ECOLE NORMALE SUPÉRIEURE DE L'ENSEIGNEMENT
TECHNIQUE DE MOHAMMEDIA
UNIVERSITÉ HASSAN II DE CASABLANCA

Département Mathématique Informatique

Filière : Master-SDIA

Modèle : Programmation Distribuées et Middlewares

Compte Rendu :

Contrôle et Projet Systèmes Distribués

Réaliser par :

- **Abdelkarim AGOUJIL**

Année Universitaire : 2022-2023

1. Introduction

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 :

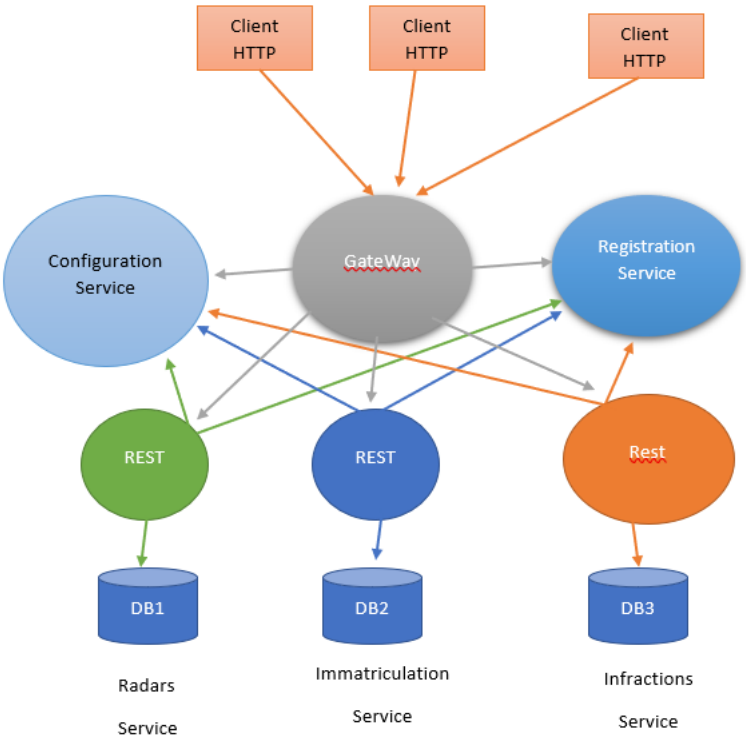
• Le micro-service qui permet de gérer les radars. Chaque radar est défini par son id, sa vitesse maximale, des coordonnées : Longitude et Latitude.

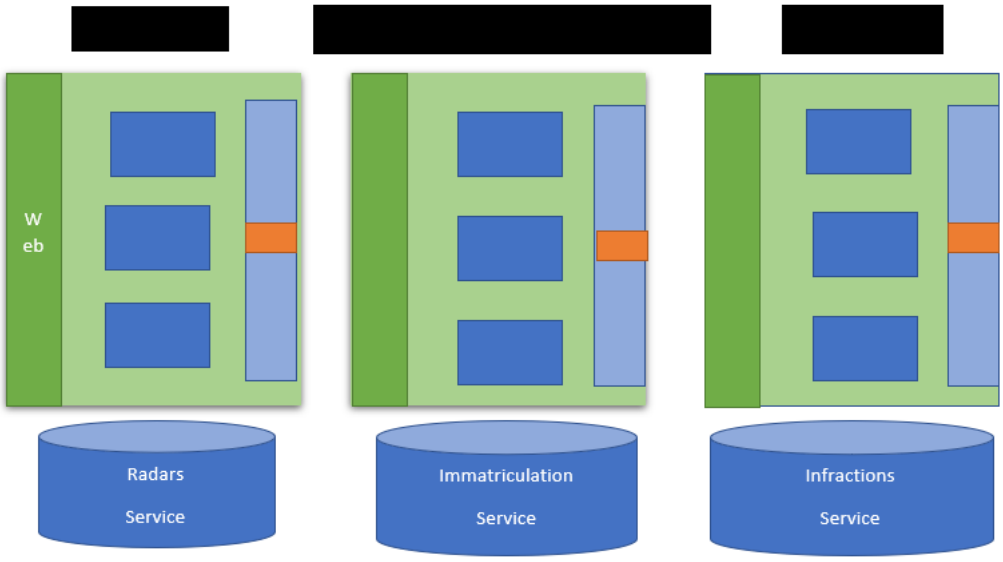
• Le micro-service d'immatriculation qui permet de gérer des véhicules appartenant des propriétaires. Chaque véhicule appartient à un seul propriétaire. Un propriétaire est défini par son id, son nom, sa date de naissance, son email et son email. Un véhicule est défini par son id, son numéro de matricule, sa marque, sa puissance fiscale et son modèle.

