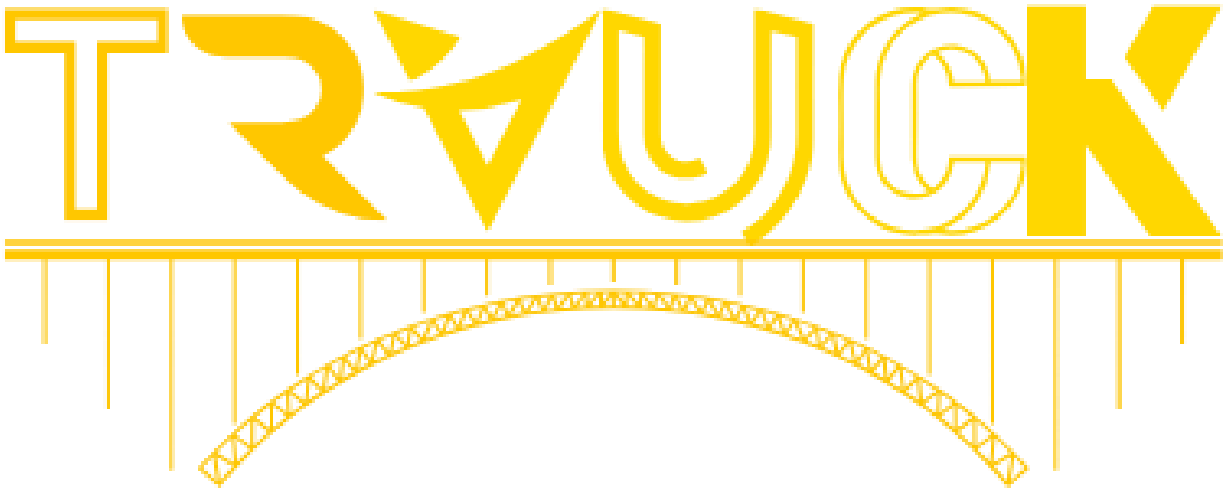
MODULISATION DU SYSTEME



CONCEPTION A BASE ENGOLOPHONE : UML

I- DEFINITION ET CONTEXTE :

UML, pour *Unified Modeling Language*, est un langage de modélisation visuel utilisé pour spécifier, visualiser, construire et documenter les composants d'un système logiciel. Il permet de représenter les différents aspects d'un projet logiciel, notamment la structure, le comportement et les interactions entre les différents composants et acteurs.

Caractéristiques principals d'UML :

1. Standardisation : UML est un standard international pour la modélisation, permettant aux équipes de développement de représenter leurs systèmes de manière cohérente et compréhensible par tous.
2. Multi dimensionnalité : UML propose plusieurs types de diagrammes, chacun illustrant un aspect particulier du système (structure, comportement, interaction, déploiement, etc.). Ces diagrammes permettent de comprendre le système sous différents angles.
3. Abstraction : UML facilite la décomposition d'un système complexe en composants plus simples, permettant ainsi de mieux appréhender et concevoir des solutions pour chaque partie du système.

Types de Diagrammes en UML :

UML se compose de 14 types de diagrammes répartis en trois grandes catégories :

1. Diagrammes de Structure : Ils montrent l'architecture statique du système, y compris les classes, objets, composants et leurs relations.
    - Exemples : Diagramme de classe, diagramme de composants, diagramme de déploiement.
2. Diagrammes Comportementaux : Ils illustrent le fonctionnement interne et les processus du système.
    - Exemples : Diagramme de cas d'utilisation, diagramme d'activité, diagramme d'état.
3. Diagrammes d'Interaction : Ils se concentrent sur la communication et les échanges entre les objets ou les composants.
    - Exemples : Diagramme de séquence, diagramme de communication, diagramme de synchronisation.

Utilisation d'UML

UML est largement utilisé dans le développement de logiciels pour :

- Analyser les besoins et comprendre le fonctionnement attendu du système.
- Concevoir l'architecture en définissant les composants et leurs interactions.
- Communiquer la conception aux parties prenantes, qu'il s'agisse d'ingénieurs, de gestionnaires ou de clients.
- Documenter le système, fournissant une référence précieuse pour la maintenance et les évolutions futures.

