

# RAPPORT DE MINI PROJET



» TAHIRI Abderrahmane  
» HDIDOU Oussama



# TABLEAU DE CONTENU

- 01** Introduction
- 02** Acteurs et paramètres
- 03** Algorithme de sélection  
des chauffeurs et gestion  
de score
- 04** Algorithme de Tarification
- 05** Conception avec UML
- 06** Conception de base de  
donnée
- 07** Conclusion

# INTRODUCTION

La gestion efficace des services de livraison constitue un défi majeur pour de nombreuses entreprises opérant au Maroc. Dans le but de simplifier ce processus complexe, nous proposons de développer une application innovante qui mettra en relation les entreprises nécessitant des services de livraison avec des chauffeurs de voiture indépendants. Cette application vise à optimiser les opérations de livraison en établissant une connexion directe entre les demandeurs et les prestataires de services, offrant ainsi une alternative à l'approche traditionnelle consistant à faire appel à des sociétés de livraison établies telles que Amana, Aramex, et bien d'autres.

La problématique de notre approche consiste à comparer l'utilisation d'une application de gestion des services de livraison, reliant les entreprises aux chauffeurs de voiture indépendants, avec l'approche classique de faire appel à des sociétés de livraison établies telles que Amana, Aramex, et d'autres acteurs du marché.

L'approche classique fait appel à des sociétés de livraison établies qui offrent des services de livraison de produits pour le compte des entreprises. Ces sociétés ont généralement une infrastructure bien établie, avec des flottes de véhicules et des chauffeurs à leur disposition. Elles assurent la responsabilité de la gestion de la livraison, y compris la planification des itinéraires, la gestion des chauffeurs, la sécurité des produits, etc. Les entreprises externalisent ainsi complètement la gestion de leurs livraisons à ces sociétés spécialisées.

En revanche, notre approche propose de mettre en relation directe les entreprises avec des chauffeurs de voiture indépendants via une application de gestion des services de livraison. Les chauffeurs indépendants sont des professionnels ou des particuliers qui offrent leurs services de livraison de manière flexible. Les entreprises peuvent sélectionner les chauffeurs en fonction de leurs besoins spécifiques, tels que les villes de destination, les contraintes horaires, les capacités de livraison, etc. Les chauffeurs indépendants sont responsables de l'exécution des livraisons, tandis que l'application facilite la gestion, la coordination et le suivi de ces livraisons.

La problématique se pose donc sur les avantages et les inconvénients de chaque approche. D'un côté, l'approche classique avec des sociétés de livraison établies offre une infrastructure solide et des services clés en main, assurant la sécurité et la fiabilité des livraisons. De l'autre côté, notre approche avec des chauffeurs indépendants via une application permet une plus grande flexibilité, une personnalisation des livraisons et une réduction potentielle des coûts.

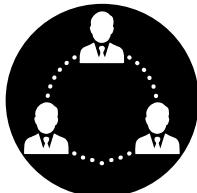
Il est donc essentiel de comparer ces deux approches en termes de coûts, de qualité de service, de flexibilité, de rapidité des livraisons, de contrôle sur le processus de livraison, de sécurité des produits, etc. Il convient également d'évaluer les risques liés à l'utilisation de chauffeurs indépendants, tels que la sécurité des produits, la confiance dans les chauffeurs, la gestion des litiges, etc.

La problématique se focalise ainsi sur la recherche d'une solution optimale pour la gestion des services de livraison au Maroc, en tenant compte des avantages et des inconvénients de l'approche classique et de notre approche innovante basée sur une application de gestion des chauffeurs indépendants.



# ACTEURS ET PARAMÈTRES

L'application que nous développons implique plusieurs acteurs clés, chacun jouant un rôle spécifique dans le processus de gestion des services de livraison au Maroc. Les acteurs principaux sont les entreprises, les chauffeurs de voiture indépendants et les administrateurs du système.



## No. 01 – les entreprises

Les entreprises sont les utilisateurs qui ont besoin de faire livrer leurs produits à travers le pays. Elles fournissent les informations sur les produits à livrer, y compris la liste des produits, les quantités, les destinations, les contraintes horaires et éventuellement d'autres détails spécifiques à la livraison. Les entreprises peuvent varier en taille et en secteur d'activité, allant des petites entreprises locales aux grandes sociétés internationales.



## No. 02 – Les chauffeurs

Les chauffeurs de voiture indépendants sont les prestataires de services de livraison qui s'inscrivent sur la plateforme de l'application en tant que conducteurs disponibles pour effectuer des livraisons. Ils sont responsables de la prise en charge des produits auprès du point de ramassage dans une ville et de leur livraison à un autre point dans une autre ville ou à plusieurs points dans différentes villes. Les paramètres pris en compte pour les chauffeurs comprennent leur disponibilité, les villes de destination auxquelles ils sont prêts à livrer, le poids maximal et le volume maximal qu'ils peuvent transporter, leur score de confiance basé sur les performances passées, ainsi que leur coût de livraison par kilomètre parcouru.



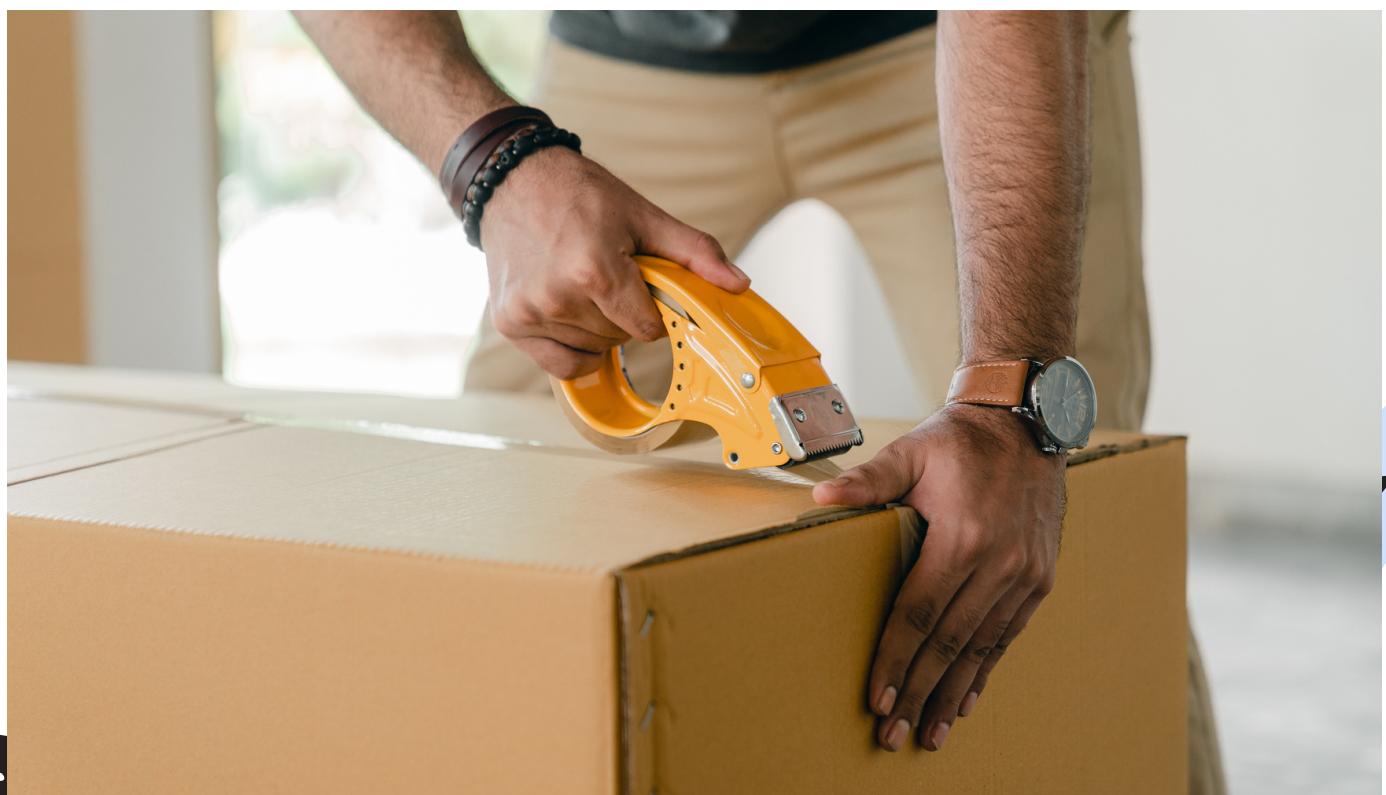
## No. 03 – Les administrateurs

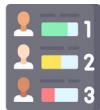
Les administrateurs du système sont responsables de la gestion globale de l'application. Ils assurent le bon fonctionnement de la plateforme, vérifient les inscriptions des chauffeurs, gèrent les paramètres et les algorithmes utilisés pour sélectionner les chauffeurs, ainsi que pour mettre à jour les scores de confiance en fonction des événements survenus. Les administrateurs jouent un rôle essentiel dans la supervision et la coordination de l'ensemble du processus de livraison.

