Projet Java  
Rapport 2 : Java IO/ Base de données  
CAR RENTAL  
CGO

**Classe : 17VP**

|  |  |
| --- | --- |
| **Votre groupe**: | **Đinh Nguyễn Quốc Bảo – 1752007**  **Trần Huy Nam – 1752035**  **Lê Tôn Huy – 1752019** |

Tableau de contenu

[L'histoire de versions 3](#_Toc38459159)

[Introduction 4](#_Toc38459160)

[Analyse et conception 5](#_Toc38459161)

[Implémentation 6](#_Toc38459162)

[Exemples de données 7](#_Toc38459163)

[Résultat 8](#_Toc38459164)

[Plan 9](#_Toc38459165)

[Références 10](#_Toc38459166)

# L'histoire de versions

[*Ajoutez dans cette section un tableau des versions de votre projet, un numéro de version, la date de création, la description*]

**∙ CGO version 1.0 :** *Notre premier objectif est de créer tous les objets pertinents pour la location de voitures*

* Commencez le processus de création d’un modèle des objets (utilisateur, membre, administrateur)
* Date de créer : *2020-03-15*

**∙ CGO version 1.1 :**

* Commencer le processus de création d’un modèle d’objets (véhicule, revenus de location, revenus de location de semaines, revenus de location de mois)
* Date de créer : *2020-03-21*

**∙ CGO version 1.2 :**

* Commencer le processus de création d’un motif des objets (voiture, camion); remplacer les modèles (revenus locatifs de semaine, revenus locatifs de mois) par le modèle (commande)
* Date de créer : *2020-03-30*

**∙ CGO version 1.3 :**

* Commencer le processus de création d’un modèle des objets (vérifier, enregistrement de location)
* Date de créer : *2020-04-04*

**∙ CGO version 1.4:** *A cette version, nous essayons de connecter l’objet (réservation) dans notre projet.*

* Commencez le processus de création d’un motif des objets (réservation).
* Date de créer : *2020-04-16*

**∙ CGO version 1.5:**

* Initialisation réussie de l'objet, nous avons procédé à la lecture du fichier xml pour obtenir les données du véhicule.
* Date de créer : *2020-04-27*

**∙ CGO version 1.6:**

* Nous commençons par créer l'administrateur et octroyer des droits pour ajouter, supprimer, modifier des données sur le fichier xml.
* Date de créer : *2020-05-06*

*à suivre…*

# Introduction

[*Vous présentez ici le but de votre rapport, expliquer comment vous stocker les données de votre application (soit dans les fichiers, soit un système de gestion de la base de données), donnez la raison de votre choix.*]

* ***Le contexte du projet***
* Aujourd’hui, les gens louent un véhicule directement à l'entreprise ou au garage du concessionnaire, ce qui rend les choses gênantes car beaucoup de situations peuvent se produire, par exemple: problèmes de distance, le client veut louer en ce moment mais il est occupé donc ne peut pas être présent, ... Capturez ce problème, notre groupe décide de résoudre ce problème en créant une application qui peut aider les gens à louer une voiture complètement en ligne, et le nom de l'application est CGO.
* ***Donnez la raison pour laquelle vous développez ce projet***
* Avant de donner la raison du développement de ce projet, rejoignez-nous pour en savoir plus sur l'idée de cette application. CGO est une application de location de voiture simple avec une opération et une procédure faciles pour aider les utilisateurs à louer une voiture. Avec une fonctionnalité unique et facile à utiliser, CGO peut trouver, réserver et louer une voiture très rapidement. Avec CGO, notre objectif est d’aider l’utilisateur plus pratique et plus facile à louer une voiture.
* ***Indiquez les applications/ressources existants, ce que ces systèmes fournissent et proposez le vôtre, expliquez clairement les différences prévues entre le vôtre et l'application existante.***

- À l'heure actuelle, nous consultons toujours <https://www.mioto.vn/> et <https://ezbook.vn/> comme source. À propos de la différence entre notre projet et ces sources; premièrement, nous avons décidé de le faire comme une application de bureau, pas comme un site Web; deuxièmement, notre projet ne gère que la ville de HCM, pas l'ensemble du pays.

