

Livrable 2

Groupe 4



RAJOELISOA Enorian | KOCHER Vania | SCHNEPP Anthony | HOFSTETTER Léo

07 octobre 2023

EVAL

Table des matières

[**I.** **Présentation de l’équipe** 2](#_Toc152676455)

[**II.** **Contexte** 3](#_Toc152676456)

[**III.** **Modélisation de la base de données** 3](#_Toc152676457)

[**V.** **Environnement de travail** 21](#_Toc152676458)

[**VI.** **Conclusion** 22](#_Toc152676459)

# **Présentation de l’équipe**

Une image contenant Visage humain, personne, sourire, habits

Description générée automatiquement

Chef du projet

Une image contenant Visage humain, verres, sourire, habits

Description générée automatiquement

RAJOELISOA Enorian

KOCHER Vanita

Une image contenant Visage humain, personne, cou, Menton

Description générée automatiquementUne image contenant Visage humain, personne, habits, Menton

Description générée automatiquement

SCHNEPP Anthony

Mark

HOFSTETTER Léo

# **Contexte**

L'objectif de ce projet est de réaliser une architecture de type client-serveur composée d’une application et d’une base de données. Et ce sont nous, l’entreprise « EVAL » qui doit réaliser celui-ci.

Ce projet se compose de deux phases distinctes, chacune jouant un rôle essentiel dans l'atteinte de nos objectifs. La première phase est axée sur l'acquisition approfondie du cahier des charges, la constitution de l'équipe de travail et son organisation, et enfin, la modélisation tant logicielle que des données. C'est dans cette étape préliminaire que nous poserons les bases solides nécessaires à la réussite du projet, en nous assurant de bien comprendre les exigences du client, en formant une équipe collaborative et en concevant une structure logicielle robuste.

La deuxième phase du projet constituera le passage de la conception à la réalisation concrète. Dans cette partie, nous aurons l’opportunité de réaliser ce qui a été demandé dans le cahier des charges, c’est-à-dire réaliser une application graphique qui sera reliée à une basse de données, et dans cette application nous devons être capables de réaliser des requêtes spécifiques qui ont été listées dans le cahier des charges.

# **Modélisation de la base de données**

Avec notre équipe, nous avons décidé de commencer cette partie de modélisation, en démarrant avec la réalisation de la modélisation entière de tout ce qui est relié à la base de données. Dans cette partie nous allons donc réaliser :

* Le Dictionnaire de Données
* Le MCD
* Le MLD

Nous allons donc commencer la modélisation en réalisant ce qui est appelé le « Dictionnaire de Données ». Nous allons premièrement définir ce qu’est un Dictionnaire de Données.

Un dictionnaire de données vise à présenter de manière complète toutes les informations requises pour le fonctionnement optimal d'une base de données, structurées de façon similaire à un tableau. On peut le décrire comme une compilation de métadonnées permettant de mieux comprendre différentes variables qui vont être utilisées dans la base de données. Celui-ci permet d’éclaircir le contexte d'une base de données et fournissant les éléments essentiels à son interprétation. Cette documentation simplifie la gestion des bases de données, offrant aux administrateurs et aux utilisateurs une compréhension intuitive de la structure de leurs bases de données.

Il sera donc structuré en plusieurs colonnes, comprenant notamment :

* Le nom du paramètre.
* Le type du paramètre, que ce soit numérique, alphanumérique, alphabétique, date, booléen, etc.
* La taille du paramètre, choisie de manière arbitraire pour garantir un fonctionnement optimal.
* La valeur minimale du paramètre (généralement non spécifiée).
* La valeur maximale du paramètre (généralement non spécifiée).
* Des commentaires, permettant d'ajouter des informations supplémentaires ou de décrire simplement le paramètre.

Voici un exemple de dictionnaire de données :

Une image contenant texte, capture d’écran, nombre, Police

Description générée automatiquement

Afin de pouvoir réaliser un bon dictionnaire de données, nous avons dû, avec notre équipe décomposer les différentes attentes du projet et réaliser plusieurs brouillons afin de vraiment comprendre les différentes relations que nous allons devoir réaliser dans le MCD à suivre ou même dans les prochains diagrammes que nous allons réaliser par la suite. Après avoir réellement bien analysé le projet nous avons donc pu commencer à réaliser notre Dictionnaire de Données.

Voici donc le dictionnaire de données que l’on a pu réaliser dans le cadre du projet :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nom** | **Type** | **Taille** | **Min** | **Max** | **Commentaires** |
| **Id\_ville** | N |  | 1 |  | Auto-incrémenté |
| Nom\_ville | A | 50 |  |  | Nom de la ville |
| Code\_postal | A | 10 |  |  | Code postal |
| **Id\_adresse** | N |  | 1 |  | auto-incrémente |
| Numero\_rue | A | 16 bits |  |  | Numéro de rue |
| Nom\_rue | A | 50 |  |  | Nom de rue |
| **Id\_personne** | N |  | 1 |  | Auto-incrémenté |
| Nom | A | 50 |  |  | Nom |
| Prenom | A | 50 |  |  | Prénom |
| Date\_naissance | Date |  |  |  | format DD-MM-YYYY |
| Date\_premier\_achat | Date |  |  |  | format DD-MM-YYYY |
| Date\_embauche | Date |  |  |  | format DD-MM-YYYY |
| **Reference\_commande** | AN | 30 | 1 |  | Auto-incrémenté |
| Date\_livraison | Date |  |  |  | format DD-MM-YYYY |
| Date\_emission | Date |  |  |  | format DD-MM-YYYY |
| **Id\_paiement** | N |  | 1 |  | Auto-incrémenté |
| Moyen\_paiement | A | 30 |  |  | Moyen de paiement |
| Date\_paiement | Date |  |  |  | format DD-MM-YYYY |
| Date\_solde\_enregistre | Date |  |  |  | format DD-MM-YYYY |
| **Id\_remise** | N |  | 1 |  | Auto-incrémenté |
| Pourcentage\_remise | N | 8 bits |  |  | Pourcentage de la remise |
| **Reference\_article** | AN | 30 | 1 |  | Auto-incrémenté |
| Nom\_article | A | 50 |  |  | Nom de l'article |
| Prix\_article\_HT | N | Monétaire |  |  | Prix de l'article HT |
| Nature\_article | A | 20 |  |  | Nature de l'article |
| Couleur\_article | A | 20 |  |  | Couleur de l'article |
| Seuil\_reapprovisionnement | N | 16 bits |  |  | Quantite max de produits avant réapprovisionnement |
| Quantite\_article | N | 16 bits |  |  | Quantite de chaque article en stock |
| Quantite\_article\_commande | N | 16 bits |  |  | Quantite de chaque article dans la commande |
| Pourcentage\_remise\_article | N | 8 bits |  |  | Pourcentage de la remise de chaque article |
| **Id\_taxe** | N |  | 1 |  | Auto-incrémenté |
| Pourcentage\_taxe | N | 8 bits |  |  | Pourcentage de la taxe |