Du côté des paramètres, pour les chauffeurs, la disponibilité est un élément crucial. Ils doivent spécifier les jours et les heures auxquels ils sont disponibles pour effectuer des livraisons. Les villes de destination sont également importantes, car certains chauffeurs peuvent avoir des préférences ou des restrictions quant aux endroits où ils sont prêts à se rendre. Le poids maximal et le volume maximal qu'un chauffeur peut transporter déterminent les types de produits qu'il est en mesure de livrer. Le score de confiance, calculé à partir de différentes métriques telles que l'ancienneté, le taux de réussite des livraisons et d'autres facteurs, permet d'évaluer la fiabilité et la qualité du service offert par chaque chauffeur. Enfin, le coût de livraison par kilomètre parcouru peut varier en fonction des politiques tarifaires de chaque chauffeur.

Pour les entreprises, les paramètres spécifiques concernent principalement les produits à livrer, comprenant la liste des produits, les quantités et les éventuelles spécifications particulières. Les contraintes horaires sont également un facteur important, car les entreprises peuvent avoir des délais stricts à respecter pour la livraison de leurs produits.

En prenant en compte ces paramètres pour les chauffeurs et les entreprises, notre application sera en mesure de faciliter la mise en relation efficace entre les demandeurs et les prestataires de services de livraison, en optimisant la sélection des chauffeurs les plus adaptés pour chaque ensemble de produits à livrer, tout en respectant les contraintes et les préférences des entreprises.





# Score de confiance du chauffeur

Le score de confiance du chauffeur joue un rôle essentiel dans le processus de sélection des chauffeurs pour les livraisons. Il permet d'évaluer la fiabilité, la compétence et la qualité du service offert par chaque chauffeur, offrant ainsi une mesure objective pour guider la décision de sélection.

Pour calculer le score de confiance, plusieurs paramètres sont pris en compte. L'ancienneté du chauffeur est un critère important, car une expérience plus longue dans le domaine de la livraison est généralement associée à une meilleure connaissance des procédures, des itinéraires et des bonnes pratiques. De plus, l'ancienneté permet de recueillir des données sur les performances passées du chauffeur, ce qui est essentiel pour évaluer sa fiabilité.

Le volume, le poids et la valeur des produits livrés avec succès sont également des facteurs déterminants dans le calcul du score de confiance. Un chauffeur qui a démontré sa capacité à gérer des livraisons de gros volumes, de poids élevé et de valeur importante inspire davantage de confiance. Ces paramètres reflètent la capacité du chauffeur à prendre en charge des missions complexes et à respecter les exigences spécifiques des entreprises.

Les retards et les pénalités sont des éléments négatifs pris en compte dans le calcul du score de confiance. Un chauffeur qui a accumulé des retards répétés ou qui a été pénalisé pour des erreurs ou des incidents lors des livraisons verra son score de confiance diminuer. Cela permet d'inciter les chauffeurs à respecter les délais et à fournir un service de qualité.

La méthode d'initialisation du score de confiance peut varier selon les besoins et les critères spécifiques de l'application. Une approche courante consiste à attribuer un score de base à chaque chauffeur lors de son inscription sur la plateforme. Ce score de base peut être calculé en prenant en compte des informations générales sur le chauffeur, telles que son expérience antérieure, ses qualifications et ses références. Il peut également être basé sur des scores de confiance provenant de sources externes ou de partenaires de confiance.

La mise à jour du score de confiance se fait en fonction des événements survenus au cours des livraisons. Lorsqu'un chauffeur effectue une livraison réussie dans les délais impartis, son score de confiance est augmenté. En revanche, en cas de retards, de produits perdus ou d'autres incidents, le score de confiance est réduit. La méthode de mise à jour peut être basée sur des mécanismes de notation, où les entreprises et les clients peuvent évaluer la performance du chauffeur après chaque livraison. Ces évaluations sont prises en compte pour ajuster le score de confiance du chauffeur en conséquence.

Il est essentiel de mettre en place des mécanismes de mise à jour réguliers pour maintenir la pertinence et la précision du score de confiance. Cela permet de refléter de manière dynamique les performances et le comportement réels de chaque chauffeur, garantissant ainsi une sélection plus efficace et fiable lors du processus de livraison.

En résumé, le score de confiance du chauffeur est un outil essentiel pour évaluer la fiabilité et la qualité du service offert par chaque chauffeur. En prenant en compte des paramètres tels que l'ancienneté, le volume, le poids, la valeur des livraisons réussies, les retards et les pénalités, le score de confiance permet d'identifier les chauffeurs les plus compétents et les plus fiables pour les missions de livraison. Sa méthode d'initialisation et de mise à jour permet de maintenir une évaluation précise et en temps réel de la performance de chaque chauffeur.

## LES ÉTAPES D'UN ALGORITHME QUI POURRAIT GÉRER LES SCORES DES CHAUFFEURS

1

### No. 01 — Initialisation des scores de confiance

- Pour chaque chauffeur inscrit sur la plateforme, attribuer un score de confiance initial basé sur des critères tels que l'expérience antérieure, les qualifications et les références.
- Définir une valeur de score de confiance de départ pour tous les chauffeurs.

2

### No. 02 — Réception des données de livraison

- Recevoir les données de livraison de chaque livraison effectuée par les chauffeurs, y compris les informations sur la réussite de la livraison, les retards éventuels, les pénalités et les commentaires des clients ou des entreprises.

3

### No. 03 — Mise à jour des scores de confiance

- Pour chaque livraison effectuée par un chauffeur, mettre à jour son score de confiance en fonction des résultats de la livraison.
- Si la livraison est réussie dans les délais impartis, augmenter le score de confiance du chauffeur.
- En cas de retard, de produits perdus ou d'autres incidents, réduire le score de confiance du chauffeur.

4

### No. 04 — Gestion des pénalités

- Si un chauffeur accumule un certain nombre de pénalités ou d'incidents négatifs au fil du temps, réduire son score de confiance de manière plus significative.
- Définir des seuils ou des critères spécifiques pour déterminer le nombre de pénalités ou d'incidents qui entraîneront une réduction plus importante du score de confiance.

**5**

## No. 05 – Réévaluation régulière des scores de confiance

- Recevoir les données de livraison de chaque livraison effectuée par les chauffeurs, y compris les informations sur la réussite de la livraison, les retards éventuels, les pénalités et les commentaires des clients ou des entreprises.

**6**

## No. 06 – Utilisation des scores de confiance pour la sélection des chauffeurs

- Lorsqu'une entreprise demande une livraison, utiliser les scores de confiance des chauffeurs pour sélectionner les chauffeurs les plus appropriés.
- Définir des critères de sélection basés sur les scores de confiance, tels que la priorité donnée aux chauffeurs ayant les scores les plus élevés ou la possibilité de filtrer les chauffeurs en fonction de leur score minimum requis.

**7**

## No. 07 – Communication des scores de confiance

- Fournir aux entreprises des informations sur les scores de confiance des chauffeurs sélectionnés pour leurs livraisons.
- Permettre aux entreprises de prendre en compte les scores de confiance lorsqu'elles évaluent la fiabilité des chauffeurs et prennent des décisions concernant leurs services de livraison.

**8**

## No. 08 – Prise en compte des ressources et des types de voitures

- Prendre en compte le coefficient de ressource de chaque chauffeur, qui représente le nombre de produits qu'il peut livrer avec son véhicule.
- Un chauffeur capable de livrer un grand nombre de produits avec son véhicule aura un coefficient de ressource plus élevé, ce qui augmentera son score de confiance.
- De même, prendre en compte les types de voitures de chaque chauffeur. Une voiture de plus grande capacité augmentera le score de confiance du chauffeur, car il sera en mesure de gérer des livraisons plus importantes.

**9**

## No. 09 – Fonction de parrainage

- Mettre en place une fonctionnalité de parrainage où les chauffeurs peuvent parrainer de nouveaux chauffeurs pour rejoindre la plateforme.
- Si un chauffeur réussit à parrainer un bon chauffeur, son score de confiance sera augmenté en récompense pour avoir contribué à l'expansion de la base de chauffeurs fiables sur la plateforme.

