par une courte introduction dans laquelle on présente le projet ainsi que ce que l’on nous à demandé de faire dans ce rendu. Ensuite, on présente l’analyse de la base de données : on décrit comment elle a été construite, les choix qu’on a fait pour insérer les données, les principales requêtes qu’on a utilisées, et s’il y a eu des changements, on explique pourquoi. Après ça, on s’intéresse à l’analyse UML de l’application : on montre comment on a organisé notre projet avec le diagramme de classe, pourquoi on a choisi de faire comme ça, les problèmes qu’on a rencontrés et comment on les a réglés. On passe ensuite à la partie sur l’implémentation, où on explique comment on a développé le cœur du projet. Puis, on présente les interfaces utilisateurs en expliquant nos choix pour rendre l’application pratique et agréable à utiliser. On parle aussi de l’organisation du travail en équipe : comment on s’est réparti les tâches, quels outils on a utilisé pour s’organiser et comment on a collaboré tout au long du projet. Et pour finir, on fait un bilan du projet en équipe, en disant ce qui a bien marché, ce qui a été plus compliqué, et ce que chacun a appris pendant cette expérience.

2) Analyse de la base de données :

Les bases de données sont essentielles dans tout projet numérique visant à organiser et exploiter efficacement de grandes quantités d'informations. Elles permettent de structurer les données, d'automatiser les recherches et d'offrir des services personnalisés aux utilisateurs. Dans un projet comme celui de LivreExpress, elles jouent un rôle clé dans la gestion des livres et des profils clients pour améliorer l'expérience globale.

2.1) Description Du MCD :

Un MCD (Modèle Conceptuel de Données) est une représentation visuelle et logique des données d'un projet. Il montre les différentes entités (comme "Livre" ou "Client"), leurs propriétés, et les relations entre elles. C'est une étape clé pour bien organiser les données avant de créer la base et avoir une représentation visuelles afin de bien manipuler les données.

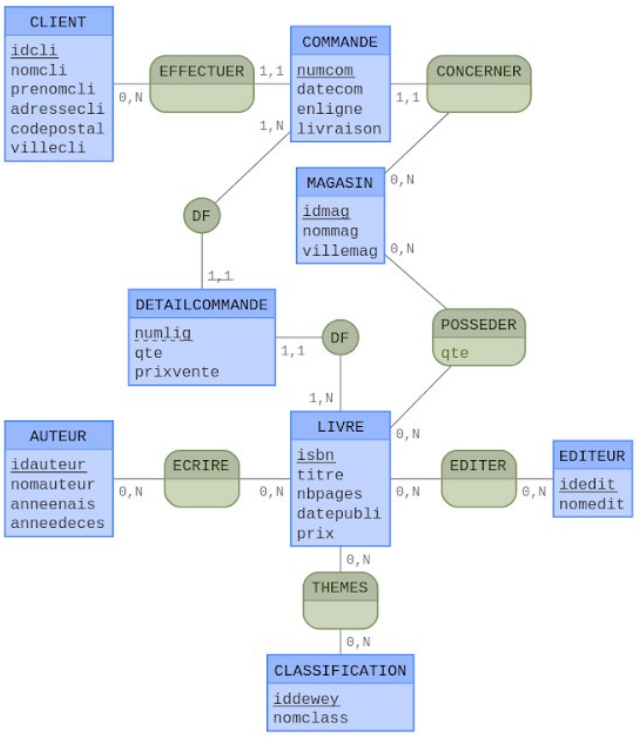


FIGURE 1 – MCD de Livre Express

Le MCD de la base de données ci-dessus est celui que nous allons devoir utiliser pour la SAE.