II- ETAPE DE MODELISATION :

Dans notre modélisation, j'ai commencée tous d'abord par choisir les principaux diagrammes qui vont faciliter la tâche au cours de réalisation d'UML

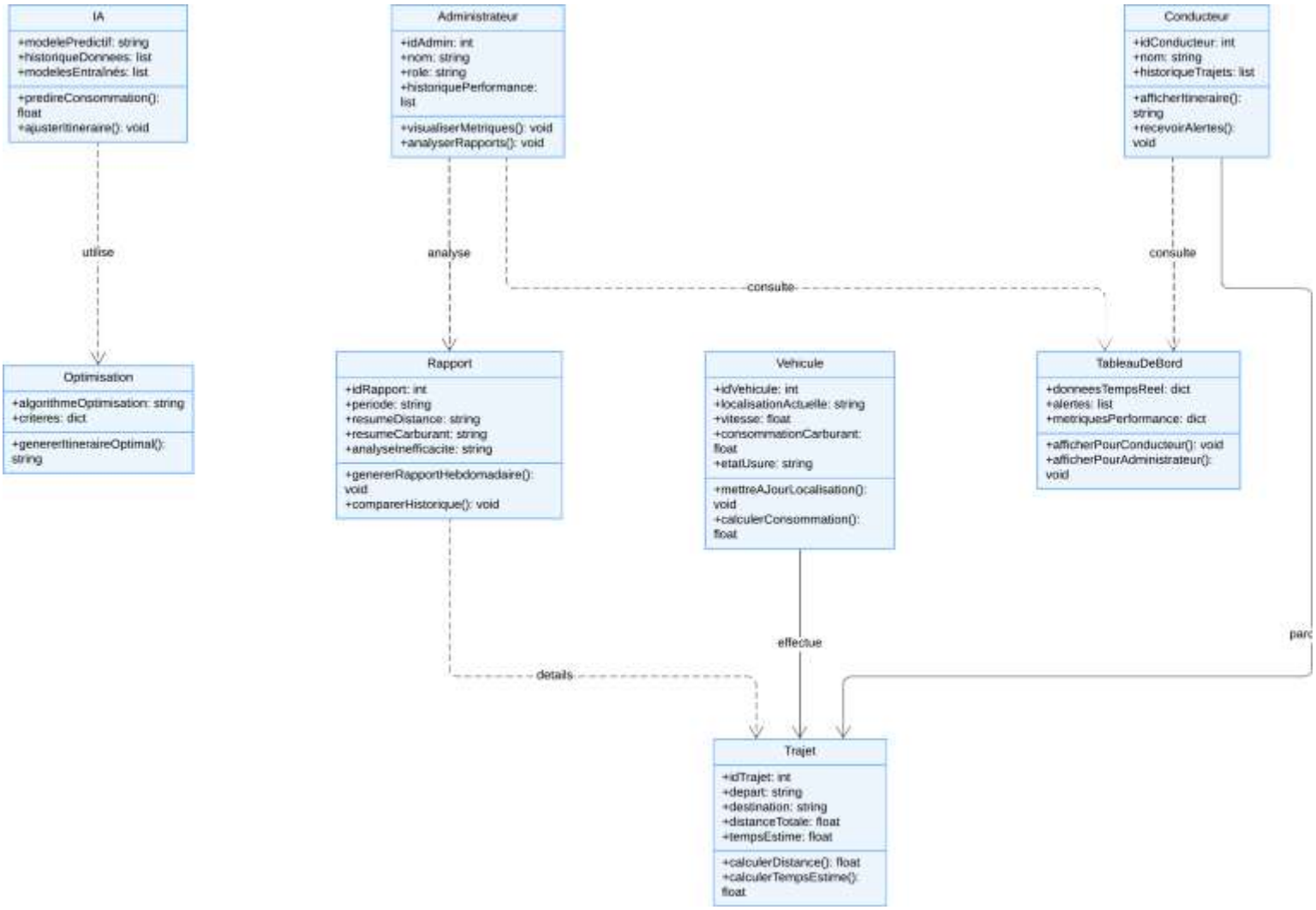
Pour moi, j'ai choisi les diagrammes suivant :

1- Diagramme de la classe :

Le diagramme de classe détaille la structure de données, les entités et les relations du système.

