

# **RAPPORT PROJET SYSTEME**

## **DISTRUBIER**

- **KARMOUT Ismail**
- **M133145236**
- **MIAAD FS Meknès**

On souhaite créer un système distribué basé sur les micro-services Cette application devrait

permettre de gérer et d'automatiser le processus des infractions concernant des véhicules

suites à des dépassement de vitesses détectés par des radars automatiques. Le système se

compose de trois micro-services :

- Le micro-service qui permet de gérer les radars. Chaque radar est défini par son id, sa

vitesse maximale, des coordonnées : Longitude et Latitude.

- Le micro-service d'immatriculation qui permet de gérer des véhicules appartenant des

propriétaires. Chaque véhicule appartient à un seul propriétaire. Un propriétaire est

défini par son id, son nom, sa date de naissance, son email et son email. Un véhicule

est défini par son id, son numéro de matricule, sa marque, sa puissance fiscale et son

modèle.

- Le micro-service qui permet de gérer les infractions. Chaque infraction est définie par

son id, sa date, le numéro du radar qui a détecté le dépassement, le matricule du

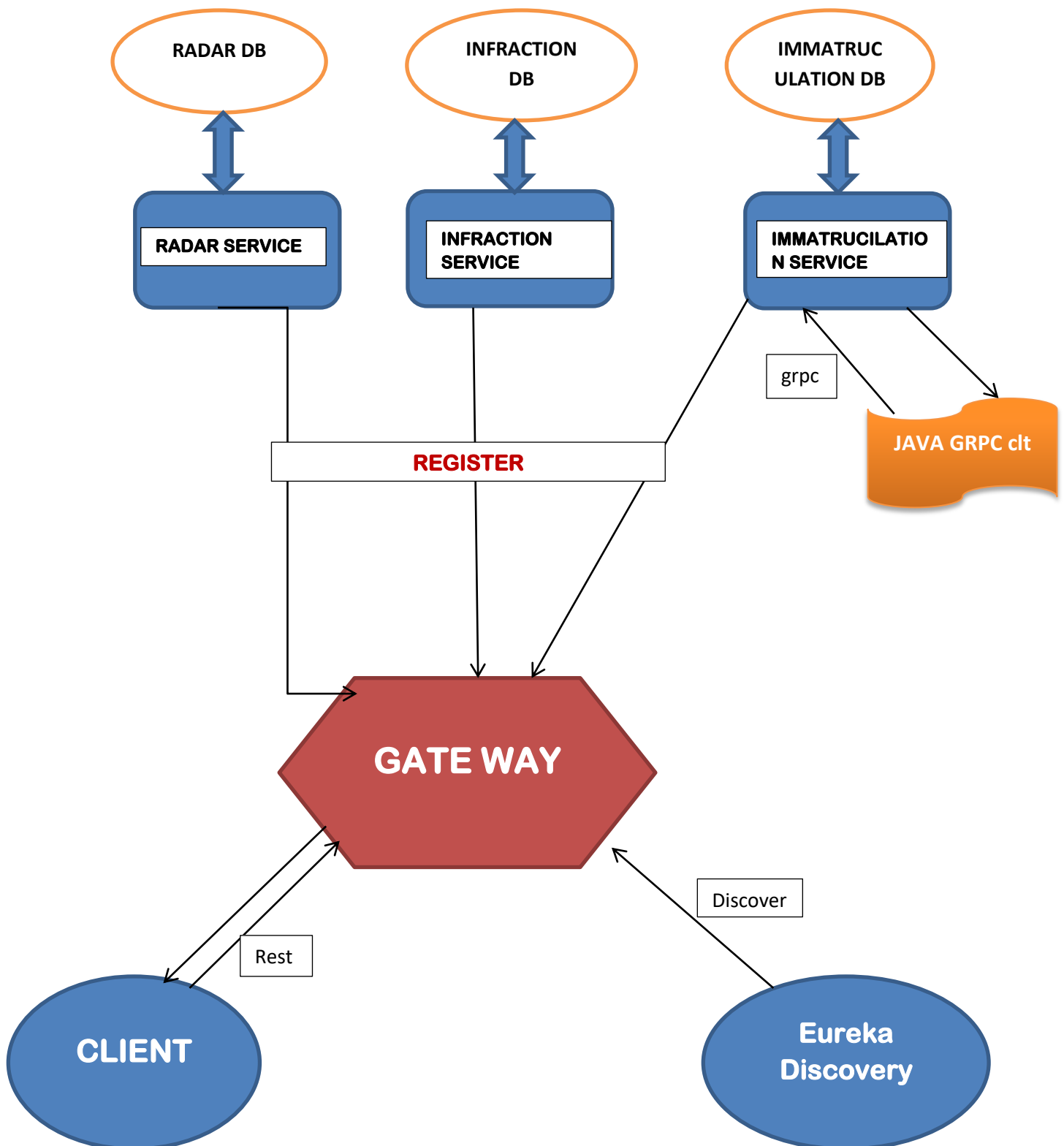
véhicule, la vitesse du véhicule, la vitesse maximale du radar et le montant de l'infraction.

