2020 - 2021

Projet

INFO0503

TONNELLE Nathan
L3 GROUPE S5O7A-2

Table des matières

Fichiers

Formats JSON

Voiture

Le fichier voiture.json est composé de la manière suivante :


```
{
    "moteur": " PuissDef : 50 id : 2 carburation : ELECTRIQUE puissance : 125",
    "dateFabrication": "Wed Dec 05 00:00:00 CET 3900",
    "modele": "nom : Citroën C3",
    "options": "camera de recul, vitres electriques, GPS,",
    "couleur": "BLEU",
    "numeroIdentification": "8V-5K4-E5"
}
```


Il possède différents attributs qui correspondent aux attributs d'un objet Voiture. Cependant je n'ai pas réussi à faire fonctionner cette fonction avec plusieurs voitures enregistrées, il aurait fallu que les voitures soient enregistrées sous forme de tableaux, comme sont enregistré les Options, Moteurs, et Modeles. Nous pouvons également voir que les attributs moteur et modele sont aussi sous forme clé-valeur, ce qui permet de créer ces objets dans la récupération de la voiture, car une voiture est composée d'un Moteur et d'un Modele qui sont des objets.

Options

Le fichier options.json est composé comme ceci :


```
[
    {"Option":"camera de recul","contient ":true},
    {"Option":"vitres electriques","contient ":true},
    {"Option":"toit ouvrant","contient ":true},
    {"Option":"GPS","contient ":true}
]
```


Contrairement à voiture.json, il est composé sous forme de tableau, ce qui permet de récupérer chaque valeurs une par une, il contient un nom d'option ainsi que s'il est activé ou non.

Moteur

Le fichier moteur.json est composé de la manière suivante :


```
[
    {"PuissDef":50,"Id ":1,"Carburation ":"essence","Puissance ":195},
    {"PuissDef":50,"Id ":2,"Carburation ":"electrique","Puissance ":125},
    {"PuissDef":50,"Id ":3,"Carburation ":"diesel","Puissance ":80},
    {"PuissDef":50,"Id ":4,"Carburation ":"GPL","Puissance ":62},
    {"PuissDef":50,"Id ":5,"Carburation ":"Hybride","Puissance ":100}
]
```


Nous pouvons voir que comme pour les options, le fichier est sous forme de tableau, ce qui permet de mettre autant de moteurs que l'on souhaite. Un objet Moteur est composé de 4 attributs, une puissance par défaut, un identifiant, un type de carburation ainsi que d'une puissance.

Modele

Le fichier modele.json est quant à lui très simple, il contient un tableau de Modele qui sont composé seulement d'un nom.

```
[
    {"nom":"Tesla Model S"},
    {"nom":"Renauld Megane 2019"},
    {"nom":"Ford Festa"},
    {"nom":"Citroën Berlingo"},
    {"nom":"Citroën C3"}
]
```

Metier - Login / Login_save

Ces 3 fichiers JSON sont liés (metier.json, login.json, login_save.json) ils possèdent la même forme et sont utiliser pour la gestion des utilisateurs.

```
{
    "concessionnaire":"hérard",
    "Directeur general d'usine":"rabat",
    "Directeur general d'usine":"tonnelle"
}
```

Dans ce fichier, nous retrouvons un ensemble clé-valeurs qui permettent d'assigner un métier à un mot de passe utilisateur. Ce qui permet de comparer les valeurs hashées des mots de passes avec ceux contenus dans login.json, pour savoir quelle métier est à quelle personne :

```
{
    "joffrey":"e4876b072a083701f3485ec3802c6cff10654d38e563fb40b244d6a14239bb3c",
    "cyril":"f5a8e4f4934888dfd59540d7848eeaa415d29a33a7aa7fef1e353fd52a276532",
    "nathan":"61c69429e35eda496cd688c8bde512578ee74c015a37b5db1bd14cb7706a4073"
}
```