* ***Présentez également le résultat prévu de ce projet***

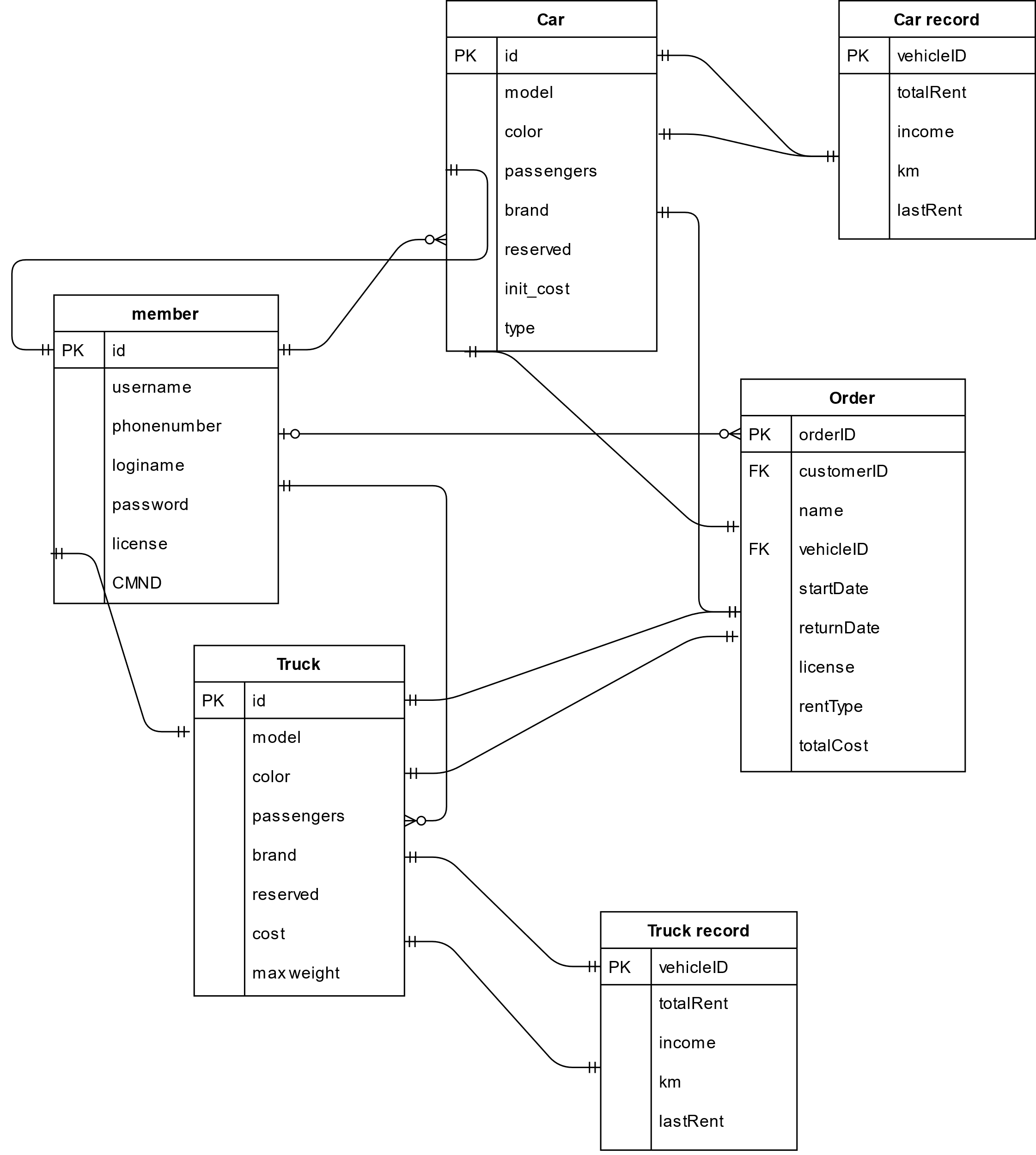
# Actuellement, nos attentes pour ce projet est de répondre à toutes nos cibles (trouver, réserver et louer une voiture), nous allons procéder à la mise à jour plus sur cette application bientôt.