# DESCRIPTION DÉTAILLÉE DE L'ALGORITHME DE SÉLECTION DES CHAUFFEURS

L'algorithme de sélection des chauffeurs vise à choisir le chauffeur le plus approprié pour effectuer la livraison d'un produit donné.

1

## No. 01 – Entrées

- Liste des chauffeurs disponibles (drivers)
- Produit à livrer (product)

2

## No. 02 – Variables

- meilleurChauffeur : variable pour stocker le chauffeur le plus approprié
- différenceMinimale : variable pour stocker la plus petite différence de score
- 

3

## No. 03 – Pour chaque chauffeur dans la liste des chauffeurs disponibles :

- Calculer la différence de score entre le chauffeur et le produit en appelant la fonction calculerDifférenceScore(chauffeur, product)
- Si la différence de score est inférieure à la différence minimale :
- Mettre à jour la différence minimale avec la différence de score
- Assigner le chauffeur comme le meilleur chauffeur

4

## No. 04 – Retourner le meilleur chauffeur

# DESCRIPTION DÉTAILLÉE DE L'ALGORITHME DE SÉLECTION DES CHAUFFEURS

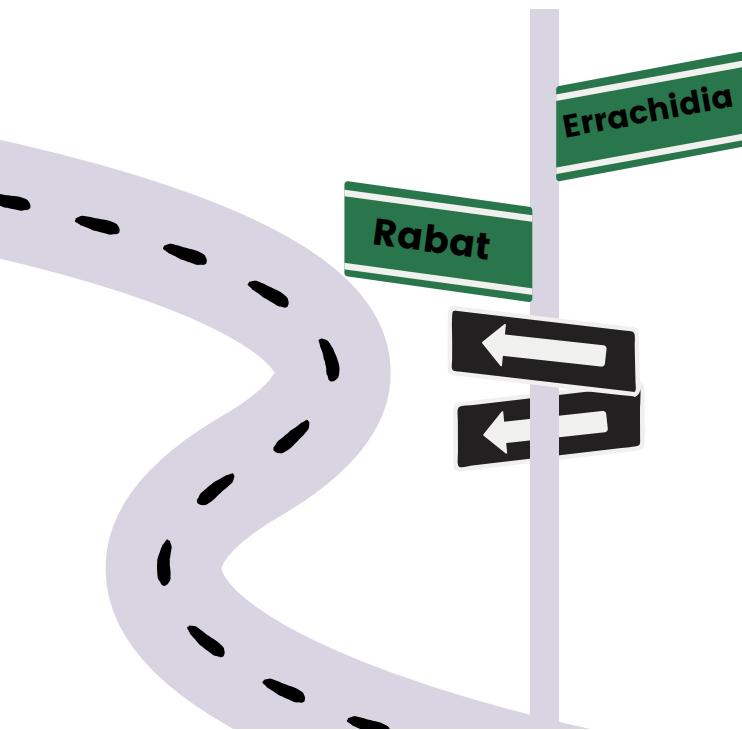


## PROBLEMATIQUE DE CHOIX DE CHAUFFEURS

Dans notre entreprise, nous sommes confrontés à un problème de monopolisation des livraisons par des chauffeurs expérimentés, ce qui limite les opportunités de développement pour les livreurs débutants. Afin de remédier à cette situation, nous avons mis en place une méthode basée sur l'attribution de scores aux chauffeurs. Ces scores sont déterminés en fonction de critères tels que l'ancienneté de travail, le modèle de voiture utilisé et le kilométrage parcouru. Par la suite, ces scores sont modifiés en fonction des performances des chauffeurs. Ainsi, si un chauffeur accomplit son travail sans retard ni dommage aux produits livrés, son score augmente, sinon il diminue en fonction de certains coefficients.

De même, chaque produit possède également un score, déterminé par des facteurs tels que le prix, la distance de livraison, le poids et la catégorie du produit. Pour choisir le chauffeur chargé de livrer un produit donné, nous calculons la différence entre le score de chaque chauffeur disponible et le score du produit à livrer. Ensuite, nous sélectionnons le chauffeur ayant la plus petite différence positive entre son score et celui du produit.

Cette méthode permet ainsi de mieux répartir les livraisons entre les chauffeurs, en offrant aux débutants des opportunités de développement et en récompensant les performances des chauffeurs expérimentés. De plus, en choisissant le chauffeur le plus adapté à chaque produit, nous nous assurons d'une livraison efficace et optimisée.



# DESCRIPTION DÉTAILLÉE DE L'ALGORITHME DE SÉLECTION DES CHAUFFEURS

## CODE DE CHOIX DE CHAUFFEURS

```
Algorithme choix de chauffeur
Fonction choisir_chauffeur(chauffeurs, nb_chauffeurs, produit)
    meilleur_chauffeur : Chauffeur ;
    difference_score_min : réel ;
    difference_score_min <- 1000.0 ; ' Valeur initiale élevée

    Pour i allant de 0 à nb_chauffeurs - 1 faire
        difference_score : réel ;
        difference_score <- valeur_absolue(chauffeurs[i].score - produit.score) ;

        Si difference_score < difference_score_min alors ;
            difference_score_min <- difference_score ;
            meilleur_chauffeur <- chauffeurs[i] ;
        Fin Si
    Fin Pour

    Retourner meilleur_chauffeur ;
Fin

Debut algorithme
nb_chauffeurs : entier ;
Afficher("Entrez le nombre de chauffeurs disponibles : ") ;
Lire(nb_chauffeurs) ;

chauffeurs : tableau de Chauffeur de taille nb_chauffeurs ;
Pour i allant de 0 à nb_chauffeurs - 1 faire
    Afficher("\n++++++Entrez les détails du Chauffeur ", i + 1, " :+++++")
    Afficher("Nom : ") ;
    Lire(chauffeurs[i].nom) ;
    Afficher("Annees d'expérience : ") ;
    Lire(chauffeurs[i].annees_experience) ;
    Afficher("Modele de voiture : ") ;
    Lire(chauffeurs[i].modele_voiture) ;
    Afficher("Kilometrage : ") ;
    Lire(chauffeurs[i].kilometrage) ;

    ' Menu numérique pour la catégorie de véhicule
    Afficher("Categorie de véhicule :") ;
    Afficher("1. Petite") ;
    Afficher("2. Moyenne") ;
    Afficher("3. Grande") ;
    Afficher("4. Très grande") ;
    choix_categorie : entier ;
    Afficher("Entrez le numéro de la catégorie : ") ;
    Lire(choix_categorie) ;

    coefficient_categorie : réel ;
    Selon choix_categorie faire ;
        Cas 1 :
            coefficient_categorie <- 0.8 ;
        Cas 2 :
            coefficient_categorie <- 1.0 ;
```

# DESCRIPTION DÉTAILLÉE DE L'ALGORITHME DE SÉLECTION DES CHAUFFEURS

## CODE DE CHOIX DE CHAUFFEURS

```
    coefficient_categorie <- 1.0 ;
Cas 3 :
    coefficient_categorie <- 1.2 ;
Cas 4 :
    coefficient_categorie <- 1.5 ;
Default :
    coefficient_categorie <- 1.0 ;' Par défaut, coefficient = 1.0
Fin Selon

' Calculer le score en utilisant le coefficient de catégorie
chauffeurs[i].score <- chauffeurs[i].annees_experience * 0.5 + chauffeurs[i].kilometrage * 0.1 * coefficient_categorie ;
Fin Pour

nb_produits : entier ;
Afficher("\nEntrez le nombre de produits à livrer : ") ;
Lire(nb_produits) ;

produits : tableau de Produit de taille nb_produits ;
Pour i allant de 0 à nb_produits - 1 faire
    Afficher("\n++++++Entrez les détails du Produit ", i + 1, " :++++++") ;
    Afficher("Nom : ") ;
    Lire(produits[i].nom) ;
    Afficher("Prix : ") ;
    Lire(produits[i].prix) ;
    Afficher("Distance : ") ;
    Lire(produits[i].distance) ;
    Afficher("Poids : ") ;
    Lire(produits[i].poids) ;

    ' Menu numérique pour la catégorie de produit
    Afficher("Categorie de produit :") ;
    Afficher("1. Electronique") ;
    Afficher("2. Toxique") ;
    Afficher("3. Vetement") ;
    Afficher("4. Matiere alimentaire") ;
    choix_categorie : entier ;
    Afficher("Entrez le numéro de la catégorie : ") ;
    Lire(choix_categorie) ;

    coefficient_categorie : réel ;
Selon choix_categorie faire ;
    Cas 1 :
        coefficient_categorie <- 1.2 ;
    Cas 2 :
        coefficient_categorie <- 1.5 ;
    Cas 3 :
        coefficient_categorie <- 0.8 ;
    Cas 4 :
        coefficient_categorie <- 1.0 ;
    Default :
        coefficient_categorie <- 1.0 ; // Par défaut, coefficient = 1.0
Fin Selon
```