Le fichier login.json, au-dessus, et login_save.json, en-dessous, sont liés car la partie valeur du login.json correspondent aux mots de passes (valeurs de login_save.json) mais hashées, ce qui permet de ne pas avoir les mots de passes en clair dans le programme. Cela est en partie vrai car pour le bien du projet nous avons fait le fichier login_save.json pour savoir le mot de passe (valeur) de chacun des pseudonymes (clé). Mais en temps normal ce fichier n'existerait pas. Les mots de passes sont hashées via une fonction dans GestionnaireUtilisateur.java et via la fonction de hashage sha-256.

```
{
    "joffrey":"hérard",
    "cyril":"rabat",
    "nathan":"tonnelle"
}
```

Couleurs

Le fichier couleurs.json est le plus simple de tous, il s'agit d'une liste de valeurs correspondant à des couleurs spécifiques.

Arborescence

Voici l'arborescence du projet, j'ai fait le choix d'une arborescence bien encadré où chaque type d'élément possède son dossier spécifique. Ainsi nous retrouvons dans :

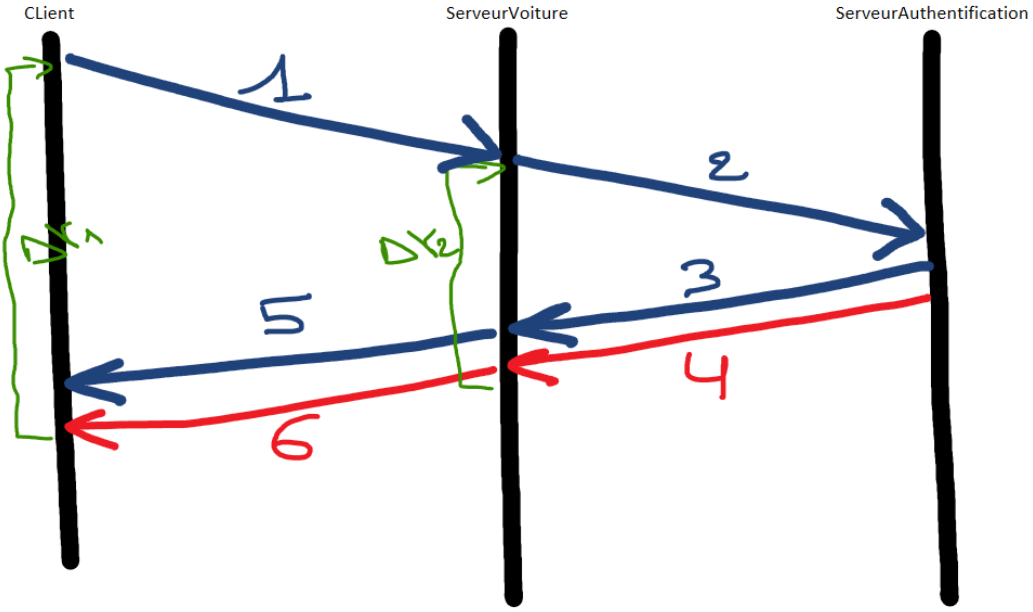
- Projet1/cls/ : les classes java compilées.
- Projet1/docs/ : la documentation du projet (non faite).
- Projet1/misc/ : les bibliothèques et fichiers de données.
- Projet1/misc/json/ : les fichiers de données JSON.
- Projet1/src/ : contient les codes des fichiers.
- Projet1/src/ java : les codes JAVA.
- Projet1/src/php : les codes PHP.

```
Projet1
  ├── cls
  ├── docs
  ├── misc
  │   └── json
  └── src
      ├── java
      └── php
```

Les opérations

Diagramme d'échange

Connexion



1 : Envoie du **login** (String) et **mot de passe** (String)
2 : Envoie des valeurs reçues au ServeurAuthentification (**login** (String) et **mot de passe** (String))
3-4 : retourne si les valeurs existent (3 = ok + retourne le la profession (String), 4 = ko + cas d'erreur)
5 : le retour est positif, renvoie vers la page d'accueil du Concessionnaire (pour la profession = Concessionnaire), vers la page d'accueil du gestionnaire d'Usines (pour la profession = Directeur general d'usine)
6 : le retour est négatif, retourne le cas d'erreur et renvoie vers le portail de connexion