# Analyse et conception

*[Présentez d'abord quelles informations doivent être stockées dans votre application.*

*Présentez en détaille comment vous organisez vos données. Par exemple:*

* *Si vous utilisez les fichiers pour sauvegarder vos données, alors indiquez le type du fichier (un fichier texte bruit, XML, JSON, etc.) et le format/schéma de vos données.*
* *Si vous utilisez un système de gestion de base de données, alors indiquez le nom du système que vous utilisez, désignez la structure de données, construisez les relations entre les tableaux, etc. ]*
* Notre application est une application de location de voiture, donc nous stockons 6 informations principales (jusqu’à présent) : informations sur les voitures et les camions, commande, historique de location de voiture, historique de location de camion et membre.
* Nous stockons et utilisons ces données en lisant le fichier XML et en l'ajoutant à notre base de données. Notre équipe choisit cette méthode car elle est simple et assez facile à utiliser.
* Pour illustrer notre projet, nous créons un schéma relationnel de location de voiture :



* Je vais vous expliquer ce schéma :

+ Un membre peut louer une ou plusieurs voitures, camions, de sorte que la relation entre la table MEMBER et la table CAR est une à plusieurs (1, n), et la relation entre la table MEMBER et la table TRUCK est également une à plusieurs ( 1, n).

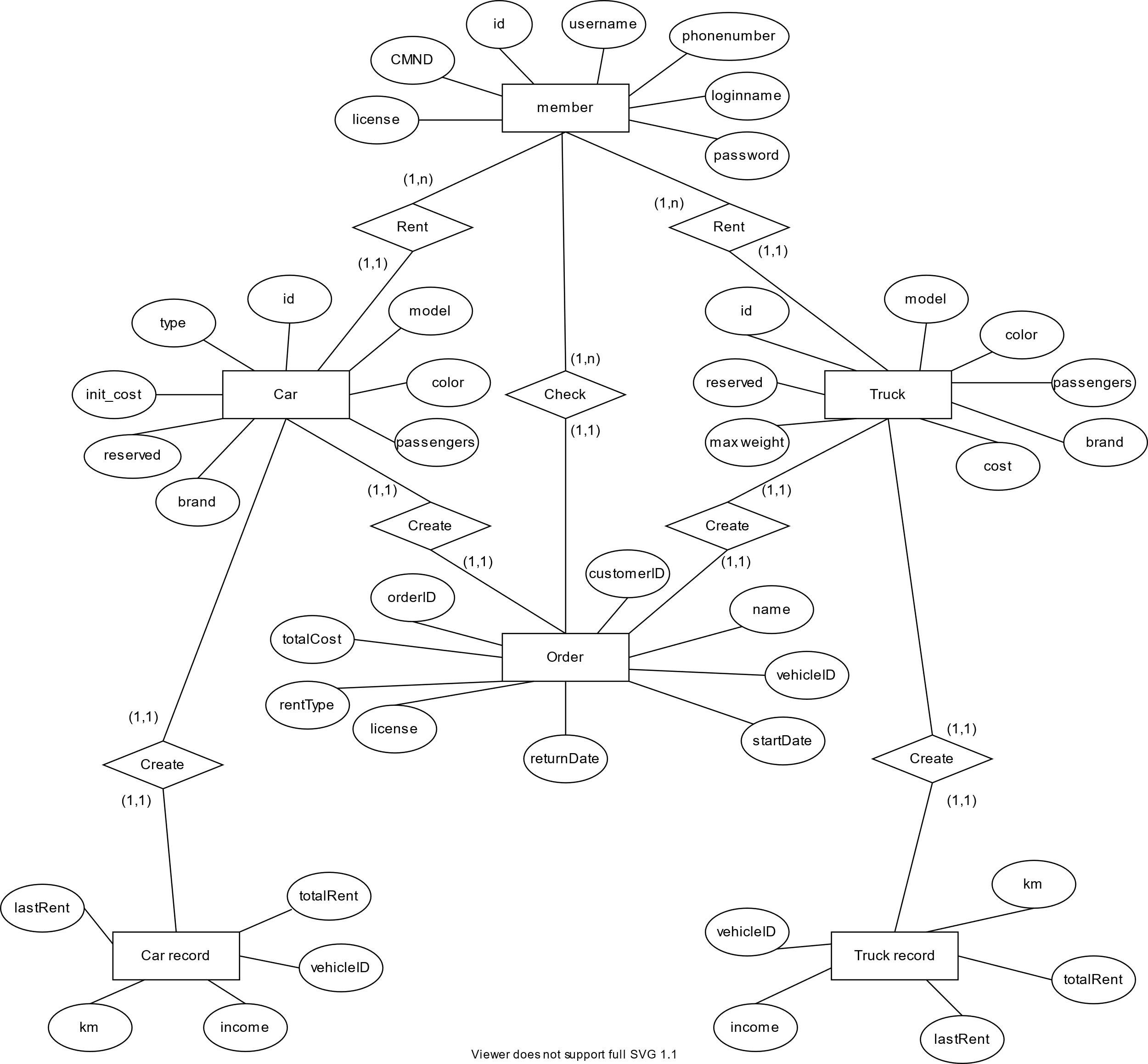
+ Une voiture ne peut être louée que par un et un seul membre, de sorte que la relation entre la table CAR et la table MEMBER est une à une (1, 1), et la relation entre la table TRUCK et la table MEMBER est également plusieurs ( 1, 1).