# DESCRIPTION DÉTAILLÉE DE L'ALGORITHME DE SÉLECTION DES CHAUFFEURS



## CODE DE CHOIX DE CHAUFFEURS

```
' Calculer le score en utilisant le coefficient de catégorie
produits[i].score <- produits[i].prix * 0.01
|           + produits[i].distance * 0.05
|           + produits[i].poids * 0.2 * coefficient_categorie ;
Fin Pour

Afficher("\nAttribution des produits aux chauffeurs...\n\n") ;

index_chauffeur : entier ; // Indice du chauffeur actuel
index_chauffeur <- 0 ;
Pour i allant de 0 à nb_produits - 1 faire ;
    chauffeur_attribue : Chauffeur ;
    chauffeur_attribue <- choisir_chauffeur(&chauffeurs[index_chauffeur], 1, produits[i]) ;
    Afficher("+++++++"+Produit ", i + 1,
    " attribué au Chauffeur ", chauffeur_attribue.nom,
    " (Modèle de voiture : ", chauffeur_attribue.modele_voiture,
    ") qui a le score : ", chauffeur_attribue.score, "+++++++"") ;

    index_chauffeur <- (index_chauffeur + 1) modulo nb_chauffeurs ;
Fin Pour

Libérer mémoire pour chauffeurs ;
Libérer mémoire pour produits ;
Fin Algorithme principal ;
```

# DESCRIPTION DÉTAILLÉE DE L'ALGORITHME DE SÉLECTION DES CHAUFFEURS

## CODE DE CHOIX DE CHAUFFEURS

Le code que j'ai fourni est un algorithme basé sur un pseudo-code pour gérer le score de confiance des chauffeurs dans un système de livraison. Ce score de confiance est utilisé pour évaluer la fiabilité et la qualité du service offert par chaque chauffeur, afin de guider la décision de sélection lors du processus de livraison.

L'algorithme commence par déclarer les variables nécessaires à son fonctionnement, telles que les coefficients de sélection des chauffeurs, l'expérience du chauffeur, le nombre de voyages effectués hebdomadairement, etc. Ces variables seront utilisées pour calculer le score de chaque chauffeur et le score de chaque produit.

Ensuite, l'algorithme définit plusieurs fonctions pour effectuer différentes opérations. Par exemple, il y a une fonction pour attribuer le score de base à un chauffeur lors de son inscription, en utilisant les coefficients de sélection et les informations spécifiques du chauffeur. Il y a aussi des fonctions pour réduire le score d'un chauffeur en cas d'incident ou de perte de produit, ainsi que pour augmenter le score en cas de livraison réussie.

L'algorithme comprend également une fonction pour calculer le score d'un chauffeur en fonction de critères tels que l'ancienneté de travail, le modèle de voiture utilisé, le kilométrage parcouru, etc. De même, il y a une fonction pour calculer le score d'un produit en fonction de critères tels que le prix, la distance de livraison, le poids, la catégorie, etc.

Enfin, l'algorithme principal est présenté. Il effectue un test en utilisant les fonctions précédemment définies pour choisir le chauffeur approprié pour une livraison donnée, en comparant les scores du chauffeur et du produit.





# PROBLEMATIQUE DE TARIFICATION



Le code généré présente un algorithme de tarification de livraison basé sur plusieurs critères tels que la distance, le poids, la catégorie du produit et le prix du produit. Il permet de calculer le prix de livraison total, le montant de paiement du chauffeur, le bénéfice généré et le montant ajouté à la caisse pour récompenser le chauffeur et gérer les pertes éventuelles.

Le programme commence par demander à l'utilisateur d'entrer le nombre de produits à livrer. Ensuite, il initialise un tableau de produits de taille nombreProduits.

En utilisant une boucle, le programme demande à l'utilisateur de fournir les détails de chaque produit, tels que la distance de livraison, le poids, la catégorie (électronique, toxique, vêtement, matière alimentaire) et le prix du produit.

Une fois que tous les produits sont saisis, l'algorithme appelle la fonction "calculerPrixTarification" en lui passant le nombre de produits et le tableau de produits. Cette fonction effectue les calculs nécessaires pour déterminer le prix total de la livraison en ajoutant les frais de base, les frais de kilomètre, les frais de poids, les frais supplémentaires pour les produits toxiques et électroniques, ainsi que la moitié du prix de chaque produit.

Ensuite, les fonctions "calculerPrixPaiement", "calculerPrixCaisse" et "calculerPrixRecompense\_Perte" sont appelées pour calculer respectivement le montant de paiement du chauffeur, le bénéfice généré et le montant ajouté à la caisse pour récompenser le chauffeur et gérer les pertes.

Enfin, le programme affiche la facture complète avec le prix de tarification, le prix de paiement du chauffeur, le montant de bénéfice et le montant ajouté à la caisse.

Le problème résolu par cet algorithme est de calculer le coût de livraison en fonction de différents paramètres tels que la distance, le poids, la catégorie du produit et le prix du produit. Il fournit également des informations sur le paiement du chauffeur, le bénéfice généré et le montant ajouté à la caisse pour récompenser le chauffeur et gérer les pertes.





# CODE DE TARIFICATION

```
Algorithme de tarification de livraison

CONSTANTE FRAIS_BASE <- 10;
CONSTANTE FRAIS_KILOMETRE <- 1;
CONSTANTE FRAIS_POIDS <- 0.5;
CONSTANTE FRAIS_TOXIQUE <- 5;
CONSTANTE FRAIS_ELECTRONIQUE <- 3;
CONSTANTE PART_CHAUFFEUR <- 1/3;
CONSTANTE PART_CAISSER <- 1/3;
CONSTANTE PART_RP <- 1/3;

Structure Produit
    distance : réel;
    poids : réel;
    categorie : entier;
    prixProduit : réel;
Fin Structure

Fonction calculerPrixTarification(nombreProduits : entier, produits[] : tableau de Produit) : réel
    prix <- FRAIS_BASE;

    Pour i allant de 0 à nombreProduits - 1 faire
        prix <- prix + produits[i].distance * FRAIS_KILOMETRE;
        prix <- prix + produits[i].poids * FRAIS_POIDS;

        Si produits[i].categorie = 2 alors
            |   prix <- prix + FRAIS_TOXIQUE;
        Fin Si

        Si produits[i].categorie = 1 alors
            |   prix <- prix + FRAIS_ELECTRONIQUE;
        Fin Si

        prix <- prix + produits[i].prixProduit / 2;
    Fin Pour

    Retourner prix
Fin Fonction

Fonction calculerPrixPaiement(prixTarification : réel) : réel
    Retourner prixTarification * PART_CHAUFFEUR;
Fin Fonction

Fonction calculerPrixCaisse(prixTarification : réel) : réel
    Retourner prixTarification * PART_CAISSER;
Fin Fonction

Fonction calculerPrixRecompense_Perte(prixTarification : réel) : réel
    Retourner prixTarification * PART_RP;
Fin Fonction
```



# CODE DE TARIFICATION

```
Algorithme principal
    nombreProduits : entier

    Afficher("++++++tarification de livraison+++++");
    Afficher("Entrez le nombre de produits à livrer : ");
    Lire(nombreProduits);

    produits : tableau de Produit de taille nombreProduits;

    Pour i allant de 0 à nombreProduits - 1 faire
        Afficher("\n-->Produit ", i+1);
        Afficher("Entrez la distance de la livraison (en kilomètres) : ");
        Lire(produits[i].distance);
        Afficher("Entrez le poids de la commande (en kilogrammes) : ");
        Lire(produits[i].poids);
        Afficher("Selectionnez la catégorie du produit :")
        Afficher("1. Electronique");
        Afficher("2. Toxique");
        Afficher("3. Vêtement");
        Afficher("4. Matière alimentaire");
        Afficher("Selectionnez une option : ");
        Lire(produits[i].categorie);
        Afficher("Entrez le prix du produit : ");
        Lire(produits[i].prixProduit);
    Fin Pour

    prixTarification <- calculerPrixTarification(nombreProduits, produits);
    prixPaiement <- calculerPrixPaiement(prixTarification);
    benifice <- calculerPrixCaisse(prixTarification);
    prixRP <- calculerPrixRecompense_Perte(prixTarification);

    Afficher( "*****FACTURE*****");
    Afficher( "--> Prix de tarification : ", prixTarification, " MAD");
    Afficher( "--> Prix de paiement du chauffeur : ", prixPaiement, " MAD");
    Afficher( "--> Le Montant de bénéfice : ", benifice, " MAD");
    Afficher( "--> Montant ajouté au caisse pour récompenser notre chauffeur fidèle et gérer les pertes : ", prixRP, " MAD");
Fin Algorithme
```



# PROBLEMATIQUE DE GESTION DES SCORES

Problématique résolue par le code : Algorithme de gestion de scores des chauffeurs.