En plus des opérations classiques de consultation et de modifications de données, le système

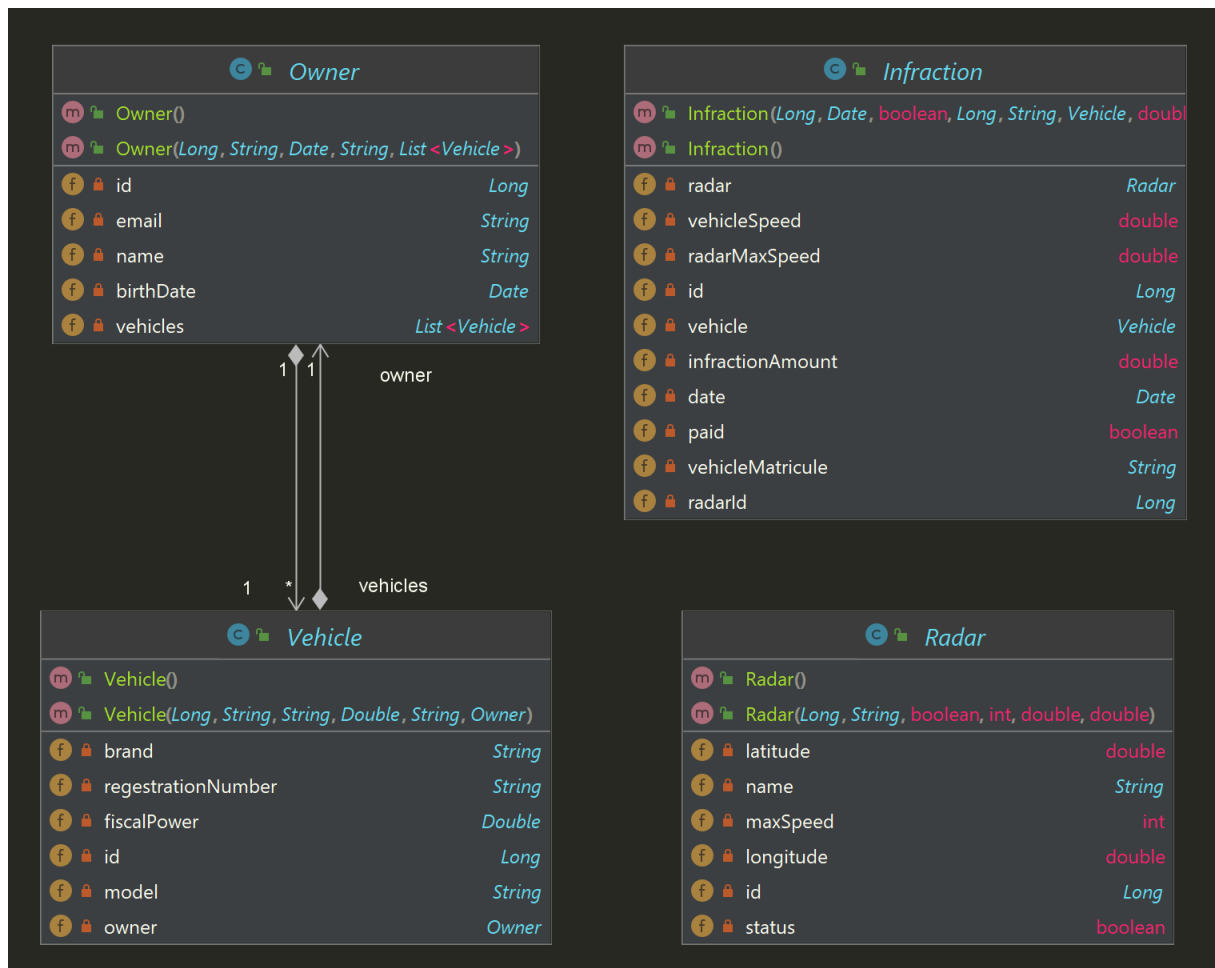
doit permettre de poster un dépassement de vitesse qui va se traduire par une infraction. En

plus, il doit permettre à un propriétaire de consulter ses infractions.