Ce qui est en gras dans le dictionnaire va représenter ce qui est appelé « clé primaire ». Pour faire simple, une clé primaire est un paramètre qui va permettre d’identifier d’autres paramètres. Par exemple si nous avons une personne base de données qui stocke des personnes. On peut avoir quelqu’un qui s’appelle Pierre mais il peut avoir une centaine de personnes qui portent le même nom, c’est pourquoi par exemple on va associer un identifiant à chaque personne, par exemple Pierre Gut est associé au numéro 1 et Pierre Paul au numéro 2, et donc cela permet de bien différencier les deux personnes.

Après avoir réalisé le dictionnaire de données qui va stocker tous les différents paramètres qui sont, soit, demandées dans le projet, soit, directement déduis par nous, nous pouvons alors réaliser le MCD associé à celui-ci. Un MCD ou Modèle conceptuel de données, est une représentation graphique de la base de données. Celui-ci va inclure les différentes entités et les différentes relations qui vont être nécessaires pour une bonne base de données. Afin que celui-ci soit bien réalisé, et donc que l’on puisse l’utiliser comme tel nous devons appliquer lors de la création de celui-ci ce qui est appelé, « les 3 formes normales ».

Lorsque nous devons réaliser un MCD il y a une sorte de « règle » qui permet justement de garder une bonne modélisation de MCD. Il y a :

* La 1ère forme normale
* La 2ème forme normale
* La 3ème forme normale

Nous nous devons donc d’appliquer ces 3 formes normales afin de garder un bon MCD. Nous allons donc vous décrire ces 3 formes normales.

**La 1ère forme normale :**

Définition :

Une table est en 1NF si elle ne contient que des valeurs atomiques et qu'il n'y a pas de groupes de valeurs ou de répétitions dans les colonnes.

Exemple :

Considérons une table "Commandes" avec les colonnes suivantes :

| **ID\_Commande** | **Produits** | **Quantités** |
| --- | --- | --- |
| 1 | Pommes, Oranges | 10, 15 |
| 2 | Bananes | 20 |

Cette table n'est pas en 1NF car la colonne "Produits" contient des valeurs multiples. Pour corriger cela, vous pourriez diviser cette table en deux : "Commandes" et "Détails\_Commandes".

**La 2ème forme normale :**

Définition :

Une table est en 2NF si elle est en 1NF et si toutes les colonnes non-clés dépendent intégralement de la clé primaire.

Exemple :

Prenons une table "Commandes" avec les colonnes suivantes :

| **ID\_Commande** | **ID\_Produit** | **Nom\_Produit** | **Quantité** |
| --- | --- | --- | --- |
| 1 | 101 | Pommes | 10 |
| 1 | 102 | Oranges | 15 |
| 2 | 103 | Bananes | 20 |

La colonne "Nom\_Produit" dépend seulement de l'ID\_Produit, qui est une partie de la clé. Pour atteindre la 2NF, vous pourriez séparer cette information dans une autre table "Produits".

**La 3ème forme normale :**

Définition :

Une table est en 3NF si elle est en 2NF et si aucune colonne non-clé ne dépend d'une autre colonne non-clé.

Exemple :

Reprenons la table "Commandes" après avoir appliqué la 2NF :

| **ID\_Commande** | **ID\_Produit** | **Quantité** |
| --- | --- | --- |
| 1 | 101 | 10 |
| 1 | 102 | 15 |
| 2 | 103 | 20 |

Supposons qu'on ajoute une colonne "Prix\_Unitaire" qui dépend de "ID\_Produit" mais n'est pas directement liée à la clé primaire. Pour atteindre la 3NF, vous pourriez créer une nouvelle table "Produits" avec "ID\_Produit" et "Prix\_Unitaire".

Après explications des 3 formes normales, nous avons donc pu réaliser notre MCD qui est ci-dessous :

Une image contenant texte, capture d’écran, diagramme, ligne

Description générée automatiquement

Pour la partie concernant la modélisation de la base de données, il ne nous reste plus qu’à réaliser le MLD ou Modèle Logique de Données.

Un MLD, ou Modèle Logique de Données, représente la structure organisationnelle des données au sein d'une base de données de manière indépendante du système de gestion de base de données (SGBD) spécifique. Contrairement au modèle conceptuel, qui se concentre sur la vue globale des données et de leurs relations, le MLD se penche sur les détails spécifiques de la manière dont les données seront stockées et organisées dans une base de données relationnelle.

En utilisant l’application « Looping » qui permet de réaliser des MCD, on peut alors obtenir directement le MLD associé au MCD, c’est donc ce qu’on a pu réaliser. Voici alors notre MLD associé à notre MCD :

Une image contenant texte, capture d’écran, diagramme, Parallèle

Description générée automatiquement

Celui-ci nous permet donc de bien voir les différentes relations entre les entités. Par exemple entre l’entité « Adresses » et « Villes », cela nous indique bien que la clé primaire de l’entité « Villes » va se retrouver comme clé étrangère dans l’entité « Adresses ». En effet une adresse est située dans une ville, cela ne marche pas dans l’autre sens. On a donc pu réaliser le même système pour toutes les autres entités et on se retrouve alors avec le MLD correspondant.