Description de la problématique :

Une entreprise de livraison doit gérer un groupe de chauffeurs et suivre leurs performances. Il est essentiel de pouvoir ajouter de nouveaux chauffeurs, consulter leur score (évaluant leur performance) et modifier leur score en fonction de différents critères, tels que la réussite ou le retard des livraisons, ainsi que les pertes de produits.

Solution apportée par le code :

L'algorithme présenté permet de résoudre cette problématique en proposant différentes fonctionnalités. Il utilise une structure "Chauffeur" pour stocker les informations relatives à chaque chauffeur, telles que le nom, le nombre d'années d'expérience, le kilométrage et le score.

1. Fonction "ajouterChauffeur":

Cette fonction permet d'ajouter un nouveau chauffeur en demandant à l'utilisateur de saisir son nom, le nombre d'années d'expérience et le kilométrage de sa voiture. Le score initial du chauffeur est calculé en fonction de ces données, en attribuant des poids spécifiques à l'expérience et au kilométrage.

2. Fonction "consulterScoreChauffeur":

Cette fonction permet à l'utilisateur de consulter le score d'un chauffeur en saisissant son nom. Elle recherche le chauffeur correspondant dans le tableau et affiche son score.

3. Fonction "modifierScoreChauffeur":

Cette fonction permet de modifier le score d'un chauffeur en fonction de différentes options. L'utilisateur doit saisir le nom du chauffeur et choisir parmi les options suivantes :

- Livraison réussie : L'utilisateur doit fournir la valeur de la livraison, le temps estimé et le temps réel. Le score est mis à jour en fonction de la valeur de la livraison, ainsi que de la différence entre le temps estimé et le temps réel.

- Livraison en retard : L'utilisateur doit fournir la valeur de la livraison, le temps estimé et le temps réel. Le score est mis à jour en fonction de la valeur de la livraison, ainsi que de la différence entre le temps réel et le temps estimé (avec une pénalité appliquée).

- Perte de produit : L'utilisateur doit fournir la valeur de la livraison perdue. Le score est réduit en fonction de la valeur de la livraison (avec une pénalité appliquée).

Ensuite, le programme principal présente un menu à l'utilisateur, lui permettant de choisir parmi les différentes fonctionnalités. Il boucle jusqu'à ce que l'utilisateur choisisse de quitter (option 0).

Ainsi, cet algorithme permet de gérer efficacement les chauffeurs, d'ajouter de nouveaux chauffeurs, de consulter et de modifier leurs scores en fonction des performances de livraison.



# PROBLEMATIQUE DE GESTION DES SCORES

```
Algorithme de gestion des chauffeurs ;  
  
CONSTANTE MAX_CHAUFFEURS <- 100 ;  
  
Structure Chauffeur  
    nom : tableau de caractères de taille 50 ;  
    annees_experience : entier ;  
    mileage : réel ;  
    score : réel ;  
Fin Structure  
  
chauffeurs : tableau de Chauffeur de taille MAX_CHAUFFEURS ;  
nbChauffeurs : entier ;  
  
fonction ajouterChauffeur()  
Debut  
    Chauffeur nouveauChauffeur ;  
    Afficher("Entrez le nom du chauffeur : ") ;  
    Lire(nouveauChauffeur.nom) ;  
    Afficher("Entrez le nombre d'années d'expérience : ") ;  
    Lire(nouveauChauffeur.annees_experience) ;  
    Afficher("Entrez le kilométrage de la voiture : ") ;  
    Lire(nouveauChauffeur.mileage) ;  
  
    // Calcul du score initial en fonction des données d'entrée  
    nouveauChauffeur.score <- nouveauChauffeur.annees_experience * 0.5 + nouveauChauffeur.mileage * 0.1 ;  
  
    chauffeurs[nbChauffeurs] <- nouveauChauffeur ;  
    nbChauffeurs <- nbChauffeurs + 1 ;  
  
    Afficher("Le chauffeur a été ajouté avec succès.") ;  
    Afficher("Score initial du chauffeur ", nouveauChauffeur.nom, " : ", nouveauChauffeur.score) ;  
Fin  
  
fonction consulterScoreChauffeur()  
Debut  
    nomChauffeur : tableau de caractères de taille 50 ;  
    Afficher("Entrez le nom du chauffeur : ") ;  
    Lire(nomChauffeur) ;  
  
    i : entier ;  
    Pour i allant de 0 à nbChauffeurs - 1 faire ;  
        Si strcmp(chauffeurs[i].nom, nomChauffeur) = 0 alors ;  
            Afficher("Score du chauffeur ", chauffeurs[i].nom, " : ", chauffeurs[i].score) ;  
            Retourner ;  
        Fin Si ;  
    Fin Pour ;  
  
    Afficher("Chauffeur non trouvé.") ;  
Fin  
  
Fonction modifierScoreChauffeur() ;  
    nomChauffeur : tableau de caractères de taille 50 ;  
    Afficher("Entrez le nom du chauffeur : ") ;  
    Lire(nomChauffeur) ;  
  
    i : entier ;  
    Pour i allant de 0 à nbChauffeurs - 1 faire ;  
        Si strcmp(chauffeurs[i].nom, nomChauffeur) = 0 alors ;  
            ancienScore : réel ;
```



# PROBLEMATIQUE DE GESTION DES SCORES

```
Fonction modifierScoreChauffeur() ;  
    nomChauffeur : tableau de caractères de taille 50 ;  
    Afficher("Entrez le nom du chauffeur : ") ;  
    Lire(nomChauffeur) ;  
  
    i : entier ;  
    Pour i allant de 0 à nbChauffeurs - 1 faire ;  
        Si strcmp(chauffeurs[i].nom, nomChauffeur) = 0 alors ;  
            ancienScore : réel ;  
            ancienScore <- chauffeurs[i].score ;  
            Afficher("Ancien score du chauffeur ", chauffeurs[i].nom, " : ", ancienScore) ;  
  
            choix : entier ;  
            Afficher("+++++Selectionnez une option +++++") ;  
            Afficher("1. Livraison réussie") ;  
            Afficher("2. Livraison en retard") ;  
            Afficher("3. Perte de produit") ;  
            Lire(choix) ;  
  
            nouveauScore : réel ;  
            penalite : réel ;  
            Switch choix faire ;  
                Cas 1 :  
                    valeurLivraison : réel ;  
                    tempsEstime, tempsReel : entier ;  
                    Afficher("Entrez la valeur de la livraison : ") ;  
                    Lire(valeurLivraison) ;  
                    Afficher("Entrez le temps estimé de la livraison : ") ;  
                    Lire(tempsEstime) ;  
                    Afficher("Entrez le temps réel de la livraison : ") ;  
                    Lire(tempsReel) ;  
                    // Livraison réussie  
                    differenceTemps : réel ;  
                    differenceTemps <- tempsEstime - tempsReel ;  
                    nouveauScore <- ancienScore + valeurLivraison * 0.1 + differenceTemps * 0.2 ;  
  
                    Quitter Cas ;  
                Cas 2 :  
                    valeurLivraison : réel ;  
                    tempsEstime, tempsReel : entier ;  
                    Afficher("Entrez la valeur de la livraison : ") ;  
                    Lire(valeurLivraison) ;  
                    Afficher("Entrez le temps estimé de la livraison : ") ;  
                    Lire(tempsEstime) ;  
                    Afficher("Entrez le temps réel de la livraison : ") ;  
                    Lire(tempsReel) ;  
                    // Livraison en retard  
                    differenceTemps : réel ;  
                    differenceTemps <- tempsReel - tempsEstime ;  
                    nouveauScore <- ancienScore - valeurLivraison * 0.1 + differenceTemps * 0.2 ;  
  
                    Quitter Cas ;  
                Cas 3 :  
                    valeurLivraison : réel ;  
                    Afficher("Entrez la valeur de la livraison perdue : ") ;  
                    Lire(valeurLivraison) ;  
  
                    nouveauScore <- ancienScore - valeurLivraison * 0.1 ;  
                    penalite <- valeurLivraison * 0.5 ;
```