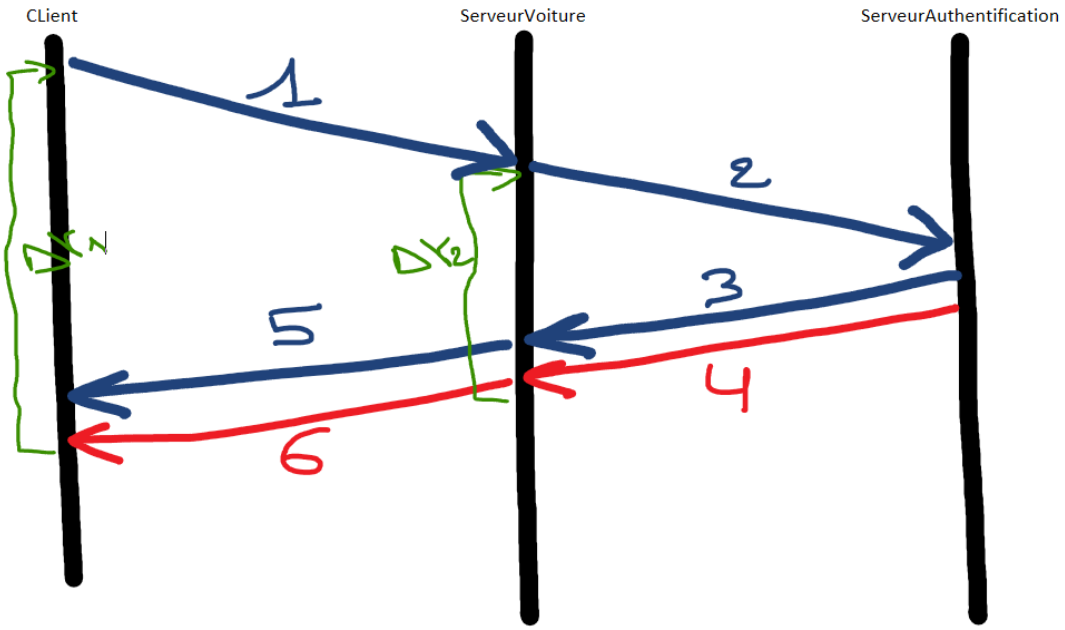
Cas d'erreur 4 :

- 1 : inconnu dans le JSON login
- 2 : lecture impossible du JSON login

Cas d'erreur 6 :

- 1 : non enregistré
- 2 : connexion au serveur impossible (timeout ou serveur non connecté)

Inscription



1 : Envoie du **login** (String), **mot de passe** (String), **profession** (Concessionnaire ou Directeur d'usine) (String)
2 : Envoie des valeurs reçues au ServeurAuthentification (**login** (String), **mot de passe** (String), **profession** (Concessionnaire ou Directeur d'usine) (String))
3-4 : retourne si les valeurs n'existent pas (3 = ok, 4 = ko + cas d'erreur)
5 : le retour est positif, renvoie vers la page d'accueil du Concessionnaire (pour la profession = Concessionnaire), vers la page d'accueil du gestionnaire d'Usines (pour la profession = Directeur general d'usine)
6 : le retour est négatif, retourne le cas d'erreur et renvoie vers le portail d'inscription

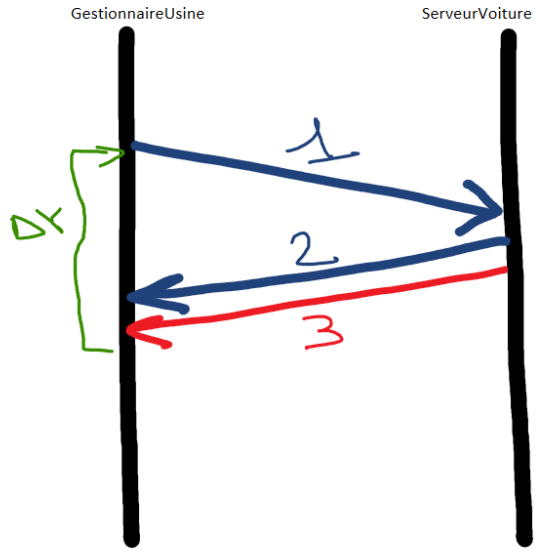
Cas d'erreur 4 :