En UML, toute classe est identifiée par un attribut (l'ensemble des caractéristiques d'entité) et les Méthodes (c.-à-d., les fonctions de chaque attribut du classe)

1. **Véhicule**
   - Attributs : `idVéhicule`, `localisationActuelle`, `vitesse`, `consommationCarburant`, `étatUsure`
   - Méthodes : `mettreAJourLocalisation()`, `calculerConsommation()`
2. **Trajet**
   - Attributs : `idTrajet`, `départ`, `destination`, `distanceTotale`, `tempsEstime`
   - Méthodes : `calculerDistance()`, `calculerTempsEstime()`
3. **Conducteur**
   - Attributs : `idConducteur`, `nom`, `historiqueTrajets`
   - Méthodes : `afficherItinéraire()`, `recevoirAlertes()`
4. **Administrateur**
   - Attributs : `idAdmin`, `nom`, `role`, `historiquePerformance`
   - Méthodes : `visualiserMétriques()`, `analyserRapports()`
5. **Optimisation**
   - Attributs : `algorithmeOptimisation`, `critères` (distance, coût, délais)
   - Méthodes : `générerItinéraireOptimal()`
6. **Rapport**
   - Attributs : `idRapport`, `periode`, `résuméDistance`, `résuméCarburant`, `analyseInefficacité`
   - Méthodes : `générerRapportHebdomadaire()`, `comparerHistorique()`
7. **IA**
   - Attributs : `modèlePrédictif`, `historiqueDonnées`, `modèlesEntraînés`
   - Méthodes : `prédireConsommation()`, `ajusterItinéraire()`
8. **TableauDeBord**
   - Attributs : `donnéesTempsRéel`, `alertes`, `métriquesPerformance`
   - Méthodes : `afficherPourConducteur()`, `afficherPourAdministrateur()`



2- Diagramme de séquence:

Dans le diagramme de séquence en se base sur les scénarios possible qu'on peut avoir dans notre système, je vous présente ici deux exemple qui vont vous aider à comprendre 2 ensemble de scénario possible du système **:**

Exemple 1 : Optimisation d'un Trajet

Dans ce scénario, le conducteur souhaite obtenir l'itinéraire le plus optimal en fonction de la localisation actuelle du véhicule et des conditions de transport.

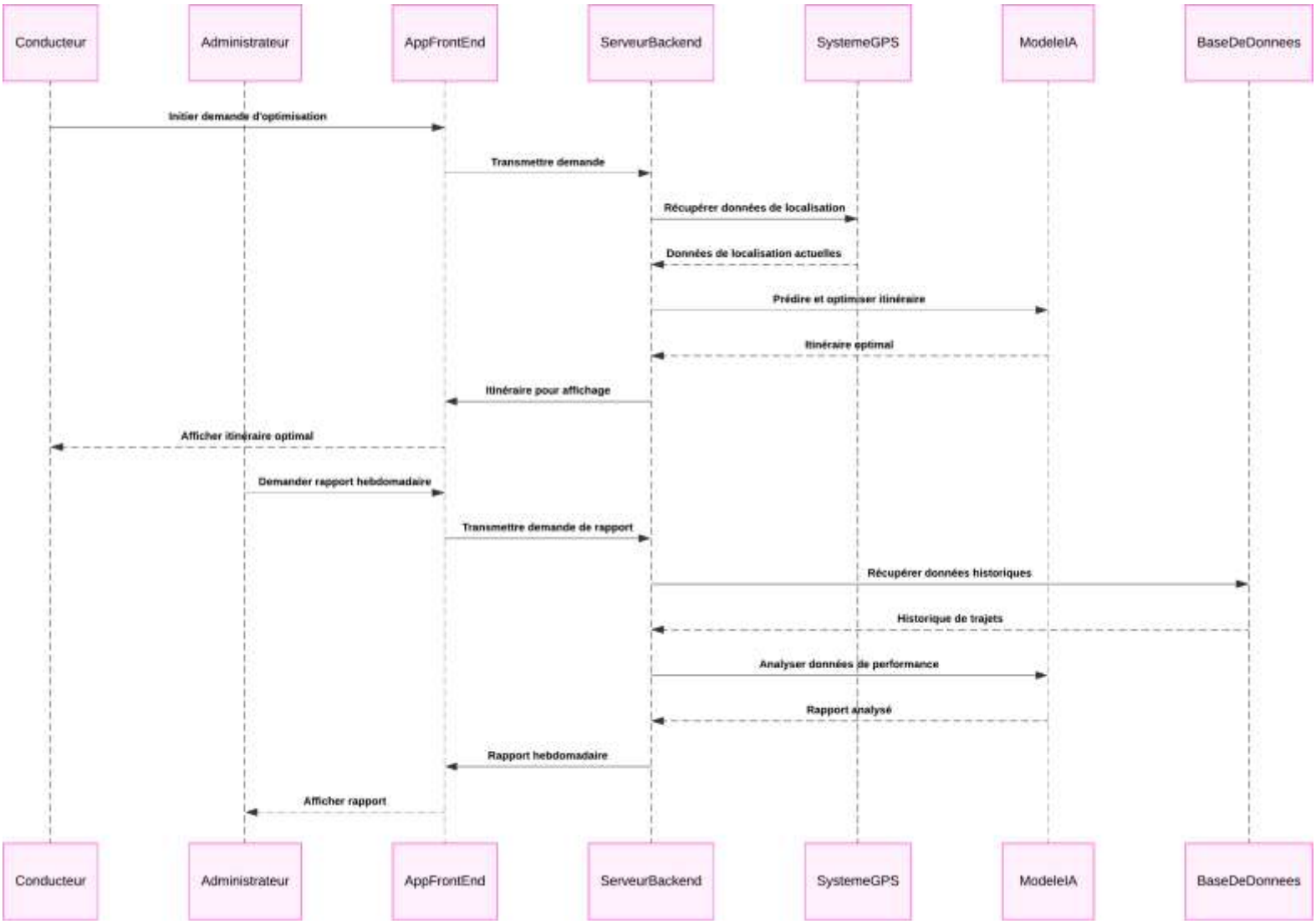
1. **Conducteur** initie une demande d'optimisation du trajet en accédant à l'interface de l'application depuis son appareil.
2. L'**application front-end** reçoit cette demande et envoie une requête au **serveur backend** pour traiter l'optimisation de l'itinéraire.
3. Le **serveur backend** demande les données de localisation actuelles au **système GPS** du véhicule.