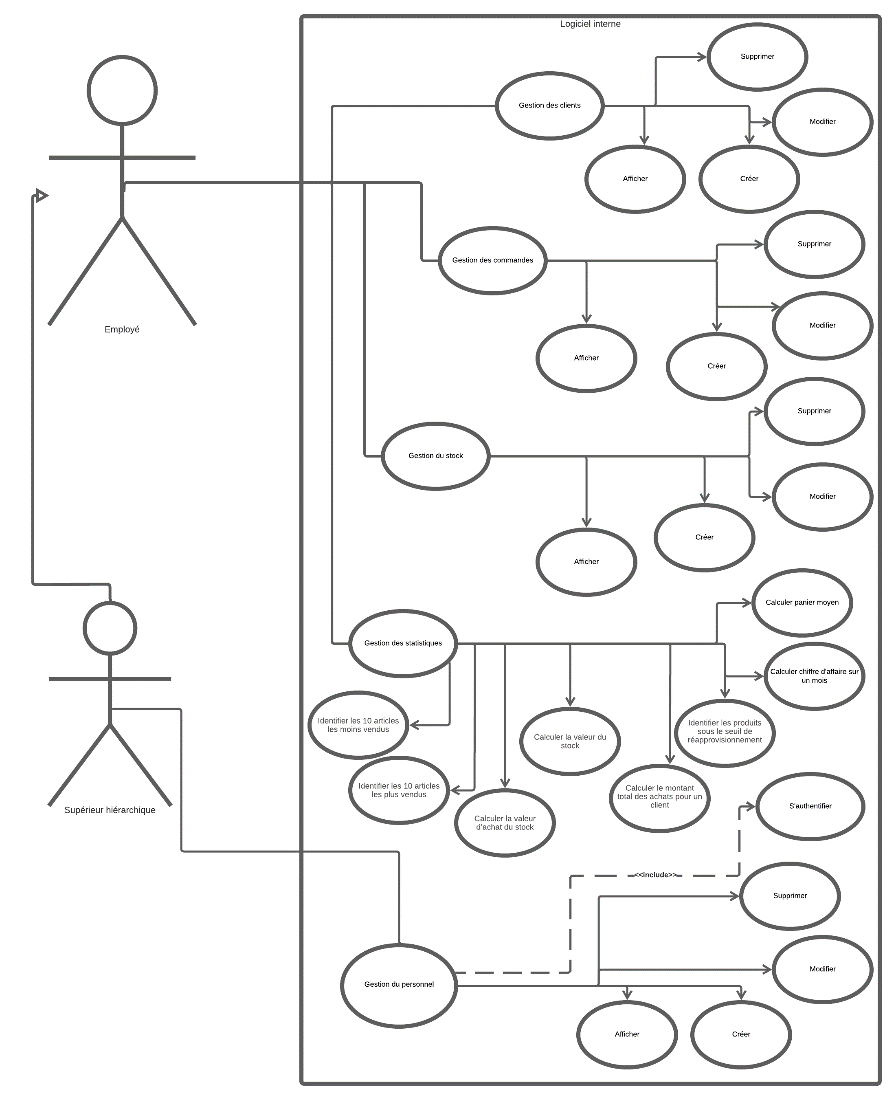
1. **Modélisation du système**

Durant cette partie nous allons nous pencher sur la réalisation des différents diagrammes qui vont porter cette fois-ci sur le système en entier. Dans cette partie nous allons réaliser 4 diagrammes qui vont être :

* Diagramme de cas d’utilisation
* Diagramme de séquences
* Diagramme d’activité
* Diagramme de classes

Nous allons premièrement nous concentrer sur le diagramme de cas d’utilisation. Ce diagramme présentera les différentes manières dont un utilisateur peut interagir avec le système. Il définira les principales fonctionnalités offertes par celui-ci et comment elles répondent aux différents besoins des utilisateurs.

Voici donc la représentation de celui-ci (zoomer pour bien voir) :



Si vous n’arrivez pas à visualiser celui-ci au complet cliquez sur ce lien :

<https://ibb.co/v12dNnz>



Ce diagramme permet donc de montrer les différentes choses que peut réaliser par exemple un employé basique, ou encore les permissions d’un supérieur hiérarchique. Par exemple un supérieur hiérarchique va hériter de tout ce que peut faire un employé, mais va avoir, en plus, le fait de pouvoir gérer justement ces employés, ceci inclut cependant qu’il va devoir s’identifier via l’application pour bien montrer qu’il est un supérieur.

On peut passer ensuite à la réalisation des diagrammes de séquences. Dans le projet on nous demande de réaliser le diagramme de séquences des fonctions :

* Créer un personnel
* Modifier un personnel
* Supprimer un personnel
* Afficher un personnel

Avant de commencer, un diagramme de séquence est un diagramme UML (Unified Modeling Language) qui représente la séquence de messages entre les objets au cours d'une interaction. Un diagramme de séquence comprend un groupe d'objets, représentés par des lignes de vie, et les messages que ces objets échangent lors de l'interaction.

Les diagrammes de séquence représentent la séquence de messages transmis entre des objets. Ils peuvent également représenter les structures de contrôle entre des objets. Par exemple, les lignes de vie dans un diagramme de séquence pour un scénario de banque peuvent représenter un client, un guichetier ou un responsable d'agence. Les communications entre le client, le guichetier et le responsable sont représentés par les messages entre ces derniers. Le diagramme de séquence représente les objets et les messages entre ces objets.