### 1- Architecture technique du projet



## 2- Class diagramme

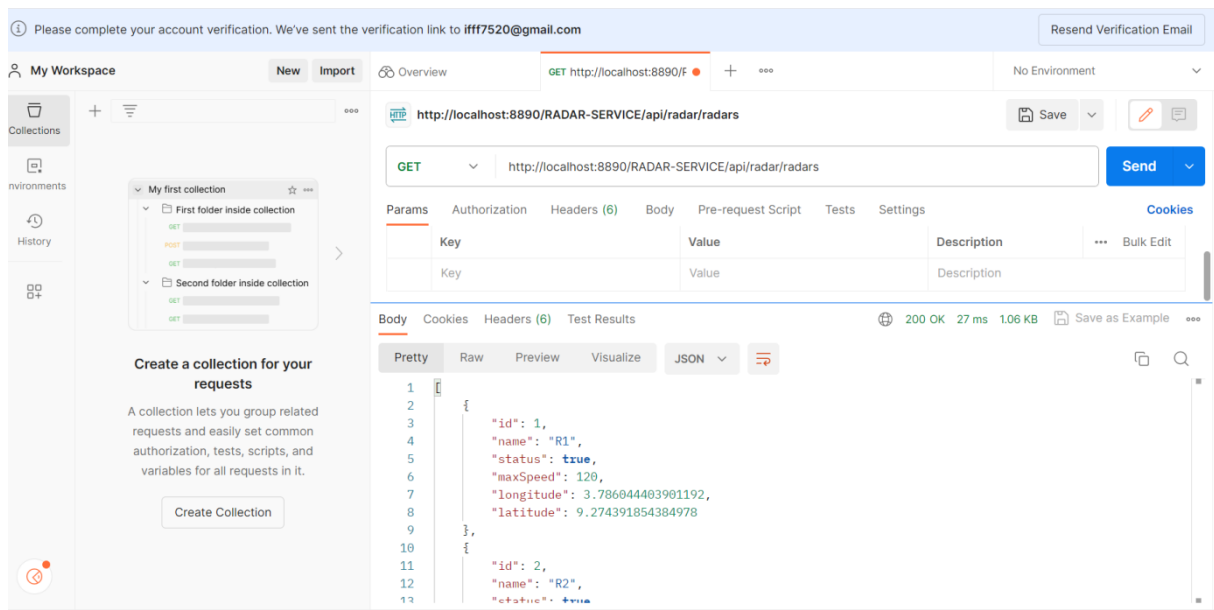


## 3-Tester les web services

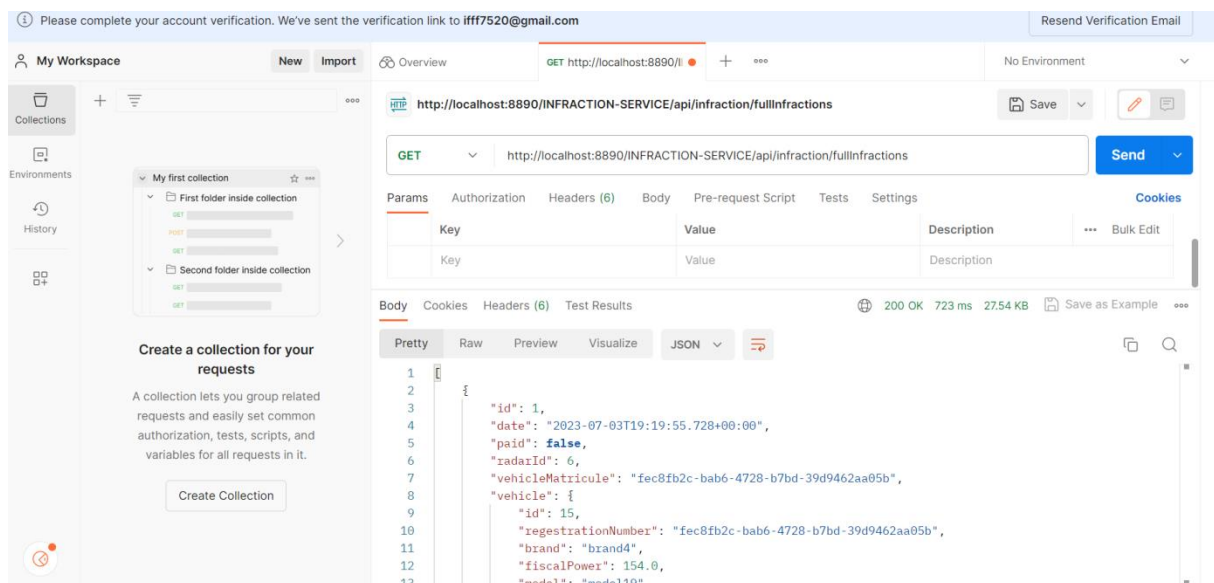
### 3-1 SOAP

### 3-2 REST

*Afficher les radars*



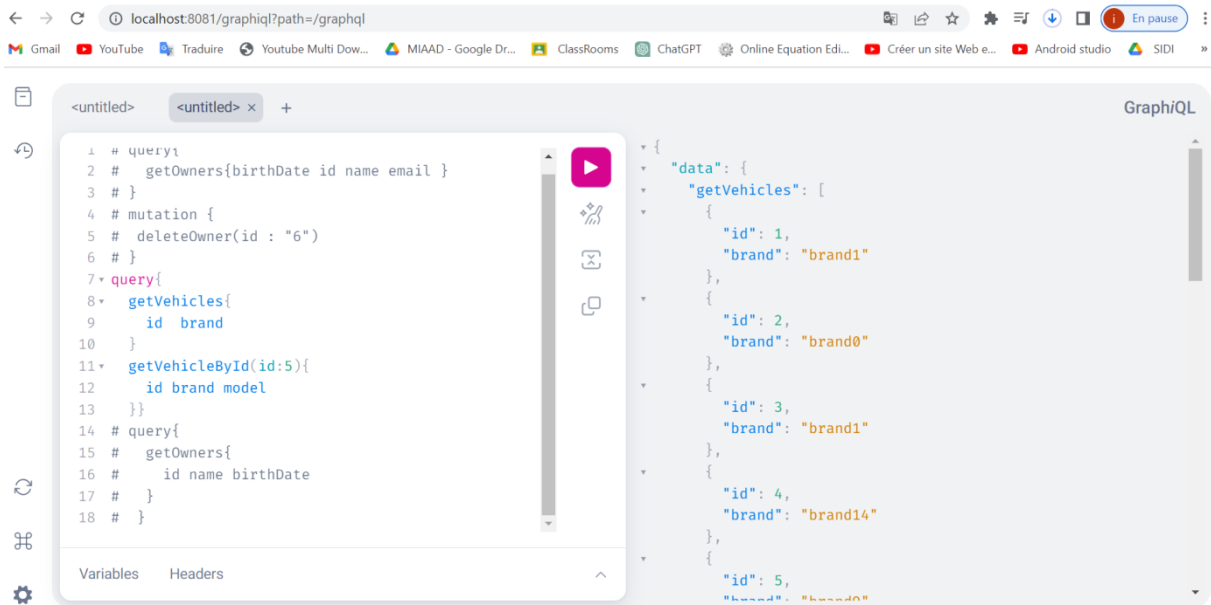
## Afficher tous les infractions



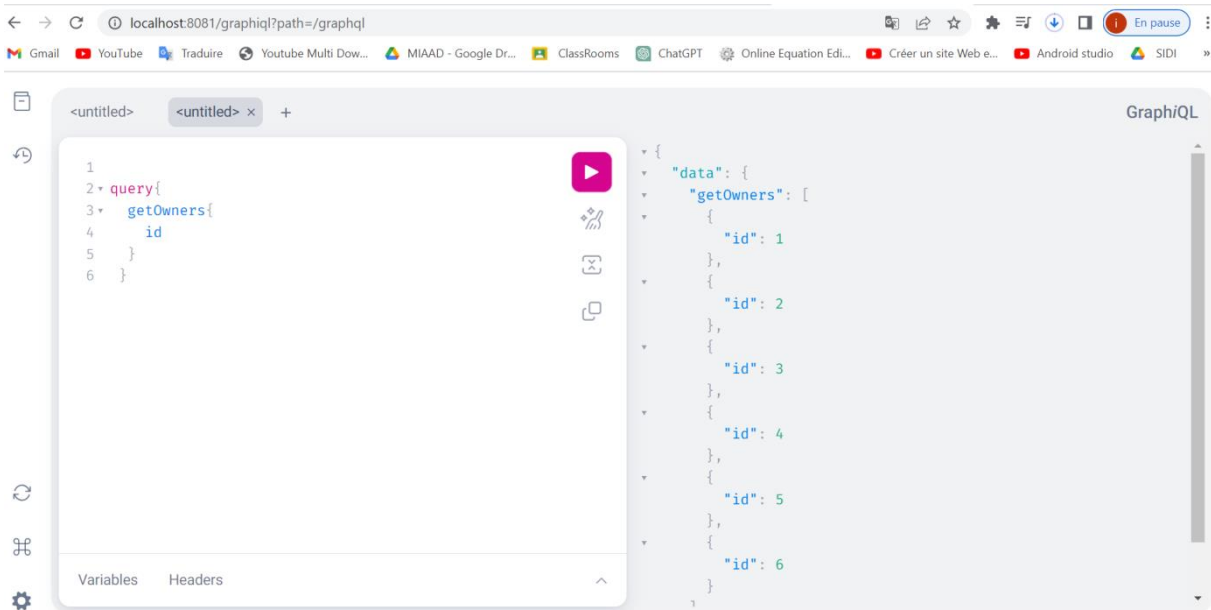
## 3-3 GraphQL

### Query

*Véhicule => id + brand*

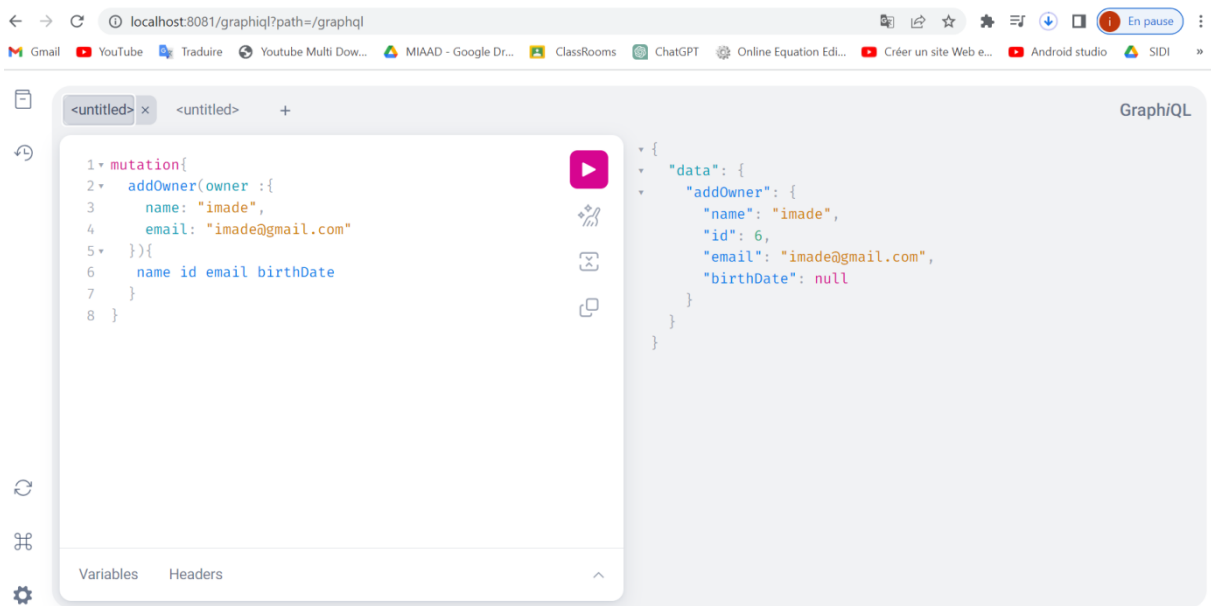


***Propriétaire => id***

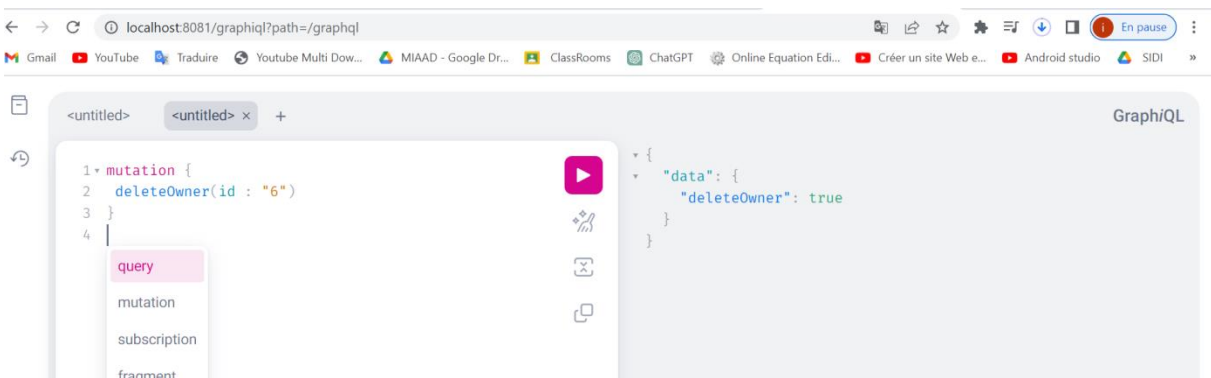