Elle contient 8 tables:

- Client
- Commande
- Magasin
- DetailCommande
- Livre

- Auteur
- Editeur
- Classification

et 4 tables-association caractérisées par les cardinalités (0,N) de chaque côté de l'association :

- Editer
- Ecrire
- Themes
- Posseder

A partir de ses informations et des autres cardinalités entre les différentes tables, on peut en déduire le MLD suivant:

CLIENT(idcli, nomcli, prenomcli, adressecli, codepostal, villecli)
COMMANDE(numcom, datecom, enligne, livraison, #idcli)
DETAILCOMMANDE(numlig, numcom*, isbn*, qte, prixvente)
LIVRE(isbn, titre, nbpages, datepubli, prix, #idedit, #iddewey)
AUTEUR(idauteur, nomauteur, annenaiss, annedeces)
ECRIRE(#idauteur, #isbn)
EDITEUR(idedit, nomedit)
CLASSIFICATION(iddewey, nomclass)
MAGASIN(idmag, nommag, villemag)
POSSEDER(#idmag, #isbn, qte)
CONCERNER(#numcom, #idmag)

Légende:

CLE_PRIMAIRE

#CLE_ETRANGERE

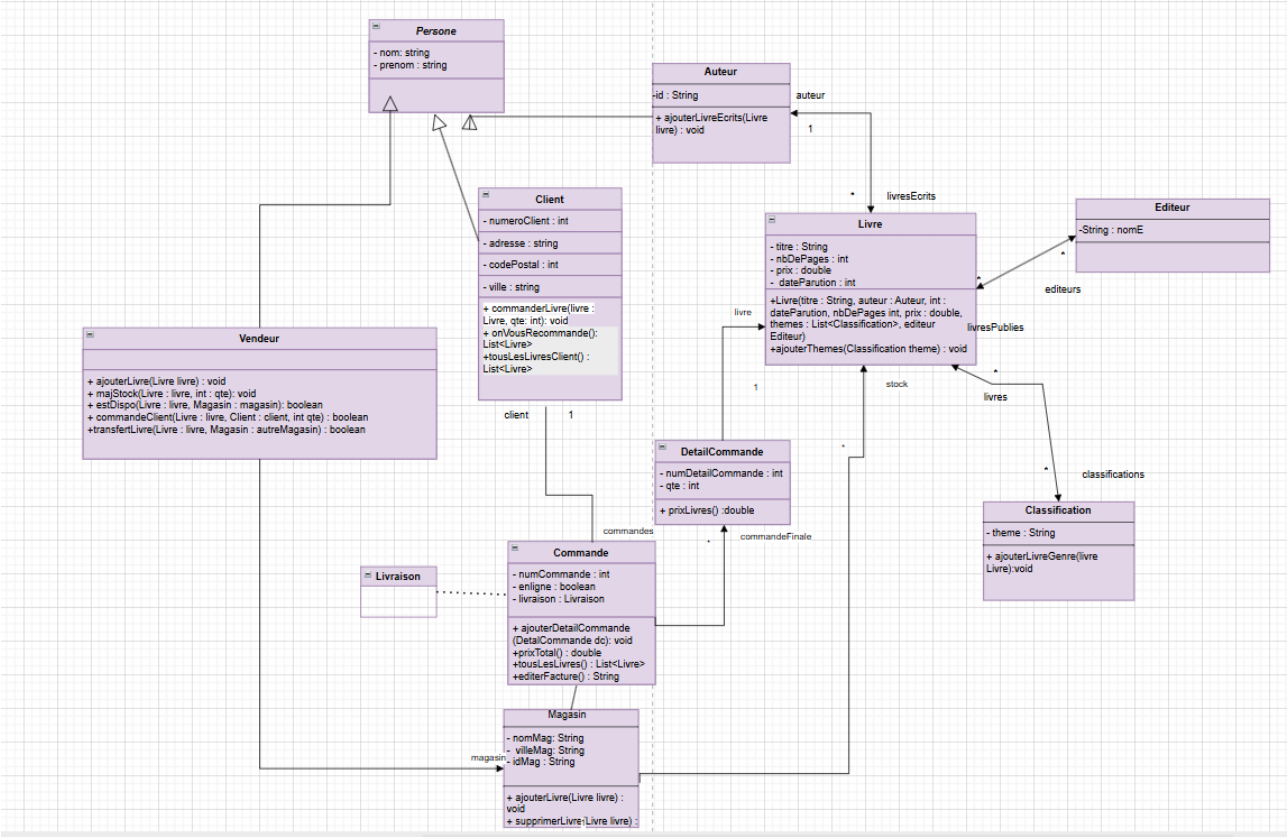
3) Analyse UML de l’application :

Avant de développer une application, il est essentiel de bien comprendre les besoins fonctionnels et la structure des données à manipuler. Pour cela, on utilise l'UML (Unified Modeling Language), un langage de modélisation standardisé permettant de représenter visuellement les composants du système.