# PROBLEMATIQUE DE GESTION DES SCORES

```
Cas 3 :  
    valeurLivraison : réel ;  
    Afficher("Entrez la valeur de la livraison perdue : ") ;  
    Lire(valeurLivraison) ;  
  
    nouveauScore <- ancienScore - valeurLivraison * 0.1 ;  
    penalite <- valeurLivraison * 0.5 ;  
  
    Quitter Cas ;  
Default :  
    Afficher("Option invalide.") ;  
    Retourner ;  
Fin Switch ;  
  
chauffeurs[i].score <- nouveauScore ;  
  
Afficher("Nouveau score du chauffeur ", chauffeurs[i].nom, " : ", nouveauScore) ;  
Si penalite > 0.0 alors ;  
|   Afficher("Pénalité : ", penalite) ;  
Fin Si ;  
  
Retourner ;  
Fin Si ;  
Fin Pour ;  
  
Afficher("Chauffeur non trouvé.") ;  
Fin  
  
Debut Algorithme ;  
  
choix : entier ;  
  
Tant que choix différent de 0 faire ;  
    Afficher("+++++Menu :+++++");  
    Afficher("1. Ajouter un nouveau chauffeur") ;  
    Afficher("2. Consulter le score d'un chauffeur") ;  
    Afficher("3. Modifier le score d'un chauffeur") ;  
    Afficher("0. Quitter") ;  
    Afficher("Selectionnez une option : ") ;  
    Lire(choix) ;  
  
    Switch choix faire ;  
        Cas 1 :  
            ajouterChauffeur() ;  
            Quitter Cas ;  
        Cas 2 :  
            consulterScoreChauffeur() ;  
            Quitter Cas ;  
        Cas 3 :  
            modifierScoreChauffeur() ;  
            Quitter Cas ;  
        Cas 0 :  
            Afficher("Programme terminé.") ;  
            Quitter Cas ;  
        Default :  
            Afficher("Option invalide.") ;  
    Fin Switch ;  
Fin Tant que ;  
  
Fin Algorithme ;
```

# CONCEPTION AVEC UML



## Conception textuel:

Dans le cadre de notre projet de conception d'une application de gestion avancée des services de livraison, nous avons pris en compte un contexte spécifique où notre entreprise est sollicitée pour effectuer des livraisons de produits entre différentes villes. Le processus de livraison implique plusieurs acteurs et paramètres clés qui seront détaillés ci-dessous.

Tout d'abord, nous avons l'entreprise cliente ou le "seller" qui souhaite expédier ses produits d'une ville à une autre. Le seller va déposer ses produits dans un dépôt ou un bureau situé dans la ville d'origine. Ces produits seront stockés en attendant d'être récupérés par un chauffeur indépendant pour effectuer la livraison.

Ensuite, nous avons les chauffeurs indépendants, qui sont les acteurs principaux de notre application. Chaque chauffeur dispose d'une voiture personnelle et est responsable de prendre en charge les produits à livrer depuis le dépôt de la ville d'origine et de les transporter jusqu'au bureau de destination. Les chauffeurs indépendants doivent être inscrits sur notre plateforme et satisfaire à certains critères de sélection tels que disponibilité, score de confiance, capacité de livraison en termes de volume et de poids, ainsi que le type de voiture dont ils disposent.

Le processus de sélection des chauffeurs se fait en utilisant un algorithme avancé qui attribue un score de confiance à chaque chauffeur. Ce score est calculé en prenant en compte plusieurs paramètres, tels que l'ancienneté du chauffeur, le nombre de livraisons réussies effectuées, les retards éventuels, les pénalités, ainsi que le coefficient de ressource du chauffeur, qui prend en compte la capacité de livraison de sa voiture. Plus le coefficient de ressource est élevé, c'est-à-dire plus le chauffeur peut livrer de produits en une seule fois, plus son score de confiance sera élevé.

Le score de confiance du chauffeur est initialisé lors de son inscription sur la plateforme. À partir de là, le score est mis à jour en fonction des événements qui surviennent lors des livraisons. Par exemple, une livraison réussie dans les délais impartis augmentera le score de confiance du chauffeur, tandis qu'un retard ou une perte de produits pourrait entraîner une diminution de son score.

# CONCEPTION AVEC UML



## Conception textuel:

Le processus de livraison implique également la gestion des trajets. Une ville peut être à la fois une ville de départ, une ville d'arrivée ou une ville de correspondance pour les livraisons. L'algorithme d'acheminement intelligent prend en compte la distance entre les différentes villes, le trafic, ainsi que les contraintes horaires pour déterminer le meilleur itinéraire possible.

Pour assurer la fiabilité des livraisons, notre entreprise met en place des protocoles stricts pour minimiser les risques de perte de produits. En cas de perte avérée, des procédures de résolution sont mises en place, telles que l'indemnisation du seller et la recherche de solutions alternatives pour la livraison.

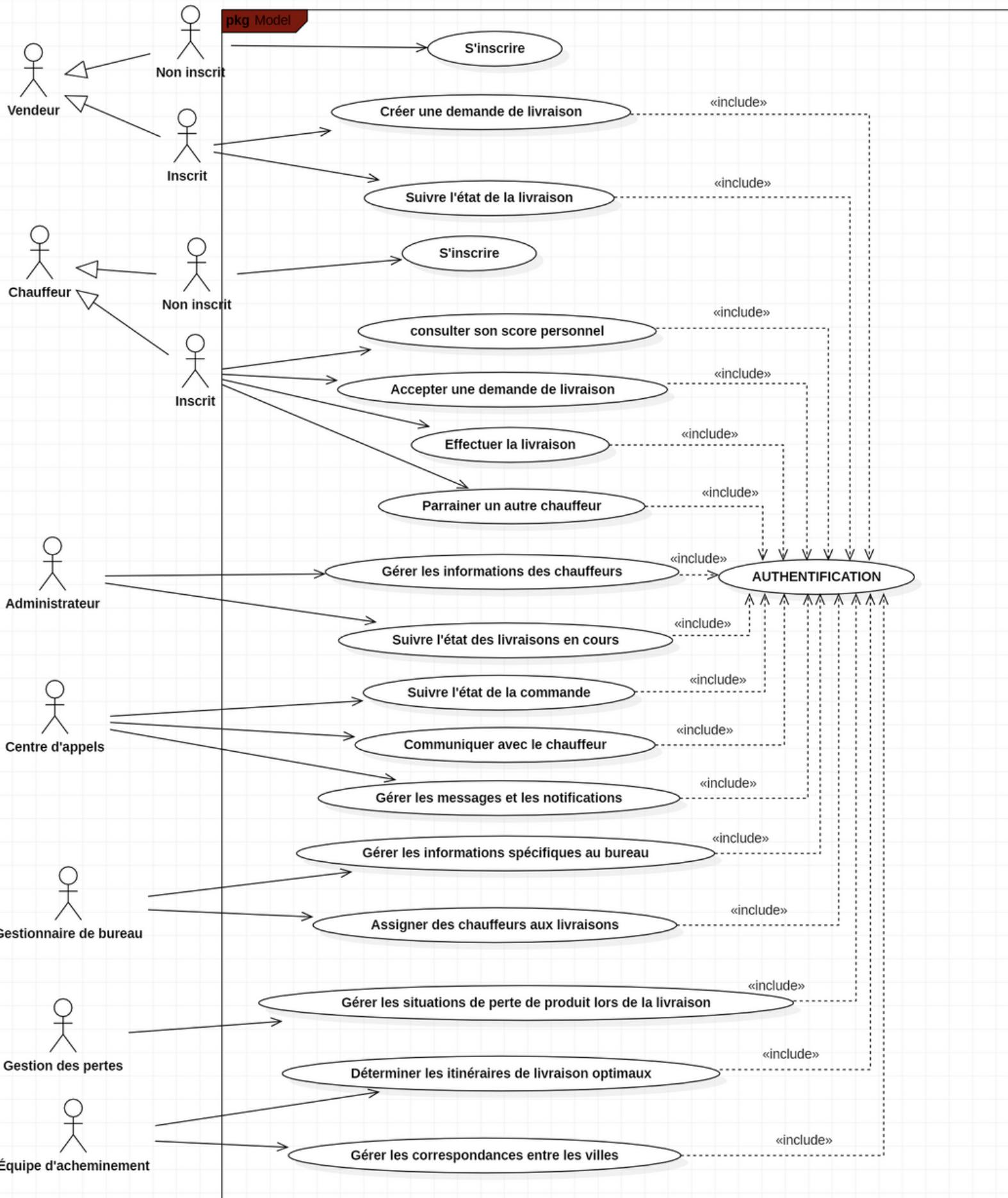
Outre les sellers, les chauffeurs indépendants, les dépôts et les bureaux de destination, notre application prend également en compte d'autres services et paramètres pour assurer le succès des livraisons. Cela peut inclure la gestion des paiements, les notifications en temps réel pour les sellers et les chauffeurs, le suivi des colis, la gestion des retours et des remboursements, ainsi que la mise en place d'un programme de parrainage où les chauffeurs peuvent augmenter leur score de confiance s'ils parrainent avec succès d'autres chauffeurs compétents.