## Mutation

***Ajouté un propriétaire***

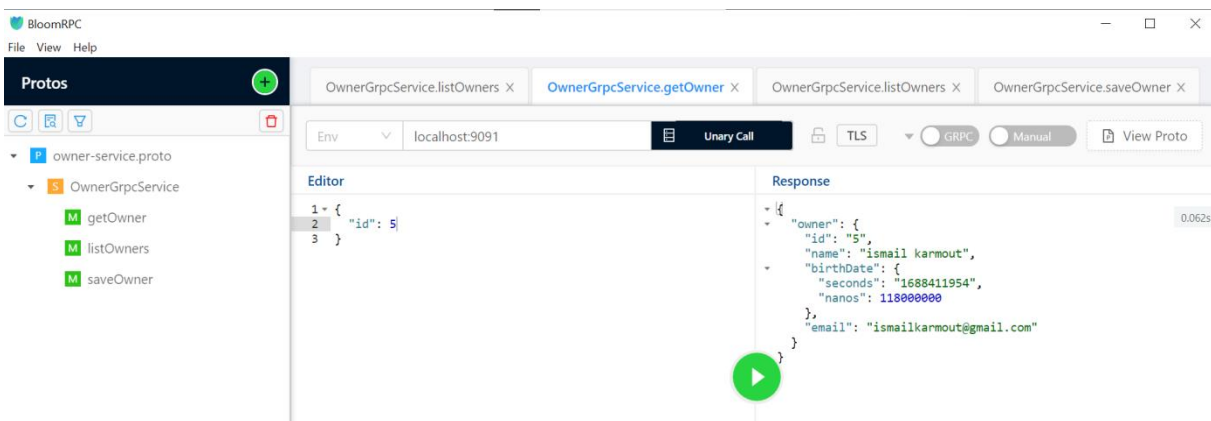


## Supprimer un propriétaire



## 3-4 GRPC

### Propriétaire par id



## 4-EURIKA & H2 data base

localhost:8761

DS Replicas

localhost

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
GATEWAY-SERVICE	n/a (1)	(1)	UP (1) - <a href="#">DESKTOP-BV12VQC:gateway-service:8890</a>
INFRACTION-SERVICE	n/a (1)	(1)	UP (1) - <a href="#">DESKTOP-BV12VQC:infraction-service:8082</a>
RADAR-SERVICE	n/a (1)	(1)	UP (1) - <a href="#">DESKTOP-BV12VQC:radar-service:8083</a>
REGISTRATION-SERVICE	n/a (1)	(1)	UP (1) - <a href="#">DESKTOP-BV12VQC:registration-service:8081</a>

General Info

## H2 data base INFRACTION

localhost:8082/h2-console/login.do?sessionId=da93119cca189022f2312bdf7ab75cb2

Auto commit: ☒ Max rows: 1000 Auto complete: ☐ Auto select: ☒

jdbc:h2:mem:infraction-db

INFRACTION

INFORMATION\_SCHEMA

Users

H2 2.1.214 (2022-06-13)

Run Run Selected Auto complete Clear SQL statement:

SELECT \* FROM INFRACTION

SELECT \* FROM INFRACTION:

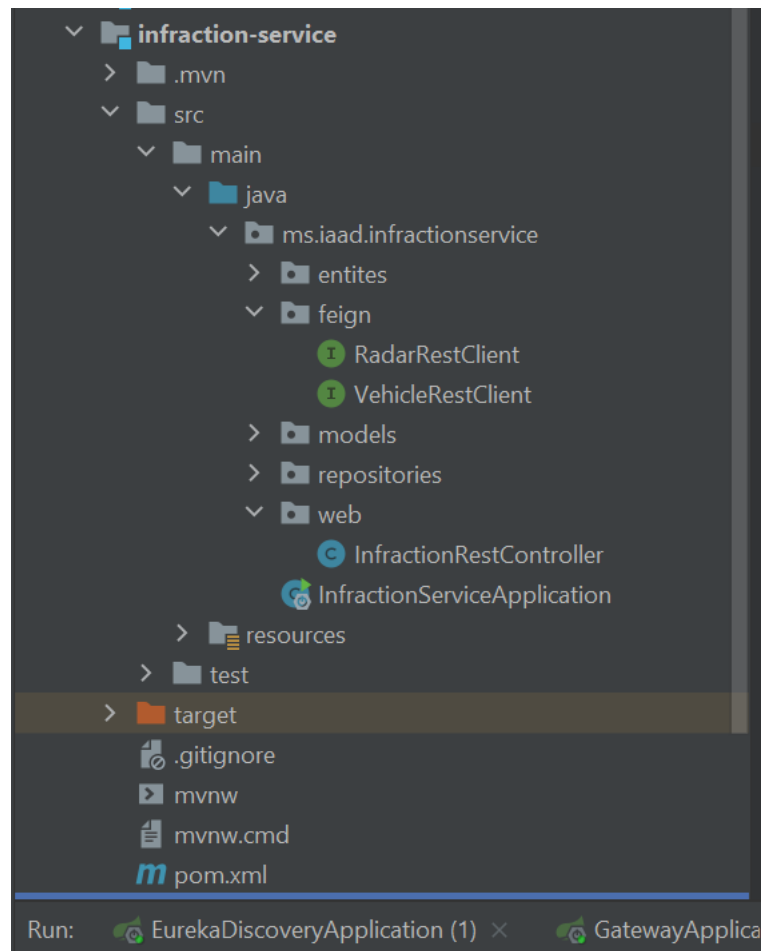
ID	DATE	INFRACTION_AMOUNT	PAID	RADAR_ID	RADAR_MAX_SPEED	VEHICLE_MATRICULE	VEHICLE_SPEED
1	2023-07-03 20:19:55.728	251.0	FALSE	6	120.0	fec8fb2c-bab6-4728-b7bd-39d9462aa05b	156.0
2	2023-07-03 20:19:59.462	361.0	FALSE	2	120.0	5e5dc620-d8be-4f45-a29b-7d538ec16784	187.0
3	2023-07-03 20:20:03.471	694.0	FALSE	6	120.0	bb72fa7c-7851-44a5-9815-4498449885fc	136.0
4	2023-07-03 20:20:07.468	316.0	FALSE	1	120.0	24bd951e-7bb6-4ca9-813a-547bce29cb96	163.0
5	2023-07-03 20:20:11.468	473.0	FALSE	4	120.0	5e5dc620-d8be-4f45-a29b-7d538ec16784	179.0
6	2023-07-03 20:20:15.44	34.0	FALSE	8	120.0	483e43cb-3487-4c84-96df-7cd9af7dea3e	214.0
7	2023-07-03 20:20:19.432	878.0	FALSE	8	120.0	c0df96c7b-2abe-4721-a4db-432b352ca355	162.0
8	2023-07-03 20:20:23.43	260.0	FALSE	7	120.0	c146f1ee-d896-4a48-ad5a-d812d7c4f686	143.0
9	2023-07-03 20:20:27.438	898.0	FALSE	8	120.0	483e43cb-3487-4c84-96df-7cd9af7dea3e	122.0
10	2023-07-03 20:20:31.502	498.0	FALSE	7	120.0	5e5dc620-d8be-4f45-a29b-7d538ec16784	213.0
11	2023-07-03 20:20:35.468	52.0	FALSE	6	120.0	24bd951e-7bb6-4ca9-813a-547bce29cb96	124.0
12	2023-07-03 20:20:39.445	457.0	FALSE	2	120.0	8e3acd03-6fb5-458d-bec5-6d13ad14cb83	204.0

## 5-BACKEND Services

### ✓ Infraction Service

*Le microservice d'infraction responsable de la gestion des violations traite chaque violation, qui est définie par son identifiant, sa date, le numéro radar ayant détecté l'infraction, le numéro d'immatriculation du véhicule, la vitesse du véhicule, la limite de vitesse maximale du radar et le montant de l'amende.*