Nous allons donc en expliquer un en détail et le reste sera sensiblement la même chose et ne nécessitera pas plus d’explications. Voici celui pour la création d’un personnel :

Une image contenant texte, capture d’écran, diagramme

Description générée automatiquement

Pour expliquer en détail, voici le processus de création d’un personnel :

1. L'utilisateur déclenche l'ajout d'un personnel en interagissant avec l'application.
2. La boucle "mode ajout personnel" commence, indiquant un processus itératif pour collecter les informations nécessaires.
3. L'application sollicite l'utilisateur pour fournir les informations requises sur le personnel.
4. L'utilisateur transmet les informations du personnel à l'application.
5. L’application va interagir avec les différentes classes nécessaires à la création
6. Les données sont envoyées à la BDD
7. Il y a une vérification des données
   * Si le personnel est déjà présent dans la BDD, l'application signale une erreur à l'utilisateur.
   * Si le personnel n'est pas encore enregistré dans la BDD, l'application procède à l'ajout dans la BDD et informe l'utilisateur du succès de l'opération.
8. Si les informations fournies par l'utilisateur sont incorrectes, l'application signale une erreur à l'utilisateur.
9. Le processus de boucle "mode ajout personnel" se répète jusqu'à ce que des informations valides soient fournies.

Le diagramme est assez simple à comprendre, il suffit juste de savoir ce que représente le « loop » et le « alt ». Pour expliquer ceci, le « loop » signifie « boucle » et donc qu'après le choix du mode « ajouter personnel » on rentre dans une boucle qui est infinie jusqu’à que l’ajout du personnel se soit bien passé. Le « alt » quant à lui, peut être traduit par une condition. Par exemple dans notre diagramme nous avons un « alt » avec comme paramètre « informations correctes » cela signifie que si les informations sont correctes, alors on peut continuer l’exécution du programme.

Voici ensuite le diagramme de séquences pour le cas de la modification d’un personnel :

Une image contenant texte, capture d’écran, diagramme

Description générée automatiquement

Le fonctionnement est exactement le même. Voici ensuite le diagramme de séquences pour la suppression d’un personnel :

Une image contenant texte, capture d’écran, diagramme

Description générée automatiquement

Le diagramme change un petit peu mais va garder toujours le même fonctionnement. Et voici enfin le dernier diagramme de séquences, qui porte sur l’affichage d’un personnel :   
Une image contenant texte, capture d’écran, diagramme

Description générée automatiquement

Si vous souhaitez visualiser en grand les diagrammes, voici les différents liens qui vont vous rediriger vers le site « mermaid », c’est avec ce site que nous avons pu réaliser ces différents diagrammes.

Les liens des diagrammes :

* [Ajout personnel](https://mermaid.live/edit#pako:eNqdVMGO0zAQ_RXLF0BKoybdplkfKhXaAxIrwa64oErISqa7XhI7azsspeq_cKT8Rn6MSdpskzSFBZ8cz3vznseT2dBIxUAZNfCQg4xgLvit5ulSElwZ11ZEIuPSko9WJMJwC7k-Dc6yLBERt0LJ0-Cbd58XaZaodbHrDd48JB9y0OtzwSsu-S2g6h7QMDKYThvKjMzuVW5J_CKXJANtlJSQ7EmJUhlJ8aqEV5hOuFyNTANM3JBhZA4pl0iOwRAhV0qnFc6QOO9L9QeLb__O5oltwyKlNUQWzBHTNTydNqrMiIRHUn-9dF33VZvZwO6Z9ROU_gxo28-pUU-cw8swwqPSxbV6NGeZByxyX8_njCzkVyWwmljUhBON3Vf8skCKH-UnItop8OBU9LrYZVge6ORoM8tivq9rTOJid48SMceq9un02W093wJfov4Bzr1Ft3lmq5WI7kAT6CFDghc4GpTIz3SxM4Dt__82jwmrdq9_vH92nIIxmJ-YPIqKn53-AxkfD6p7tLpWyGf07bNr9SSGG-rQFFBHxDi4NuXxkto7SGFJGW5jrr8s6VJuEcdzq27WMqLM6hwcmmcxKh2GHGUrjrYdisOGsg39Rtml746Gw4sg9MLLYDQcew5dUzbwfXcyHk-CC8_3Aj_wx5OtQ78rhSk81xuWaxSMxmE4CkPfoRALq_TVfrJWA7bS-FQRKs3tby4Vw74)
* [Modification personnel](https://mermaid.live/edit#pako:eNqdVMGOmzAQ_RXLl7YSQYFsCOtDpG2TQ6VGane1lwqpsvBk1yrYrDHdplH-pcdNf4Mf60DCBhLSbusDwvZ78x4zw6xprAVQRnN4KEDFMJP8zvA0UgRXxo2Vscy4suTWykTm3EJhTi-vsiyRMbdSq9PLdx--zNMs0aty23t585B8KsCszl0uuOJ3gKo7QMvIYDptKTOy0EIu9zsiXhWKZGByrRQkO26idUZS_OLqcYAeoarVijtAmZYoIzNIucIYAnIi1VKbtMblRBR9of5g-P3f2TyxXVisjYHYQn7AHBueTls5Z0TBI2l2r13XfdNltrA7ZlMQRm4zgb77OQ3qmbOvEyM8rlxc68f8LHOPRe7b2YyRufqmJWYTk5pwYrAXy18WSPmz2iKiGwIPTkWvy22G6YGjGF1mlcyPTY6JwlRlptzmgL0mOOa2T63PdKeIc6xH81Ocq8hxC10tsfnuwRDoIUOCn3Gw-f_ODjF27d78gP_sMoU8RwGSF3FcPh11HihxOKi9d_pVqhd07Ivz8yyGL9ShKaCOFDjA1tVxRO09pBBRhq-Cm68RjdQGcbyw-malYsqsKcChRd3Y-2FH2ZKjbYfi0KFsTb9Tdum7o-HwIgi98DIYDceeQ1eUDXzfnYzHk-DC873AD_zxZOPQH1pjCM_1htUaBaNxGI7C0HcoCGm1WewmbD1oa43PNaHW3PwG26XF6g)
* [Suppression personnel](https://mermaid.live/edit#pako:eNqdVNFu2jAU_RXLL92kEJFQQuoHpG7wUGmVtqK9TEiTlVxaa4nt2s46hviXPZb9Rn5sN4GUAGHqZuUhts-55_j6-q5oolKgjFp4LEAmMBH83vB8LgkOzY0TidBcOvLZiUxY7qAwp5vXWmci4U4oebr5_sPXaa4ztSw3nZuzx-xTAWZ5bvOWS34PqLoFtIz0xuOWMiOzQmsD1uKEpBeFJBqMVVJCtqVmSmmS44GJ3SIr4BGmGq2gPdRoKTIygZxLjIBfdnEzIWnRFeEvJs9yeObIjVwok9dISxJlDCQO7B5z7G48bmWXEQlPpJm98X3_7SGzhd0ym9RXx8rAQTenQb1wdjfCCE8qF3fqyZ5l7rDIfTeZMDKV35UAW6ePE4NVV_52QMpf1RQRhyFw4VT0rtxoTA8cxThkVsn82OSYSEyVNuXGAlZVyjG3XWpdpg-ubor30ZT_uRs5rpfrxUIkD2AIdJAhw2Psbf6_s32MurJF3ry1f7aZ4_tBBYyTJOXzUemBTPcLtfmDghXyFSX76gS9iOEP9WgOqCNS7FWranlO3QPkMKcMf1Nuvs3pXK4RxwunZkuZUOZMAR4tdIpKu75G2YKjbY9if6FsRX9QdhX6g37_MoqD-Coa9IeBR5eU9cLQHw2Ho-gyCIMojMLhaO3Rn0phiMAP-tUYRINhHA_iOPQopMIpc7ttpnVPrTW-1IRac_0Hy7y80Q)
* [Affichage personnel](https://mermaid.live/edit#pako:eNqdVF1vmzAU_SuWX7pJBAVSCPVDpG7kodIqbY32MkWaLLhprYFNjVmXRfkve83-Bn9sF_IFBKZuFg9gn3PP4frobmikYqCM5vBcgIwgFPxR83QpCa6MayMikXFpyGcjEpFzA4W-PLzNskRE3AglLw_ff_g6T7NErctd7-HiOflUgF4PHd5zyR8BVfeAhpHRbNZQZuR2tRLRE4JJfFVIkoHOlZSQ7ImJUhlJ8XcJP-E6kGo1Ko5QoCHHSAgpl1gAn-TqLiRx0VfhLw4HOTwx5E6ulE5rZE4ipTVEBvIzputuNmu0lhEJL-T49ca27bdtZgO7Zx77zsgCEpTq5xxRJ87hOhjhUeXiQb3kg8wDFrnvwpCRufyuBOR1-zjRGLnytwFS_qo-EdEugRuXog_lLsP2QKdGm1k18-Oxx0RiqzJd7nLASMUce9un1me6dXVzvI9j9odupJuXfSBBE-ghQ4K_cbb5_84auRlI1z95FY0YdhzL-LxR229FVshXhPbVLTqJ4Qu1aAqoI2IcVZtqe0nNE6SwpAxfY66_LelSbhHHC6MWaxlRZnQBFi2yGJUOY42yFUfbFsXxQtmG_qDsxrUn4_G1HzjBjT8Ze45F15SNXNeeet7Uv3Zcx3d915tuLfpTKSzh2M64WhN_4gXBJAhci0IsjNL3-1laj9Ra40tNqDW3fwDzhrqC)