L’analyse UML permet :

- de clarifier les rôles des différentes entités (classes),
- de définir les relations entre elles (associations, agrégations, compositions...),
- de repérer les responsabilités de chaque classe (attributs, méthodes),

- de faciliter la conception orientée objet avant la phase de codage.



Lecture détaillée du diagramme :

1. Classe Personne

- Classe générique avec des attributs communs : nom, prenom.
- Classe abstraite de Client et Auteur, ce qui évite la duplication de données personnelles.

2. Classe Client (hérite de Personne)

- Attribut : numClient.
- Méthodes :
    - commanderLivre() : permet de passer une commande.
    - consulterCatalogue() : retourne une liste de livres disponibles.
    - OnVousRecommande() : retourne les livre recommandés de ce client

3. Classe Auteur (hérite de Personne)

- Aucun attribut ou méthode supplémentaire précisé ici, mais l'héritage est pertinent pour une éventuelle évolution.

4. Classe Livre

- Attributs : titre, nbDePages, prix, datePublication.

- Reliée à :
    - Auteur : un livre a un auteur.
    - Editeur : un livre a un éditeur.
    - Classification : pour organiser les livres par thème.
    - DetailCommande : chaque ligne de commande fait référence à un livre.

5. Classe Editeur

- Attributs : nomE, livresPublies.
- Méthodes pour accéder et modifier ces attributs.

6. Classe Classification

- Sert à catégoriser les livres par thème.
- Attribut : theme.

7. Classe Vendeur (associée à Livre)

- Méthodes :
    - ajouterLivre() : ajoute un livre avec ses détails.
    - majStock() : met à jour les quantités.
    - estDispo() : vérifie la disponibilité.
    - commandeClient(Client) : prépare une commande de livres pour un client
    - transfertLivre(Livre,Magasin) : transfère un livre de ce magasin vers un autre

8. Classe Commande

- Attribut : numCommande. Enligne, livraison
- Liée à :
    - Client (qui commande),
    - Magasin (où la commande est passée),
    - DetailCommande (les lignes de commande).
    - Livraison (mode de livraison)

9. Classe DetailCommande

- Classe intermédiaire entre Commande et Livre.
- Sert à gérer le contenu d’une commande (quantité, livre, etc.).