- 1 : compte déjà existant
- 2 : lecture impossible du JSON login

Cas d'erreur 6 :

- 1 : compte déjà existant
- 2 : connexion au serveur impossible (timeout ou serveur non connecté)

Accès FormulaireAjoutVoiture

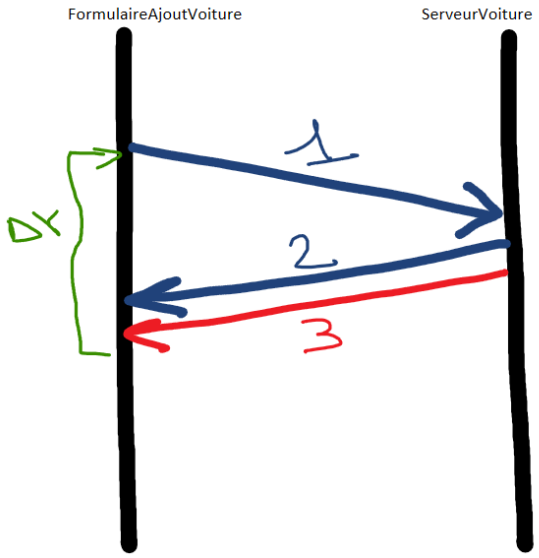


1 : Demande l'accès au formulaireAjoutVoiture
2-3 : retourne la page web php du formulaire si la profession = Directeur général D'usine (2 = ok, 3 = ko + cas d'erreur)

Cas d'erreur 3 :

- 1 : Connexion impossible au serveur
- 2 : le fichier php n'existe pas
- 3 : la profession est incorrecte

Envoie du FormulaireAjoutVoiture

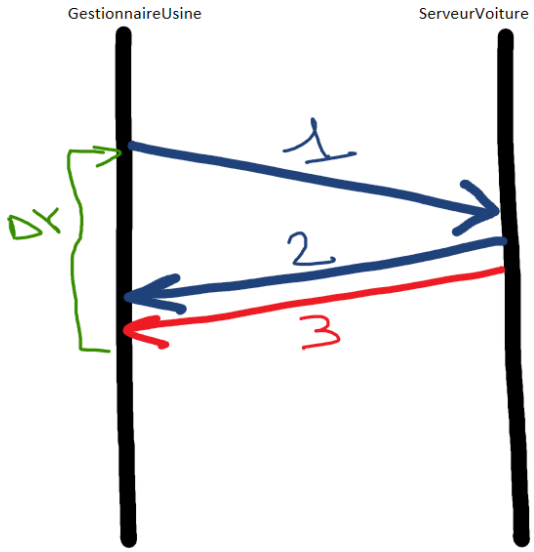


1 : Envoie des champs du formulaireAjoutVoiture
2-3 : retourne si la voiture a bien été créé (2 = ok, 3 = ko + cas d'erreur)

Cas d'erreur 3 :

- 1 : Connexion impossible au serveur
- 2 : la voiture ne peut pas être crée
- 3 : Ecriture de la voiture dans le JSON impossible