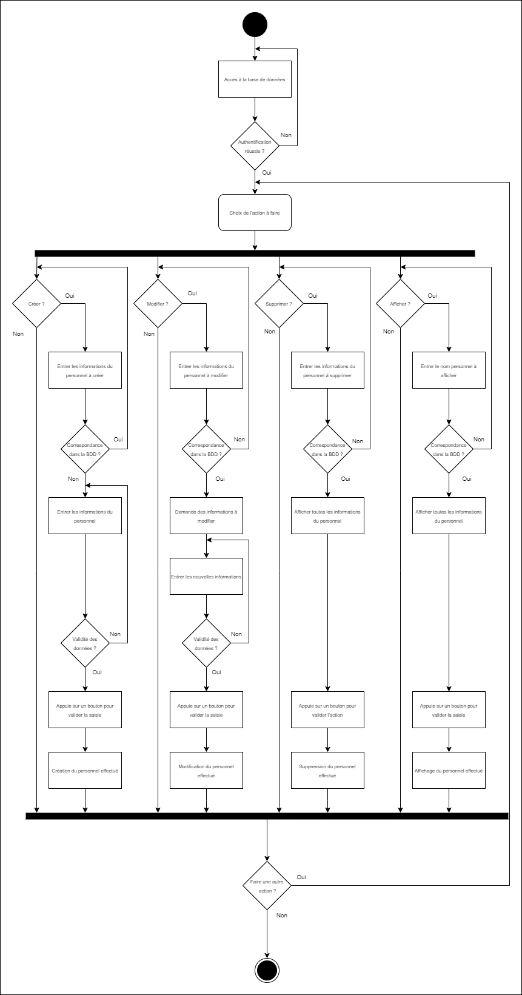
Ces différents diagrammes de séquence nous ont donc permis de bien comprendre le fonctionnement entre l’utilisateur l’interface graphique et enfin la base de données. Grâce à ça, une personne lambda peut comprendre assez facilement comment le système va fonctionner et comment celui-ci va interagir.

Nous continuons notre modélisation du système avec maintenant le diagramme d’activité.

Avant de le réaliser, nous allons le définir. Dans le langage UML, un diagramme d'activité fournit une vue du comportement d'un système en décrivant la séquence d'actions d'un processus. Les diagrammes d'activité sont similaires aux organigrammes de traitement de l'information, car ils montrent les flux entre les actions dans une activité. Les diagrammes d'activité peuvent, cependant, aussi montrer les flux parallèles simultanés et les flux de remplacement.

Dans les diagrammes d'activité, vous utilisez des nœuds d'activité et des bords d'activité pour modéliser le flux de commande et de données entre les actions.