+ chaque voiture ne peut avoir qu'une seule commande et chaque commande peut être pour une seule voiture, donc la relation entre la table CAR et la table ORDER est une à une(1,1), la relation entre la table ORDER et la table CAR est également une à une(1,1).

+ Chaque camion ne peut avoir qu'une seule commande et chaque commande ne peut concerner qu'un seul camion, donc la relation entre la table TRUCK et la table ORDER est de un à un (1,1), la relation entre la table ORDER et la table TRUCK est également un à un (1.1).

+ Chaque commande de voiture ne peut avoir qu'un seul enregistrement de voiture et chaque enregistrement de voiture ne peut concerner qu'une seule commande de voiture, de sorte que la relation entre la table ORDER et la table CAR RECORD est de un à un (1.1), la relation entre la table CAR RECORD et la table ORDER est également une par une (1.1).

+ Chaque commande de camion ne peut avoir qu'un seul enregistrement de camion et chaque enregistrement de camion ne peut se rapporter qu'à une seule commande de camion, de sorte que la relation entre la table ORDER et la table TRUCK RECORD est de un à un (1.1), la relation entre la table TRUCK RECORD et la table ORDER est également un par un (1.1).

* Je crée également un diagramme ER pour visualiser :

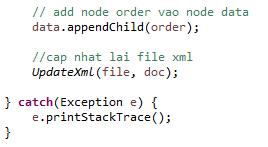
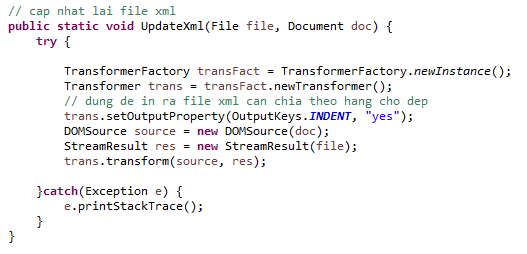
# Implémentation

*[Présentez ici comment vous chargez les données depuis le fichier entré ou sauvegardez les données dans le fichier de sortie/ comment vous vous connectez avec la base de données pour charger, modifier ou supprimer les informations, etc. en utilisant Java]*