RechercherVoiture

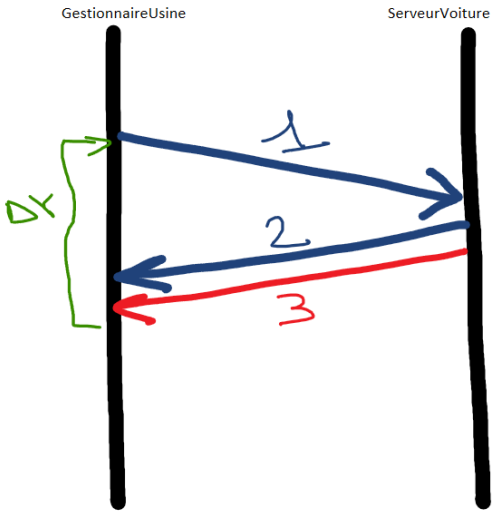


1 : Envoie l'objet Voiture au serveur
2-3 : retourne, si la voiture est dans le Json Voiture, (2 = ok, 3 = ko + cas d'erreur)

Cas d'erreur 3 :

- 1 : Connexion impossible au serveur
- 2 : la voiture n'existe pas
- 3 : Lecture du json impossible
- 4 : le fichier json n'existe pas

SupprimerVoiture

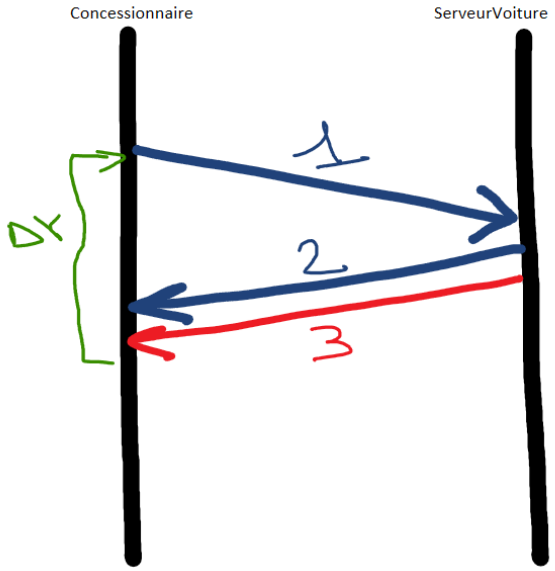


1 : Envoie l'objet Voiture au serveur
2-3 : retourne, si la voiture est dans le Json Voiture et est supprimée, (2 = ok, 3 = ko + cas d'erreur)

Cas d'erreur 3 :

- 1 : Connexion impossible au serveur
- 2 : la voiture n'existe pas
- 3 : Lecture du json impossible
- 4 : le fichier json n'existe pas

AfficherCatalogue

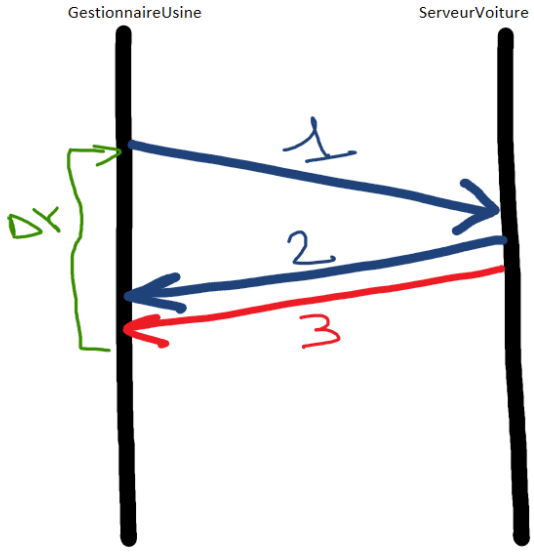


1 : Requête d'afficher le catalogue
2-3 : retourne, si le catalogue existe (2 = ok + catalogue, 3 = ko + cas d'erreur)

Cas d'erreur 3 :