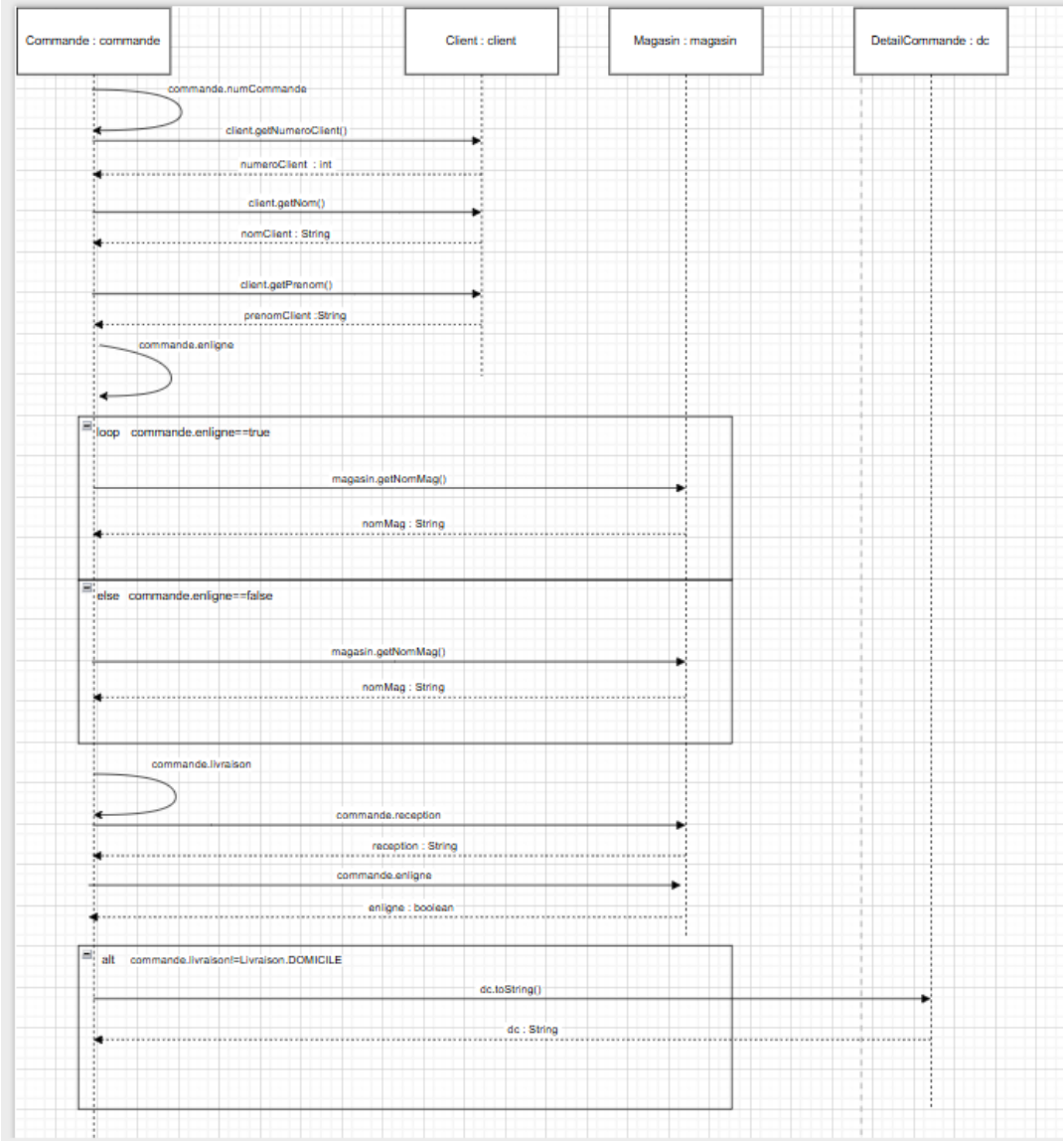
10. Classe Magasin

- Attributs : nomMag, villeMag.
- Liée à Commande

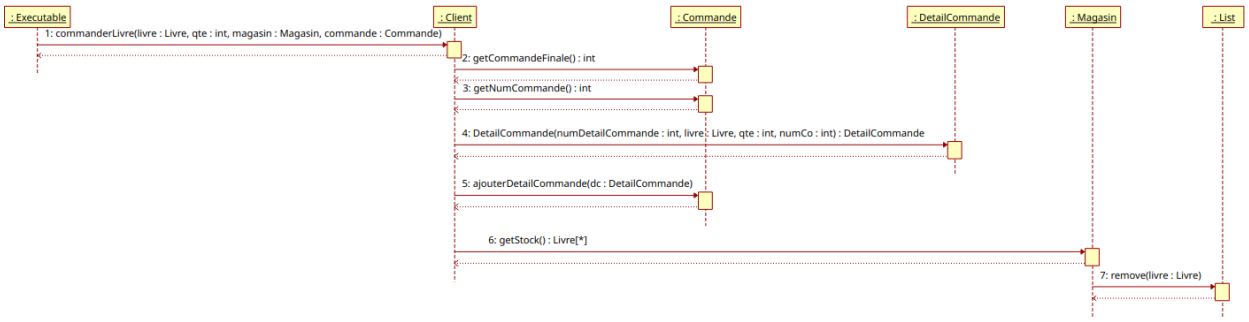
Justification des choix de conception :

- Héritage : Client et Auteur héritent de Personne → cela centralise les données personnelles, ce qui facilite la maintenance.
- Composition avec DetailCommande : un bon choix pour gérer les lignes de commande (quantité, livre, etc.) de manière claire.
- Séparation des responsabilités : les classes ont un rôle bien défini :
    - Vendeur gère le stock,
    - Client commande et parcours la Bibliothèque,
    - Livre contient les informations du produit,
    - Commande structure les achats et les répertories afin d'avoir un suivie aussi bien pour le Client que pour les Magasins,
    - Classification organise les livres par genre.
    - Relations bien orientées : les cardinalités montrent clairement les dépendances entre objets (1..n, etc.) nous indiquant lors du code les classes qui ont besoin d'informations dans d'autre classe et combien 1 ou plusieurs(n).