4. Le **système GPS** envoie les informations de localisation en temps réel, y compris la position actuelle et la vitesse du véhicule, au **serveur backend**.
5. Le **serveur backend** traite les données reçues et consulte le **modèle IA** pour prédire les conditions de route et proposer une optimisation.
6. Le **modèle IA** utilise des algorithmes d'optimisation (Dijkstra, Bellman-Ford, etc.) et tient compte des critères de coût, de distance, de délais, ainsi que des données externes (météo, trafic) pour calculer l'itinéraire optimal.
7. Une fois l'itinéraire optimal déterminé, le **modèle IA** envoie les recommandations de trajet au **serveur backend**.
8. Le **serveur backend** renvoie ensuite l'itinéraire optimisé à l'**application front-end** du conducteur.
9. L'**application front-end** affiche l'itinéraire optimisé, les détails tels que l'heure estimée d'arrivée (ETA) et les alertes éventuelles, que le **conducteur** peut suivre en temps réel.

Exemple 2 : Génération de Rapport Hebdomadaire

Dans ce processus, l'administrateur demande un rapport hebdomadaire pour évaluer la performance des trajets, la consommation de carburant, et les gains d'efficacité.

1. **Administrateur** initie une demande de génération de rapport via l'interface web de l'application.
2. L'**application front-end** transmet la demande au **serveur backend** pour démarrer le processus de compilation des données.
3. Le **serveur backend** récupère l'historique des trajets, la consommation de carburant, et les données de performance depuis la **base de données**.
4. Le **serveur backend** envoie ces données au **modèle IA**, qui effectue une analyse des inefficacités en fonction des données historiques et actuelles.
5. Le **modèle IA** effectue une analyse prédictive pour identifier les améliorations potentielles, en tenant compte de critères tels que la consommation de carburant, la durée des trajets, et les tendances observées.
6. Le **modèle IA** envoie les résultats d'analyse au **serveur backend**, incluant des recommandations spécifiques pour améliorer la performance logistique.
7. Le **serveur backend** compile ces informations dans un rapport, résumant les indicateurs clés, les distances parcourues, les coûts énergétiques, et les recommandations d'optimisation.
8. Le rapport final est transmis à l'**application front-end**, où l'**administrateur** peut le consulter, analyser les performances et prendre des décisions en vue des futures optimisations



3- logigramme:

Le logigramme, (*flowchart*) joue un rôle crucial dans la modélisation de l'UX de l'utilisateur dans notre système. Il permet de visualiser les processus étape par étape, en montrant clairement les différentes étapes, décisions et actions qui mènent à l'accomplissement d'un objectif.

Dans notre contexte, le logigramme illustre les éléments suivants :

-les Processus d'Optimisation de Trajet

-le Processus de Génération de Rapports Hebdomadaires

- la Prise de Décision dans les Processus Basés sur l'IA

- la Communication et la Documentation

- Identification des Points de Faiblesse