- **Structure de service**



## ✓ Registration Service

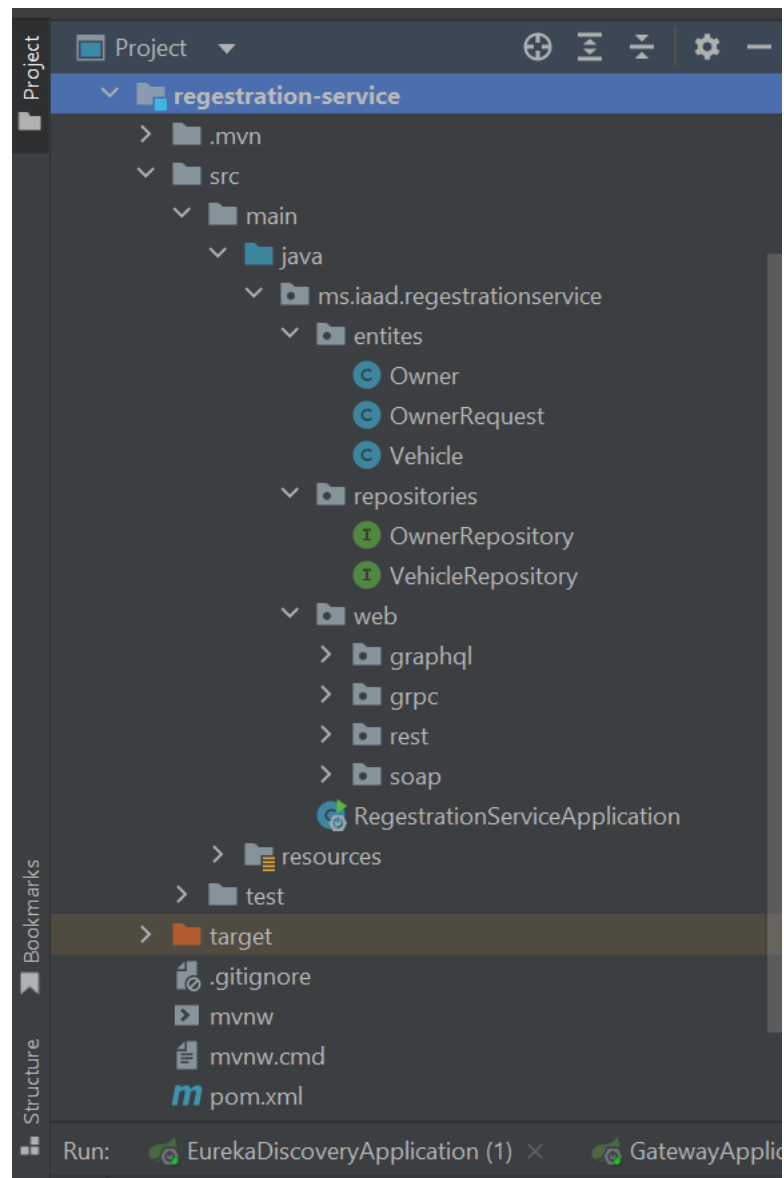
*Le microservice d'enregistrement gère les véhicules appartenant aux propriétaires. Chaque véhicule appartient à un seul propriétaire.*

*Un propriétaire est défini par son identifiant, son nom, sa date de naissance, son email.*

*Un véhicule est défini par son identifiant, son numéro d'immatriculation, sa marque, sa puissance fiscale et son modèle*

- **Structure de service**

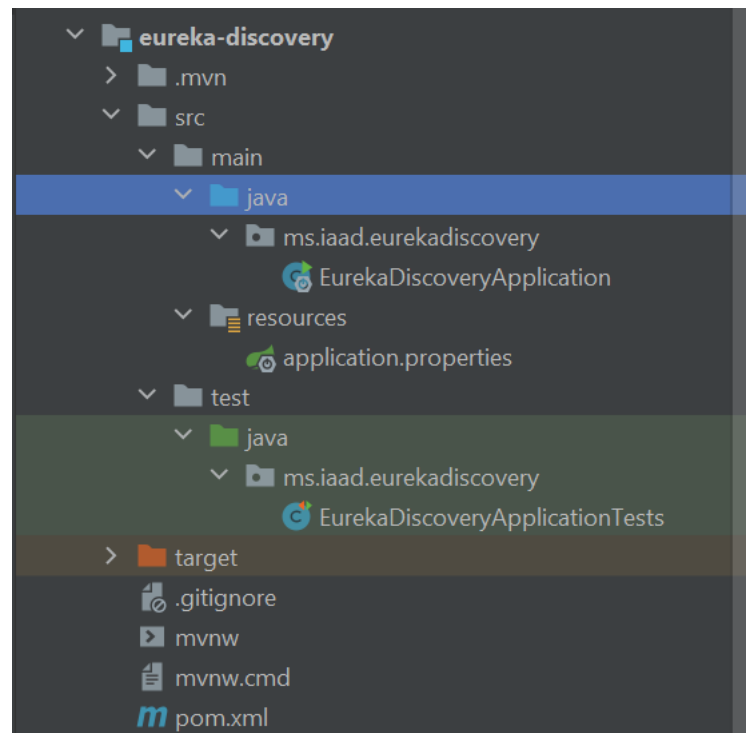




## ✓ Eureka Discovery Service

*composant côté serveur dans la pile OSS de Netflix qui permet aux services de s'enregistrer*