Nous pouvons donc passer à notre modélisation (zoomer dans word pour mieux voir) :



Ou voici le lien internet : <https://drive.google.com/file/d/1BZ2wnIHAz-9fB4yGO7Vj0wB6QvtAGwjb/view?usp=sharing>

Il ne nous reste plus que le diagramme de classes à réaliser pour finir et c’est ce que nous allons faire, mais avant de commencer il serait bien de définir ce qu’est ce diagramme.

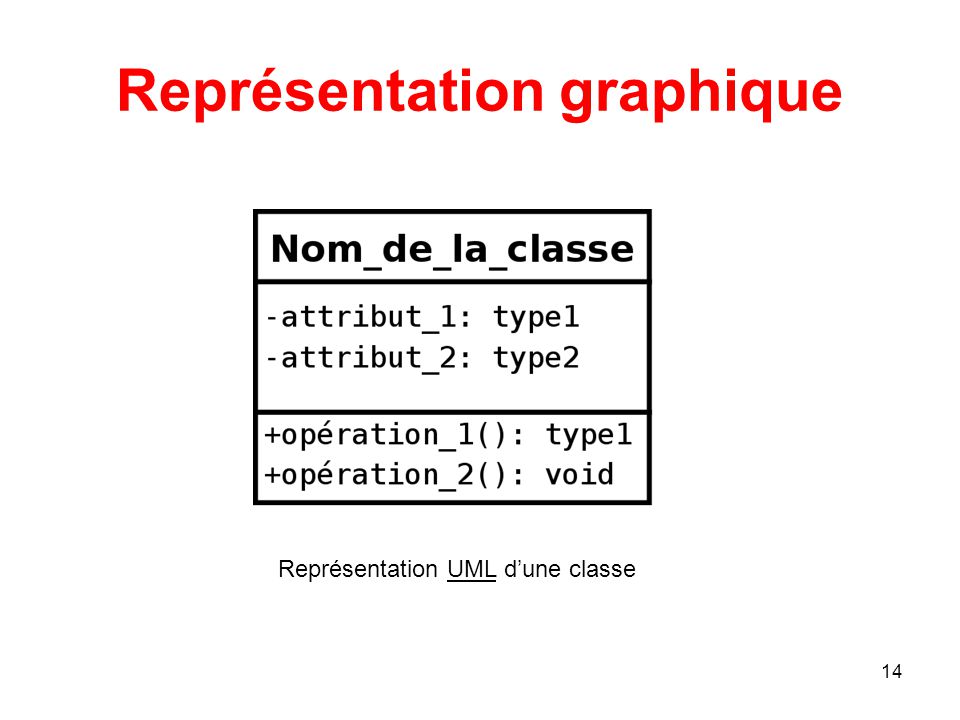
Dans le langage UML, les diagrammes de classes appartiennent à l'un des six types de diagramme structurel. Les diagrammes de classes sont fondamentaux pour le processus de modélisation des objets et modélisent la structure statique d'un système. Suivant la complexité d'un système, vous pouvez utiliser un seul diagramme de classes pour modéliser un système complet ou bien vous pouvez utiliser différents diagrammes de classes pour modéliser les composants d'un système.

Les diagrammes de classes sont les plans de votre système ou sous-système. Vous pouvez utiliser des diagrammes de classes pour modéliser les objets qui constituent le système, pour afficher les relations entre les objets et pour décrire ce que ces objets font et les services qu'ils fournissent.

Et voici donc le diagramme de classes que l’on a pu réaliser :