- 1 : Connexion impossible au serveur
- 2 : le catalogue n'existe pas

AfficherMoteurs

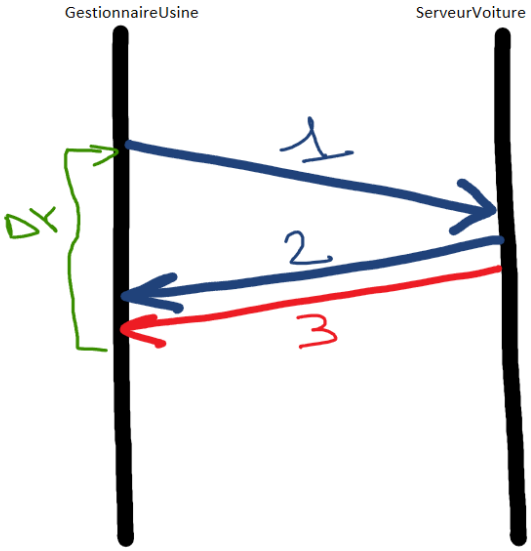


1 : Requête
2-3 : retourne, si le fichier Json des moteurs peut être lu, (2 = ok + liste Moteurs, 3 = ko + cas d'erreur)

Cas d'erreur 3 :

- 1 : Connexion impossible au serveur
- 2 : le fichier json n'existe pas
- 3 : Lecture du json impossible

AfficherModeles

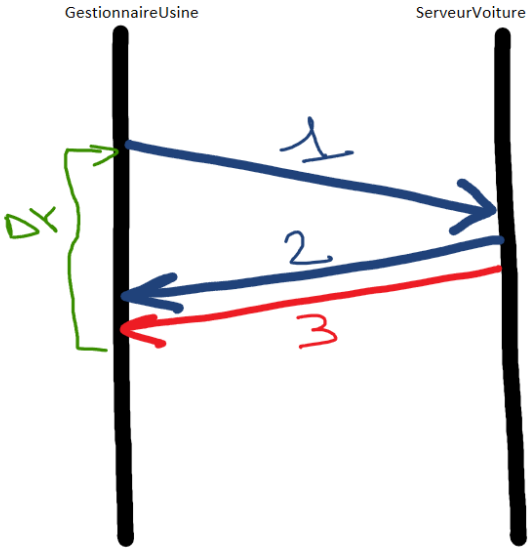


1 : Requête
2-3 : retourne, si le fichier Json des modeles peut être lu,
(2 = ok + liste Modeles, 3 = ko + cas d'erreur)

Cas d'erreur 3 :

- 1 : Connexion impossible au serveur
- 2 : le fichier json n'existe pas
- 3 : Lecture du json impossible

AfficherCouleurs



1 : Requête
2-3 : retourne, si le fichier Json des Couleurs peut être lu,
(2 = ok + liste couleurs, 3 = ko + cas d'erreur)

Cas d'erreur 3 :