*  **Afficher toutes les commandes dans le fichier « Order.xml »**

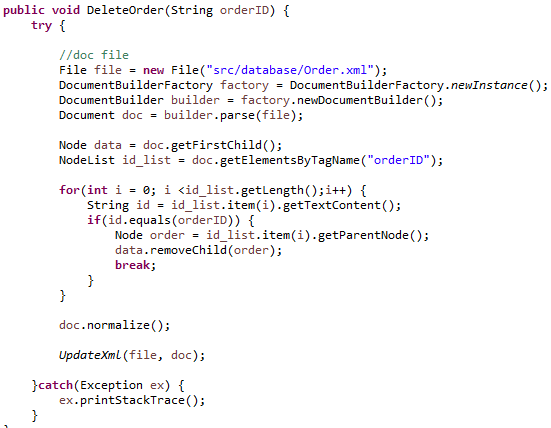
Cette fonction utilise un « DocumentBuilderFactory », un « DocumentBuilder » pour analyser du fichier « Order.xml » dans un « Document ». Après, le « Document » obtient une liste des commandes par une fonction « getElementByTagName(« order ») et les stocke dans une « NodeList ». Cette liste obtient de commande ayant la position « num » et cette commande se tranforme en « Element ». Alors ce « Element » obtient la valeur de text pour chaque la balise enfant dans la balise parent « order » et l’assigner à l’attribut correspondant dans la classe « Order ». Par exemple : this.orderID = element.getElementByTagName(« orderID »).getTextContent() ;

* **Ajouter une commande au fichier « Order.xml »**

La méthode pour accéder le fichier est le même. Après, le “Document” obtient le nœud racine “data” et crée un nœud “order”. Puis, le document crée les nœudes et ses valeur de texte est les valeur de l’attribut de l’objet “Order”. Ces nœudes sont les enfant du nœud “order”.

Ensuite, le nœud “data” ajoute le nœud “order”. La fonction appel la fonction UpdateXml(file, doc). Cette fonction est utilisé pour écrire tout le contenu de “Document doc” dans le fichier “file”.

* **Supprimer une commande**



La fonction trouve la commande ayant l’identité le même à “orderID”. Le nœud racine “data” va supprimer le nœud enfant “order” ayant cet identité.

# Exemples de données

*[Donnez ici un exemple de la structure de données et donnez votre explication pour chaque exemple. C'est les données que vous pouvez utiliser pour tester votre application plus tard...]*

* Structure de données utilisée :

+ NODE :

* Par exemple :



* Expliquer :

Dans cet exemple, j'utilise le nœud pour obtenir une balise <Car> située à la position numérique (num) que l'utilisateur souhaite exporter dans la liste des voitures disponibles.

+ NODE\_LIST :

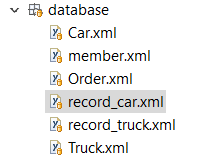
* Par exemple :



* Expliquer :

Dans cet exemple, j'utilise node\_list pour obtenir une liste de toutes les balises <car> ... </car> contenues dans la base de données.

* Les tables sont créées dans la section base de données :



1. - Car.xml est équivalent à la table Car, utilisée pour stocker les détails de chaque voiture.

* Dans la table Car, il existe des propriétés équivalentes aux colonnes id, modèle, couleur, passagers, marque, réservé, init\_cost et type.



+ id : chaque voiture aura un identifiant unique

+ modèle : enregistrer le nom de la voiture

+ couleur : enregistrez la couleur de la voiture

+ passagers : enregistrez le numéro de siège de cette voiture

+ brand : enregistrer le nom de marque d'une voiture

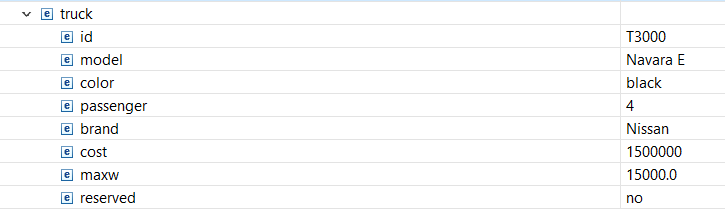
+ réservé : enregistrer l'état actuel de la voiture (louée (oui) ou non louée (non))

+ init\_cost : économisez le loyer de cette voiture

+ type : enregistrez le nom et le type de voiture de cette voiture

1. - Truck.xml est équivalent à une table Truck, utilisée pour stocker des informations détaillées sur chaque camion

* Dans la table Truck, il existe des propriétés équivalentes aux colonnes id, modèle, couleur, passagers, marque, coût, maxw et réservé.