<https://mermaid.live/edit#pako:eNqtWd1u2zYUfhVBu4nbOK2TOk2N3QR2ixVogrTeejEIE1iJdrhKlCNKXrOufZ7lOfJiO6T4TyrdsOYm0vnj-fnOIUV_ToumxOkiLSrE2IqgbYvqjC4rgmmXpbMsnU6z9DhLz8sWM4ZZkiwSJJ8zKqnJdPooiUgkGb3CLWsoBfKPf02nyct6VzW393cgg-VjIDMsDhLF4IX0ZjpdNnWNaIk5Sz4CUz0pZ-HfK1R0fcvlNsOT9tSWSiJilrWjo0dZ-kjIisfztiNFBU4uEtY1xUe05WYlUQjRLOUBqOd1hzrCOnLTC6Wiod0Qj06CCHfd73BLcN_-RHCL2uKaK2R0fVNdIAqLtFLspnrb4_bW-KjpUs4ybIepaqSrktGMinInJvGfM5rA3w-kzHeSCAqEeyvotKnhfd21hG5_k7Rde38XoT_e4u61MXMwsQ09Zi4zWI9L7xtSGluXTS1suIuwgQHra06oejV4GFdXTBnGmBn0e9N3uD3wyKzf7VpSh4y6KcmGhHS02ZDiOqSbIhxMQhJkJCHlofItAUfNi_TcEGSBJ0ny5EmybCjr2r7oAFoJ2uMi2SFo7vu_u5a3Ll_pq724UFphrZPRLxwpCiu6vSVUpkjjSvWWQ88rsm8RAeujEkPXQZM8IAMN02LIdLgQL7AkgPMuixmWNKSVJx5Gzn13HzBmCQVRfnMBK9qHlrDFIon65jImYQ-tYkmFmY6voREAdgOSGjJg7DCxXt6o9DjUVyYeh_5Su6CW-GqtKgHpwFGjMWl77Hf5NOGbW75rWIcqewpNkz2pqoj8Dt2yYFYkkN0lGLoSdiB6M80S5rDE_LJTBprv-Uqg5NlkiqOnjqf3ro9rcfqYzhW4H1USjKiW7iCfoGYK5PWQR2zn0kwckUdrIsE6bp3UlqQ2mBJ1ALX6A-phGAZbCu3rHCbr_Z3YEJ0dCIXbmIL-yrYZxM98AccFM_atHcfxwik4C7iuy8MeFhlQ4WBiYt2Jv82E-0uwsQQ7iqqdSjYf5gFt2Er0djsUFQgynaaIPD8vZXoGKYhxrUM0rekuZxV9OLCp1hT5pjAFGKJF2HSCvWtxDbHlqLhGXSCiDpzO2dOpvjbPfYnX34i4DsUQsAp8GrfrioXRxOxbo_TcDYhZzHNVGTWOvydWhhrZSJGUACc2Li5V1lzy1RDwOY_XB5bZIgOOta9NFKiUXy6g5JFXnz0ovb_DeWE-CMxsB7ACUpuc0ALORzXmp253-osa2ceTCCBBEyIlY-wdIrgevlJcbt3cYjrOFsqsqWCUgnPbKi6FzKeG8bqG7wdEu_yay2-qBnUWntx88Ko6M8vne_nzp9ZlLIWB0bhUNP3-AiunAuO9ZUTcmo32rC7buE0j4hZ6fA7IYj4wApSEg42YwQsHHlGLnogLqFEnfUyNOxtIRjEZnVoSliG8FEMB16_3hcYu17XByxymwbiE-PcceVZ3BBT3bCmjUNMGVvoEM70pexK237RD_ae82yOPDKncQCPgYcuzhs_9HSNbqr563Ma_6SF60t3f5eKKwVFkuCcVrxACV_aEQ5aq8WGqIb56HXcjGfclvAC93IPJn2WMEVuapfIQar8zqQigY_OslAUDw8paHNq2gJ3jGJDfumkOnPL5Xll859ZjlQkMj0uOVtdZ7H-2gYVx-3JK4xxRfmoRE8dDc3FNNpsW52ANkeHKzOKypme5iCC6Y3XN8P1lq-wqUNljWvYssocSOsrcowoOoWLrwm1BUOU7IwXUSdJltUSAIh-EXLbVQVYiYu1js-2khdBfunmLGPMlvEyHJtc62SG8DMuUZGwWi6o8MI4HvlPD0JkrU8ZoW9p8q-TxrdFUfWRjtARsjMSsvQ9gEgk1IhTCKwz6vYWwcasD20ZjzJILyKg1T8QHsbFqt3fsUtlclqC2Rbf68q9K3sAowLl-z1_Sgp_R1ZcBvzBI5OzRHwZVjgepg7LpP0Abvl5NHAU9nP6VStzjKPVAfyz8RyU7QeZ6XWeF3VTLuozdIoHvBU95lA2mllpiuOiXL8aAOGdwVS4rXhQPTl_ovEQ7SO3AXxmCkuGkNebTrByedNJ0FObDjm-4zR_qukd9p02UESWGhIeWpLwYSvw0iV8dVJIeJ68pw21ndLzLpjWuIPZR9i87ftocZa9AO8q2XJK_2qiq0aZ2L4nd6qjPEXB6T7yjWNVsG1d3R7eW2UL9EOWaFBdhmmfMPXBDdek4GZ1unogb1sgtlR3Z6OeZ4nuZ8LeFN24yuDmdDW7O53vJE7mbuBEX-oohGq1km0zHolyaZAcR2jyrKKN3cF5dwh8JvtcxKz1MQbhGpEwXqQBqlnbXcJzL0gU8lqj9mKWDHOq7Zn1Li3SxQRXDh2kvOkT-EqupcMpIF5_TT-lienx6ejSbz2bPZ_OTs_nps9P5YXoL9JMXz46O52fzF7P52ezk6fHpl8P0z6YBGydHJ_OT2ezp8xdPj-fHp2dz0MAl6Zr2Qv74y_-JRX4VCl3b4y__APyWZ3Y>

Avant d’aborder directement les explications du diagramme de classe il faut comprendre les différentes parties qui le composent.

Voici une image qui vous permettra de mieux situer les prochaines explications : 

Premièrement, les classes :

Une classe est un moyen de modélisation de la structure et du comportement d’un objet. On y trouve le nom de la classe, les attributs qui sont des propriétés de la classe et les méthodes/opérations qui sont les actions que la classe peut effectuer. Dans ces méthodes on trouvera des méthodes/opérations nommées « constructeurs » et « destructeur » qui sont responsables de l'initialisation des objets lors de leur création et de la libération des ressources associées à ces objets lors de leur destruction. Les « constructeurs » sont des méthodes spéciales appelées automatiquement lorsqu'un objet est créé, permettant d'initialiser ses attributs et d'effectuer d'autres opérations nécessaires à son bon fonctionnement. En revanche, les « destructeurs » sont des méthodes spéciales appelées automatiquement lorsqu'un objet est détruit pour libérer des ressources telles que la mémoire allouée dynamiquement.

Vous pouvez remarquer sur la représentation graphique des ‘-‘ ou des ‘+’ qui représentent la visibilité des attributs et des méthodes. Un ‘-‘ correspond au privé (seul cette classe peut l’utiliser), un ‘+’ correspond au public (toutes les classes peuvent l’utiliser), et un ‘#’ correspond à du protected (seule cette classe et ses sous-classes peuvent l’utiliser).

Deuxièmement, les relations entre les classes :

Voici une image qui vous permettra de mieux visualiser les prochaines explications :



Dans notre diagramme nous avons utilisé seulement les relations d’associations, d’héritage, et de composition, nous allons donc expliquer seulement celles-ci.

-L’association représente une relation structurelle entre deux classes ce qui veut dire qu’une classe est liée à une autre par le biais d’une connexion. L’association peut être unidirectionnelle ou bidirectionnelle.

-La composition indique qu'une classe est composée de plusieurs instances d'une autre classe, mais les instances de la classe composée ne peuvent pas exister indépendamment de la classe composite.

-L’héritage représente une relation de généralisation/spécialisation entre deux classe, une classe appelée sous-classe hérite des attributs et des méthodes d’une classe appelée superclasse. La flèche est dirigée de la sous-classe vers la superclasse.

Dernièrement, les cardinalités : La cardinalité dans les diagrammes de classes indique le nombre d'instances d'une classe qui peuvent être liées à une instance de l'autre classe. Par exemple, chaque entreprise aura un ou plusieurs employés (et non zéro), et chaque employé travaille actuellement pour zéro ou une entreprise.

Les différentes cardinalités peuvent être :

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Dans notre diagramme nous utilisons seulement les cardinalités 1, n, O..n, 1..\*, et 1..n donc nous allons seulement expliquer celles-ci.