En conclusion, notre application de gestion avancée des services de livraison vise à faciliter et optimiser le processus de livraison de produits entre différentes villes. En prenant en compte les sellers, les chauffeurs indépendants, les dépôts, les bureaux de destination et en utilisant un algorithme de sélection des chauffeurs basé sur le score de confiance, nous cherchons à fournir un service fiable, efficace et rentable. Tout en offrant une interface conviviale pour les utilisateurs et en intégrant des fonctionnalités avancées telles que la gestion des trajets, la résolution des incidents et la gestion des retours, notre application vise à améliorer l'expérience globale de livraison pour tous les acteurs impliqués.



# CONCEPTION AVEC UML

## DIAGRAMME DE CAS D'UTILISATION



# CONCEPTION AVEC UML



## la description des classes avec les règles de gestion, les attributs et les méthodes associés :

### 1. Seller (Vendeur)

- Attributs :
  - nom : nom du vendeur
  - adresse : adresse du vendeur
- Méthodes :
  - créerDemandeLivraison(détailsProduit, villeOrigine, villeDestination) : crée une demande de livraison pour un produit spécifié, avec la ville d'origine et la ville de destination.
  - suivreLivraison() : permet au vendeur de suivre l'état de la livraison de ses produits.

### 2. Driver (Chauffeur)

- Attributs :
  - nom : nom du chauffeur
  - véhicule : type de véhicule du chauffeur
  - capacitéMax : capacité maximale du véhicule du chauffeur
- Méthodes :
  - s'inscrire(nom, véhicule) : permet au chauffeur de s'inscrire en fournissant son nom et les détails de son véhicule.
  - accepterLivraison(demandeLivraison) : accepte une demande de livraison spécifiée.
  - effectuerLivraison() : effectue la livraison assignée.

### 3. Admin (Administrateur)

- Attributs :
  - nom : nom de l'administrateur
- Méthodes :
  - gérerChauffeurs() : permet à l'administrateur de gérer les informations des chauffeurs.
  - suivreLivraisons() : permet à l'administrateur de suivre l'état des livraisons en cours.

### 4. CallCenter (Centre d'appels)

- Attributs :
  - nom : nom du centre d'appels
- Méthodes :
  - suivreCommande(commande) : permet au centre d'appels de suivre l'état d'une commande spécifiée.
  - communiquerChauffeur(message) : permet au centre d'appels de communiquer avec le chauffeur en lui envoyant un message.
  - gérerMessagesNotifications() : gère les messages et notifications liés aux livraisons et aux chauffeurs.

### 5. OfficeManager (Gestionnaire de bureau)

- Attributs :
  - nom : nom du gestionnaire de bureau
  - bureau : bureau spécifique géré par le gestionnaire
- Méthodes :
  - gérerInformationsBureau() : permet au gestionnaire de bureau de gérer les informations spécifiques à son bureau.
  - assignerChauffeurLivraison(demandeLivraison, chauffeur) : assigne un chauffeur à une demande de livraison spécifiée.

# CONCEPTION AVEC UML



## La description des classes avec les règles de gestion, les attributs et les méthodes associés :

### 6. LossManagement (Gestion des pertes)

- Attributs :

    demandesRemboursement : liste des demandes de remboursement liées aux pertes de produits

- Méthodes :

    soumettreDemandeRemboursement(demandeLivraison, produitPerdu) : soumet une demande de remboursement suite à une perte de produit lors de la livraison.

### 7. RoutingTeam (Équipe d'acheminement)

- Méthodes :

- déterminerItinéraireOptimal(villeOrigine, villeDestination) : détermine l'itinéraire optimal pour la livraison entre une ville d'origine et une ville de destination.

### 8. Commande

- Attributs:

- numero de commande : un numero permettant de suivre la commande lors de la livr

## Associations :

1. vendeur et produit: Association 1 à N (un vendeur peut avoir plusieurs produits)

2. vendeur et Commande : Association 1 à N (un vendeur peut passer plusieurs commandes)

3. chauffeur et Commande : Association N à M (un chauffeur peut effectuer plusieurs livraisons, et une livraison peut être effectuée par plusieurs chauffeurs)

4. Admin et vendeur : Association 1 à N (un administrateur gère plusieurs chauffeurs)

5. centre d'appels et Commande : Association 1 à N (un centre d'appels peut suivre plusieurs commandes)

6. centre d'appels et chauffeur : Association 1 à N (un centre d'appels peut communiquer avec plusieurs chauffeurs)

7. gestionnaire de bureau et chauffeur : Association 1 à N (un gestionnaire de bureau gère plusieurs chauffeurs)

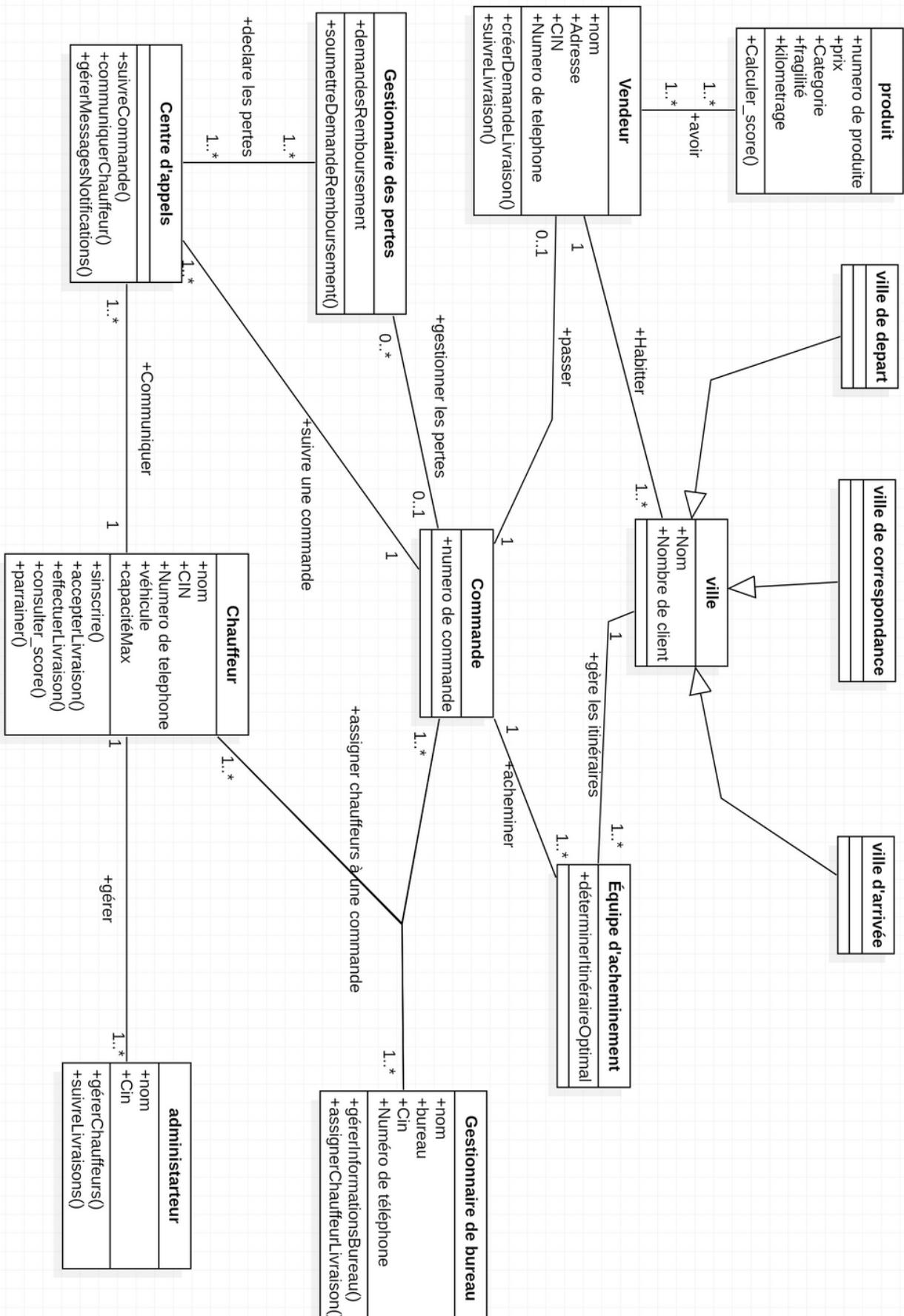
8. gestionnaire de bureau et Commande : Association 1 à N (un gestionnaire de bureau peut assigner plusieurs chauffeurs à une commande)

9. Gestionnaire des pertes et Commande : Association 1 à N (la gestion des pertes concerne plusieurs commandes)

10. l'équipe d'acheminement et Ville : Association 1 à N (l'équipe d'acheminement gère les itinéraires entre plusieurs villes)

# CONCEPTION AVEC UML

## Diagramme de CLASSE



# CONCEPTION DE BASE DE DONNÉE

## **Modèle Conceptuel de Données (MCD) pour la base de données du système de livraison:**

La conception d'une base de données est une étape essentielle dans le développement d'une application informatique. Cela implique la modélisation et la structuration des données qui seront stockées et manipulées par le système. Une conception efficace de la base de données est cruciale pour assurer la performance, la fiabilité et la pérennité de l'application.