+ id : chaque camion aura un identifiant unique

+ modèle : enregistrez le nom de chaque camion

+ couleur : enregistrez la couleur du camion

+ passagers : enregistrez le numéro de siège de ce camion

+ brand : enregistrez le nom de marque du camion

+ cost : économiser le prix de location de ce camion

+ maxw : enregistrez la charge maximale de ce camion

+ réservé : enregistrer l'état actuel de ce camion (loué (oui) ou non loué (non))

1. - Order.xml est équivalent à la table Order, utilisée pour stocker les détails de chaque commande client

* Dans la table Order, il existe des propriétés équivalentes aux colonnes : orderID, customerID, name, vehicleID, startDate, returnDate, license, rentType, totalCost



+ orderID : chaque commande aura un identifiant unique

+ customerID : chaque client aura un identifiant unique

+ nom : enregistrer le nom du client

+ ID véhicule : ID du véhicule de location (voiture (C1xxx) ou camion (T3xxx))

+ startDate : enregistrez la date de début du bail

+ returnDate : date d'expiration de la location

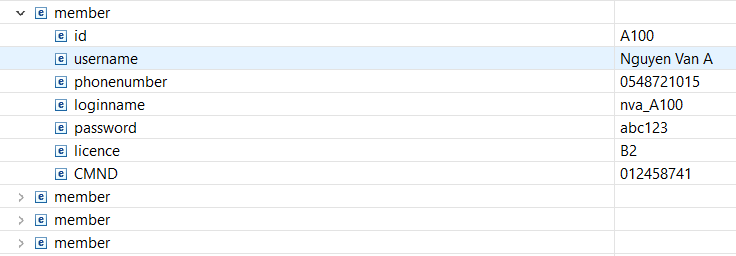
+ licence: enregistrer le permis de conduire

+ rentType : type de loyer (1: hebdomadaire, 2: mensuel)

+ totalCost : Prix total de location de voiture (camion)

1. member.xml est équivalent à table member, qui est utilisé pour stocker les informations sur les membres lorsque les clients s'enregistrent

* Dans la table, le membre comprend des attributs équivalents aux colonnes: id, nom d'utilisateur, numéro de téléphone, nom de connexion, mot de passe, licence, carte d'identité



+ id : chaque membre aura un identifiant unique

+ nom d'utilisateur : le nom du membre

+ numéro de téléphone : numéro de téléphone du membre

+ loginname : nom d'utilisateur

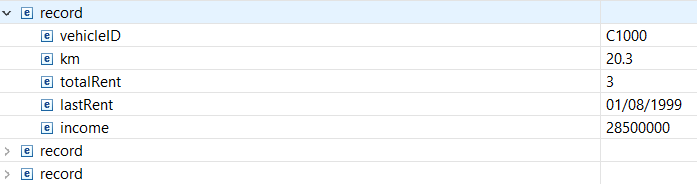
+ password : mot de passe pour se connecter

+ licence : enregistrer le permis de conduire

+ Carte d'identité : enregistrez la carte d'identité du membre (CMND)

1. - record\_car.xml est équivalent à la table record\_car, qui est utilisée pour enregistrer des informations après chaque location de chaque voiture.

* Dans la table record\_car, il existe des propriétés équivalentes aux colonnes VehicleID, km, totalRent, lastRent et revenue.