-Une cardinalité de 1 signifie qu'il y a exactement une instance de la classe associée. Par exemple, si une classe A est associée à une classe B avec une cardinalité de "1", cela signifie qu'une instance de la classe A est associée à exactement une instance de la classe B.

-Une cardinalité de n signifie qu'il y a exactement n instances de la classe associée. Par exemple, si une classe A est associée à une classe B avec une cardinalité de n, cela signifie qu'une instance de la classe A est associée à exactement n instances de la classe B.

-Une cardinalité 0..n indique qu'il peut y avoir zéro ou plusieurs instances de la classe associée. Par exemple, si une classe A est associée à une classe B avec une cardinalité de "0..n", cela signifie qu'une instance de la classe A peut être associée à aucune, une, ou plusieurs instances de la classe B.

-Une cardinalité 1..\* signifie qu'il doit y avoir au moins une instance de la classe associée, mais il peut y en avoir plusieurs. Par exemple, si une classe A est associée à une classe B avec une cardinalité de "1..\*", cela signifie qu'il doit y avoir au moins une instance de la classe B associée à une instance de la classe A, mais il peut y en avoir plus.

-Finalement, une cardinalité 1..n indique qu'il doit y avoir au moins une instance de la classe associée, et il peut y en avoir plusieurs. Cependant, contrairement à "1..\*", il spécifie un nombre maximal défini (n) d'instances associées. Par exemple, si une classe A est associée à une classe B avec une cardinalité de "1..n", cela signifie qu'il doit y avoir au moins une instance de la classe B associée à une instance de la classe A, mais il ne peut y en avoir que jusqu'à n instances.

Passons donc maintenant à l’explication de notre diagramme de classe.

Notre diagramme a des classes principales comme suit : Personnes, Adresses, Employé, Client, Commande, Facture, Articles, Statistiques.

Et des classes utilitaires comme suit : SuperieurHierarchique, SqlQuery, SqlManager, et Fonctions.

Parlons maintenant de leurs relations :

Voici le type de relation pour chacune des déclarations :

1. \*\*Client"1"--"2"Adresses : - Association

2. \*\*Adresse --\* Adresses : - Composition

3. \*\*Personnes <|-- Employé : - Héritage

4. \*\*Personnes <|-- Client : - Héritage

5. \*\*Client--Commande : - Association

6. \*\*Commande"1"--"1"Facture : Association

7. \*\*Adresse"1"--"1" Facture : Association

8. \*\*Commande"1..\*"--"1..\*"Articles : Association

9. \*\*Articles"1..n"<|--"1..n"Statistiques : Héritage

10. \*\*Employé <|-- SuperieurHierarchique - Héritage

11. \*\*SqlManager <|-- SqlQuery\*\* - Héritage

12. \*\*Commande <|-- SqlManager\*\* - Héritage

13. \*\*Employé "1"--"1"Adresse : - Association

Mais parlons de leurs cardinalités:

- Client"1"--"2"Adresses : adresses:

Chaque instance de la classe "Client" est associée exactement à 2 instances de la classe "Adresse", et 2 instances de la classe “Adresse” peuvent être liées à une instance de la classe “Client”

-Commande"1" -- "1" Facture :

Chaque instance de la classe "Commande" est associée exactement à une instance de la classe "Facture", et vice versa.

-Adresses"1" -- "1" Facture :

Chaque instance de la classe "Adresses" est associée exactement à une instance de la classe "Facture", et vice versa.

-Commande"1..\*" \*-- "1..\*" Articles :

Chaque instance de la classe "Commande" est associée à au moins une instance de la classe "Articles", et chaque instance de la classe "Articles" est associée à au moins une instance de la classe "Commande". Il y a une relation de composition entre Commande et Articles.

-Articles"1..n" <|-- "1..n" Statistiques :

Chaque instance de la classe "Articles" contient au moins une instance de la classe "Statistiques", et chaque instance de la classe "Statistiques" est contenue dans au moins une instance de la classe "Articles".

- Employé"1" -- "1"Adresse :

Chaque instance de la classe "Employé" est associée exactement à une instance de la classe "Adresse", et vice versa.

# **Environnement de travail**

On a pu réaliser un environnement de travail afin de pouvoir facilement gérer le partage de fichiers. Cet environnement de travail est un « Git » qui vient du site internet « GitHub ». GitHub est une plateforme open source de gestion de versions et de collaboration destinée aux développeurs de logiciels. Sur ce site web on peut créer des projets et ajouter des personnes à ces différents projets. Dans notre cas c’est Enorian, le chef du groupe qui a pu réaliser cet espace de travail et qui nous a tous ajoutés à celui-ci. GitHub permet d’actualiser les fichiers en temps réels et d’importer des modifications qui seront visibles à tous les autres membres du projet. Par exemple quand nous devrons réaliser le programme, si une personne de l’équipe réalise un changement dans le code et que celle-ci fait ce qu’on appelle un « push », alors les modifications seront enregistrées et nous aurons donc la version du programme après changements.

Voici donc le lien vers notre espace de travail :

<https://github.com/En0ri4n/Projet-POO>

# **Conclusion**

En conclusion, nous avons pu, dans ce livrable, réaliser toute la partie modélisation du système en entier. Nous avons donc réalisé la modélisation de la base de données avec le dictionnaire de données, le MCD et le MLD. Ensuite nous sommes passés à la modélisation du système en lui-même et pour ceci nous avons réalisé différents diagrammes UML qui sont :

* Diagramme de cas d’utilisation
* Diagramme d’activité
* Diagrammes de séquences
* Diagramme de classes

Ce livrable est essentiel à n’importe quel projet afin de partir sur une bonne base et de pouvoir réellement s’imaginer le projet en entier. Et nous avons aussi pu créer un environnement de travail. Celui-ci va nous permettre de pouvoir facilement transférer des fichiers avec toute l’équipe sans devoir tout le temps renvoyer des fichiers par mails ou différemment.