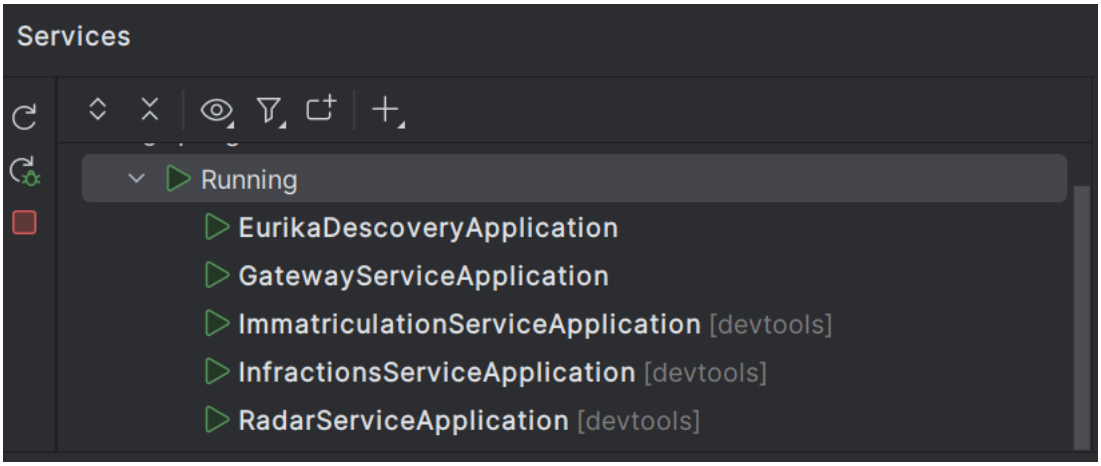
• Le micro-service qui permet de gérer les infractions. Chaque infraction est définie par son id, sa date, le numéro du radar qui a détecté le dépassement, le matricule du véhicule, la vitesse du véhicule, la vitesse maximale du radar et le montant de l'infraction.

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.