Voici le diagramme de séquence de la méthode editerFacture():



Voici le diagramme de séquence de la méthode commanderLivre():



4) Implémentation :

Pour ce qui est de l'implémentation, il y a plusieurs fonctions importantes (autres que les fonctions essentielles comme les constructeurs). On a choisi de vous présenter 3 d'entre elles.

Tout d'abord, il y a la fonction editerFacture. Cette fonction permet comme elle le dit d'éditer une facture en fonction d'une commande. Cette facture va contenir plusieurs informations : nom du client, son numéro, son adresse, puis le numéro de commande, son contenu, son prix, etc… Nous avons choisi de l'implémenter dans la classe Commande. Pour la représenter, on fait un print dans le terminal de plusieurs chaînes de caractères misent bout à bout.

Voici le rendu :

```
---------
Commande n°1

Numéro de client : 1
Client : Dupont Jean

Commande en ligne sur le magasin Librairie de Paris
Livraison à domicile
Adresse de livraison: 222 rue de Bourgogne
Commande en ligne: Oui

Détails de la commande:
numDetailCommande 0, livre Les Trois Mousquetaires, qte 2, numCo 1
numDetailCommande 1, livre Les Misérables, qte 5, numCo 1
Prix total: 86.93 €

---------
```

Ensuite, nous avons aussi implémenter la fonction pour commander que nous avons appelé commanderLivre. Cette méthode est mise dans la classe Client. Elle ressemble à ceci :

```java
public void commanderLivre(Livre livre, int qte, Magasin magasin, Commande commande){
    // On vérifie si le livre est disponible dans le magasin
    if (!magasin.getStock().containsKey(livre)) {
        System.out.println("Le livre n'est pas disponible dans le magasin.");
        return;
    }
    // On vérifie si la quantité demandée est disponible
    if (magasin.getStock().get(livre) < qte) {
        System.out.println("La quantité demandée n'est pas disponible.");
        return;
    }
    DetailCommande detailCommande = new DetailCommande(commande.getCommandeFinale().size()+1,livre, qte, commande.getNumCommande());
    commande.ajouterDetailCommande(detailCommande);
    magasin.getStock().remove(livre);
}
```

Elle permet au client d'ajouter un livre à sa commande, en choisissant la quantité qu'il veut, mais aussi le Magasin dans lequel il commande.

Enfin, une des fonctions principale que nous avons dû implémenter est la fonction onVousRecommande qui est expliquée dans le code.

```java
public List<Livre> onVousRecommande(Client client, List<Commande> toutesLesCommandes){
    List<Livre> recommandations= new ArrayList<>();
    Iterator<Commande> it = toutesLesCommandes.iterator();
    while(it.hasNext()){
    Commande commande = it.next();
    if(!(client.commandes.contains(commande))){//si la commande qu'on regarde n'est pas celle du client
    for(Livre livre : client.tousLesLivresClient()){// on parcourt chaque livre que le client a commandé
    if(commande.tousLesLivres().contains(livre)){// si dans la commande qu'on regarde il y a le livre que le client a commandé
    // (il faut donc lui recommander les livres de cette commande)
    for(Livre livreCommande : commande.tousLesLivres() ){// pour chaque livre de la commande
    if(!(client.tousLesLivresClient().contains(livreCommande))&& !(recommandations.contains(livreCommande))){
    //si le client n'a jamais commandé ce livre et qu'il n'est pas dans les recommandations
    recommandations.add(livreCommande);// l'ajouter aux recommandations
                        }
                    }
                }
            }
        }
    }
    return recommandations;
}
```