*et découvrir les uns les autres dans une architecture de microservices.*

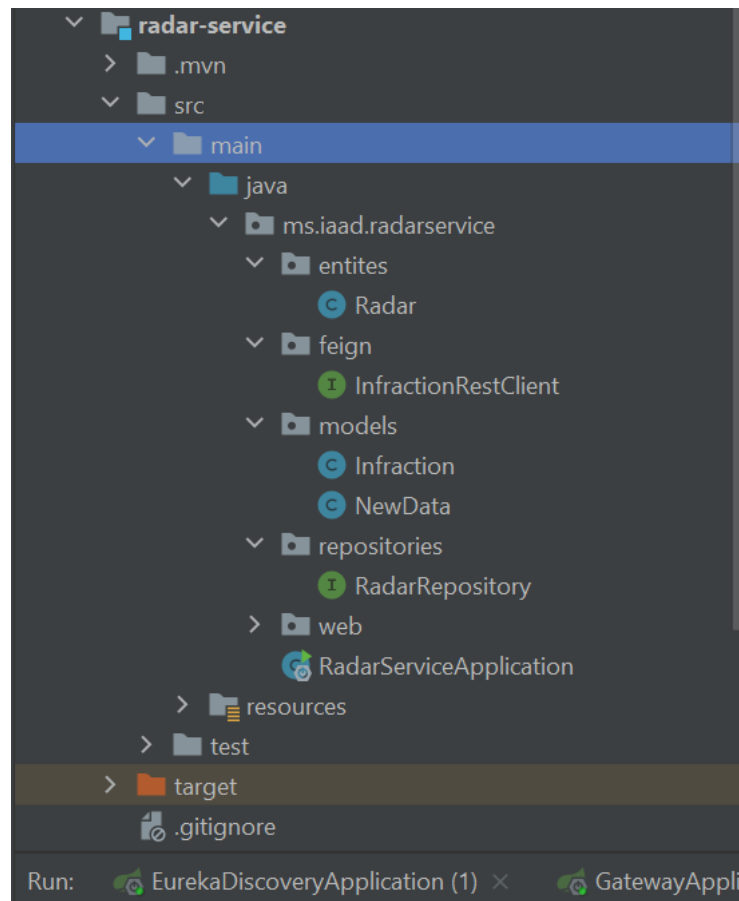


## ✓ Radar Service

*Le microservice radar responsable de la gestion des radars gère les entités radar définies*

*par leur identifiant, leur limite de vitesse maximale et leurs coordonnées (longitude et latitude)*

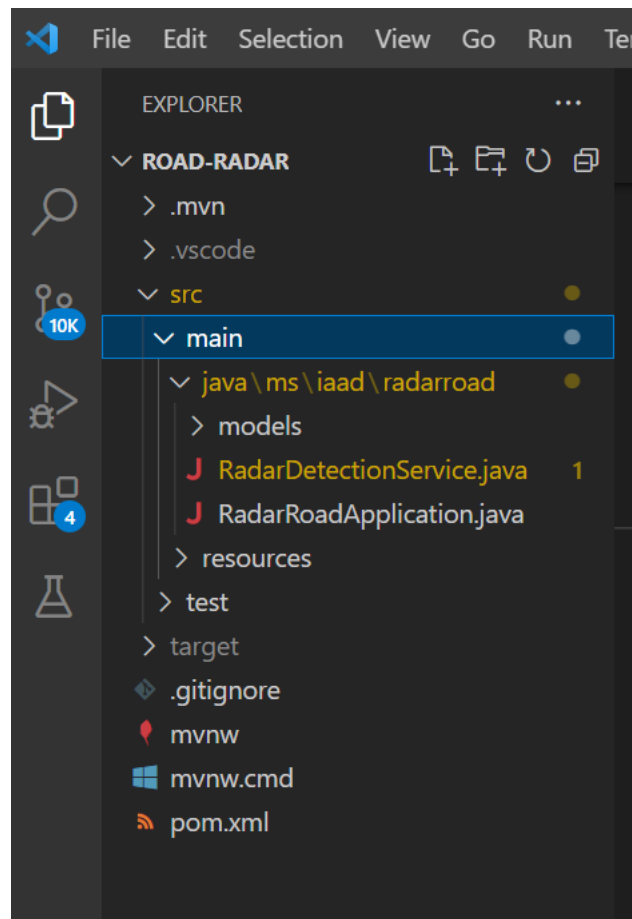
- **Structure de service**



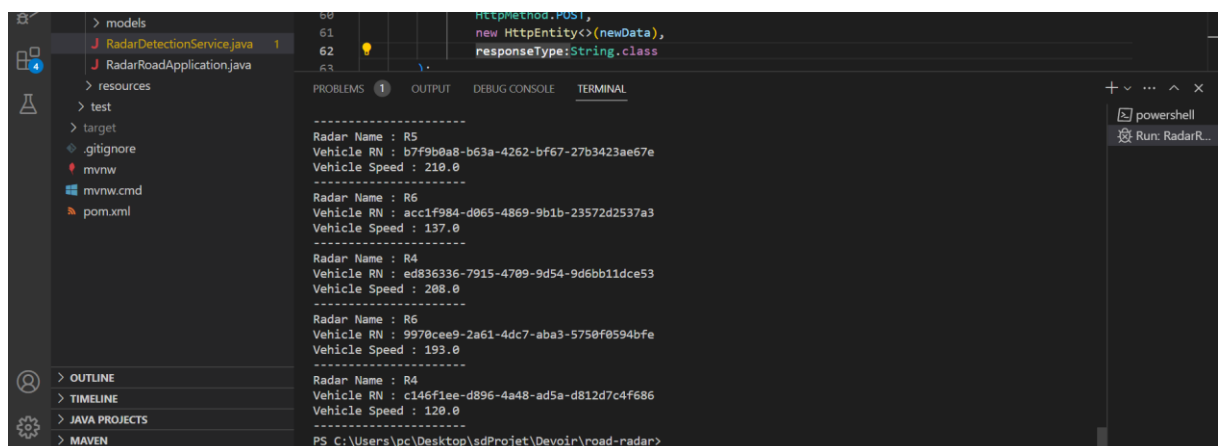
## ✓ Road Radar

*Une application Java qui simule un système radar générant des excès de vitesse aléatoires et de les envoyer au Radar-Service.*

- **Structure de service**



## • Test



## ✓ Gateway Service

*composant côté serveur dans la pile OSS de Netflix qui permet aux services de s'enregistrer et découvrir les uns les autres dans une architecture de microservices.*

- **Structure de service**