+ vehicleID : ID du véhicule de location (voiture (C1xxx) ou camion (T3xxx))

+ km : enregistrez le nombre de kilomètres parcourus par le véhicule

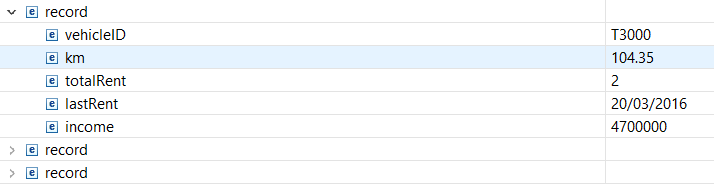
+ totalRent : nombre total de locations

+ lastRent : dernière date de location

+ income : statistiques de vente d'un véhicule

1. record\_truck.xml est équivalent à la table record\_truck, qui est utilisée pour enregistrer des informations après chaque location de chaque camion

* Dans la table record\_truck comprend également des attributs équivalents aux colonnes VehicleID, km, totalRent, lastRent et revenue.



+ vehicleID : ID du véhicule de location (voiture (C1xxx) ou camion (T3xxx))

+ km : enregistrez le nombre de kilomètres parcourus par le véhicule

+ totalRent : nombre total de locations

+ lastRent : dernière date de location

+ income : statistiques de vente d'un véhicule

# Résultat

[*Expliquez ce que vous obtenez jusqu'à présente, des avantages, désavantages et les solutions prévues (si c'est possible)*]

* Remplissez la majeure partie de la base de données de l'application
* ***Avantages:***

+ Ajoutez la fonction pour ajouter et supprimer des applications.

+ Mise à jour des données des fichiers XML pour des applications plus diverses, avec plus de nouvelles données.

+ Il existe un fichier XML séparé pour enregistrer les commandes des clients qui ont effectué une location de voiture.

* ***Inconvénients:***

+ Il y a encore des difficultés à modifier le fichier XML.

# Plan

[*Donnez le plan de votre projet (en détaille) jusqu'à la fin du projet: qui fait quoi, la durée de chaque tâche*]

C’est le plan estimée pour ce projet.

* Semaine 1 : Le groupe trouve l’idée du projet, choisit et définit les classes nécessaires et les structures de ce projet.
* Semaine 2, 3 : Bảo va créer des classes principals dans Java. Les classes : Order, User, Admin, Member et Nam va créer les classes Vehicule, Car, Truck. Huy modifie et ajoute les attributs si nécessaire.
* Semaine 4, 5 : Huy et Nam créent des bases de données nécessaire (Notre projet utilise XML pour stocker les données). Bảo implémente les méthodes ajouter (la fonction AddOrder) et supprimer (la fonction DeleteOrder) pour « Order » ; Bảo écrire la fonction updateXML à utiliser pour ajouter, supprimer et modifier des fonctions. Nam implémente les méthodes ajouter (la fonction AddCar) et supprimer (la fonction DeleteCar) pour « Car ». Huy implémente les méthodes ajouter (la fonction AddTruck et supprimer (la fonction DeleteTruck) pour « Truck ». Bao a complété 2 fichiers member.xml et Order.xml, Nam a complété Car.xml et record\_car.xml, Huy a complété Truck.xml et record\_truck.xml.
* Semaine 6, 7 : Chaque personne implémente des méthodes dans les classes. Huy -> classe « Order » et « Rental\_record » ; Nam -> classe « Vehicle » et ses subclasses ; Bảo -> classe « User » et ses subclasses.
* Semaine 8, 9 : Chaque personne crée des GUIs nécessaire. Huy -> la page de connéxion et d’inscription. Bảo -> la page d’accueil et la page de gestion des voitures. Nam -> la page de gestion des commandes et la page d’afficher des véhicules.

# Références

[*Listez tous les ressources à utiliser dans votre projet, y compris les codes existantes, les algorithmes utilisés, les livres, les rapports, les liens à consulter, etc.*]

* Comment créer un fichier XML :

<https://www.w3schools.com/xml/>

* Comment lire un fichier XML en Java :

<https://www.javatpoint.com/how-to-read-xml-file-in-java>

* Ajouter des données au fichier XML

<https://www.w3schools.com/xml/dom_nodes_add.asp>

* Supprimer des données XML en Java

<https://www.w3schools.com/xml/dom_nodes_remove.asp>

* Modifier le fichier XML en Java

<https://examples.javacodegeeks.com/core-java/xml/parsers/documentbuilderfactory/modify-xml-file-in-java-using-dom-parser-example/>