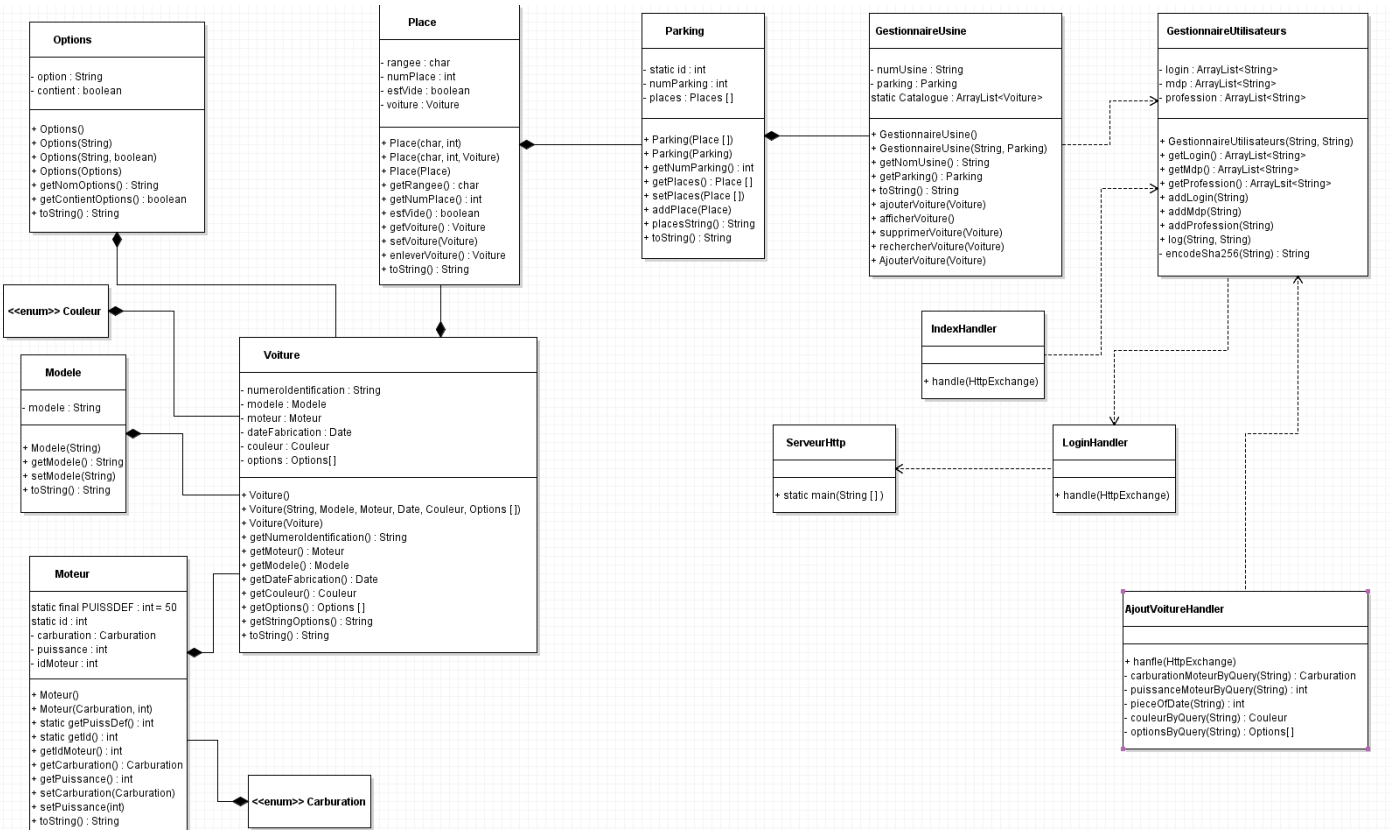
- 1 : Connexion impossible au serveur
- 2 : le fichier json n'existe pas
- 3 : Lecture du json impossible

Format des JSON échangés

Les échanges nécessitant un fichier JSON sont ceux où il doit y avoir une consultation des données, ou une écriture dans les fichiers JSON. Les fichiers JSON sont les mêmes fichiers d'écris dans la première partie.

Les diagrammes de classes

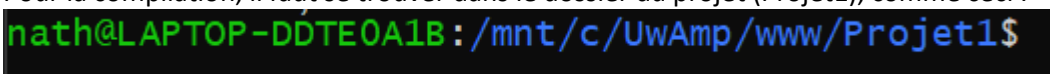
Voici le diagramme de classe pour ce projet, il se trouve également dans les fichiers du projet, en format HTML.



L'installation de l'application

Compilation

Pour la compilation, il faut se trouver dans le dossier du projet (Projet1), comme ceci :



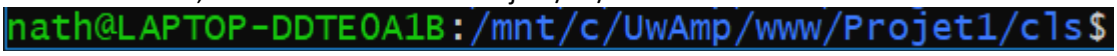
Ensuite exécuter la commande suivante : **javac -d ./cls/ -cp ./misc/json.jar -sourcepath ./src/java/ ./src/java/*.java**
Cette commande va compiler les classes java et les mettre dans le dossier **cls** du projet.

Configuration

Pour la configuration, rien de plus simple, mettre le dossier "Projet1" dans le dossier "www" de Uwamp.
Pour la configuration de Uwamp, ne pas mettre le port : 9876 car il est utilisé par le serveur.

Exécution

Pour l'exécution, il faut se rendre dans Projet1/cls/ :



Et exécuter la commande : **java -cp ../../misc/"json.jar:." ServeurHttp**