Dans le contexte de notre projet, qui vise à simplifier le processus de gestion des services de livraison, la conception de la base de données revêt une importance particulière. La base de données doit être capable de stocker les informations liées aux vendeurs, aux chauffeurs, aux livraisons, aux produits, ainsi qu'aux différentes entités impliquées dans le processus, telles que les bureaux, les équipes de routage et de communication, etc.

L'objectif de cette étape de conception est de définir une structure de données optimale, en prenant en compte les besoins fonctionnels et les contraintes du système. Cela inclut l'identification des entités principales, la définition de leurs attributs, ainsi que les relations et les contraintes qui les lient.

Dans ce rapport, nous présenterons la conception de la base de données pour notre application de gestion des services de livraison. Nous décrirons les entités principales, les relations entre ces entités, ainsi que les attributs associés. Nous mettrons également en évidence les contraintes et les cardinalités qui régissent ces relations.

Cette conception de base de données nous permettra de stocker et de gérer efficacement les informations nécessaires au bon fonctionnement de notre application, en garantissant une manipulation fiable et cohérente des données.



# CONCEPTION DE BASE DE DONNÉE

## Modèle Conceptuel de Données (MCD) pour la base de données du système de livraison:

1. Entité "Commande":

- Attributs : Numero\_commande, Date\_commande.

2. Entité "Bureau":

- Attributs : ID\_Bureau, Location, ChiffrreDaffaire.

3. Entité "Vendeur":

- Attributs : CIN, Nom, Prénom, Numero\_Telephone, Adresse.

4. Entité "Admin":

- Attributs : CIN, Nom, Prenom, Numero\_telephone.

5. Entité "Ville":

- Attributs : Nom, CodePostale.

6. Entité "Produit":

- Attributs : CodeProduit, Nom, Poids, Prix, Categorie, Kilometrage.

7. Entité "EquipePerte":

- Attribut : Numero\_de\_bureau.

8. Entité "Perte":

- Attributs : NumeroPertes, DatePerte.

9. Entité "CentreDappele":

- Attributs : ID, Message.

10. Entité "EquipeAcheminement":

- Attribut : Numero\_de\_bureau.

11. Entité "GestionnaireBureau":

- Attributs : CIN, Nom, Prenom, Salaire, ID\_Bureau.

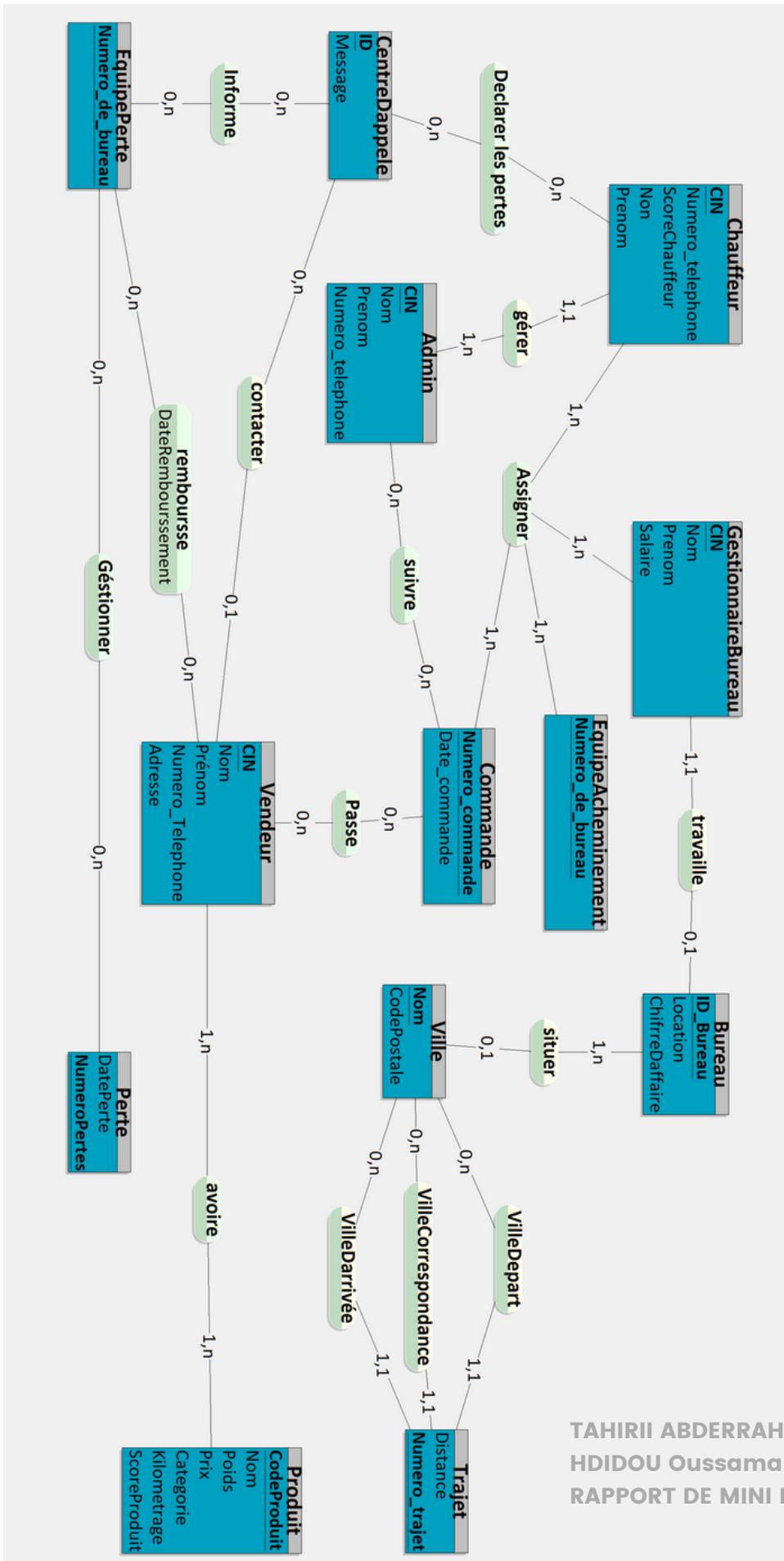
12. Entité "Chauffeur":

- Attributs : CIN, Numero\_telephone, Score, Non, Prenom, CIN\_1.

# **CONCEPTION DE BASE DE DONNÉE**



## **Modèle Conceptuel de Données (MCD) pour la base de données du système de livraison:**



# TAHIRII ABDERRAHMANE

## HDIDOU Oussama

### RAPPORT DE MINI PROJET



# CONCLUSION

En conclusion, ce projet vise à simplifier la gestion des services de livraison d'une entreprise au Maroc grâce à une application innovante. Nous avons comparé cette approche avec l'approche classique de faire appel à des sociétés de livraison, en identifiant les avantages et les risques associés. Nous avons défini les acteurs impliqués, tels que les entreprises, les chauffeurs indépendants et les équipes de gestion. Le score de confiance des chauffeurs joue un rôle clé, calculé en fonction de plusieurs paramètres. L'algorithme de sélection des chauffeurs prend en compte le score de confiance, la disponibilité, les destinations et les contraintes horaires. L'approche proposée offre une flexibilité accrue et une amélioration de l'efficacité des livraisons. Des perspectives d'amélioration incluent la gestion des bureaux, des pertes de produits et l'extension des fonctionnalités. Ce projet offre une solution novatrice pour optimiser les livraisons et réduire les coûts, répondant ainsi aux besoins évolutifs des entreprises et des clients.