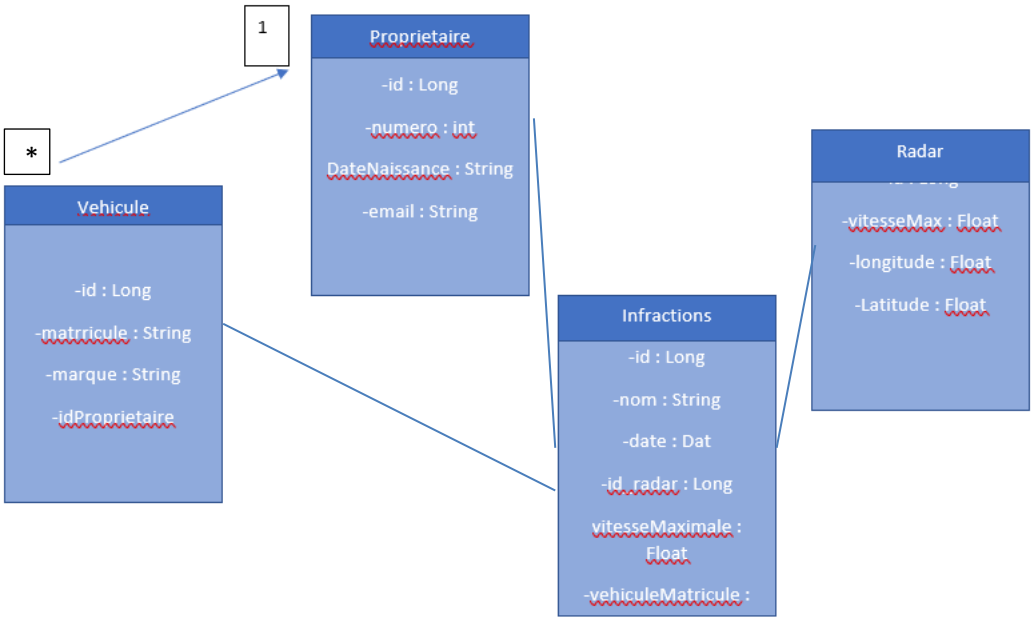
2. Architecture technique







3. Diagramme de classe



|Radar

- id: String |

- vitesseMaximale: int

- longitude: double

- latitude: double

+ getId(): String

+ getVitesseMaximale(): int

+ getLongitude(): double

+ getLatitude(): double

Proprietaire

- id: String

- nom: String

- dateNaissance: Date

- email: String

+ getId(): String

+ getNom(): String

+ getDateNaissance(): Date

+ getEmail(): String

Vehicule

- id: String

- numeroMatricule: String

- marque: String

- puissanceFiscale: int

- modele: String

- proprietaire: Proprietaire

+ getId(): String

+ getNumeroMatricule(): String

+ getMarque(): String

+ getPuissanceFiscale(): int

+ getModele(): String

+ getProprietaire(): Proprietaire

Infraction

- id: String

- date: Date

- radar: Radar

- vehicule: Vehicule

- vitesseVehicule: int

- vitesseMaximaleRadar: int

- montant: double

+ getId(): String

+ getDate(): Date

+ getRadar(): Radar

+ getVehicule(): Vehicule

+ getVitesseVehicule(): int

+ getVitesseMaximaleRadar(): int

+ getMontant(): double

---

3. Développer le micro-service Immatriculation :

a. Entités JPA et Interface JpaRepository basées sur Spring data

- **Entité Propriétaire**

```
package ma.enset.immatriculation_service.entities;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
```

```
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.ToString;

import java.util.List;


@Entity
@Data @AllArgsConstructor @NoArgsConstructor @Builber
public class Proprietaire {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private  Long id;
    private String nom;
    private String dateNaissance;
    private String email;
    @OneToMany(mappedBy = "proprietaire")

    @Transient
    private List<Vehicule> vehicules;
}
```

- **Entité Véhicule**

```
package ma.enset.immatriculation_service.entities;

import com.fasterxml.jackson.annotation.JsonProperty;
import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.ToString;

@Entity
@Data @AllArgsConstructor @NoArgsConstructor @Builder
public class Vehicule {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private  Long id;
    private String matrecule;
    private String marque;
    private String fiscalePuissance;
    private String model;
    @ManyToOne
    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private Proprietaire proprietaire;
}
```

- **Propriétaire Repository**

```
package ma.enset.immatriculation_service.repositories;


import ma.enset.immatriculation_service.entities.Proprietaire;
import org.springframework.data.jpa.repository.JpaRepository;
import
org.springframework.data.rest.core.annotation.RepositoryRestResource;


@RepositoryRestResource
public interface ProprietaireRepository extends
JpaRepository<Proprietaire,Long> {

Optional<Proprietaire> findByNom(String nom);

}
```

- **Véhicule Repository**

```
package ma.enset.immatriculation_service.repositories;


import ma.enset.immatriculation_service.entities.Vehicule;
import org.springframework.data.jpa.repository.JpaRepository;
import
org.springframework.data.rest.core.annotation.RepositoryRestResource;

import java.util.List;

@RepositoryRestResource
public interface VehiculeRepository extends
JpaRepository<Vehicule,Long> {

}
```

- **ImmatriculationServiceApplication**

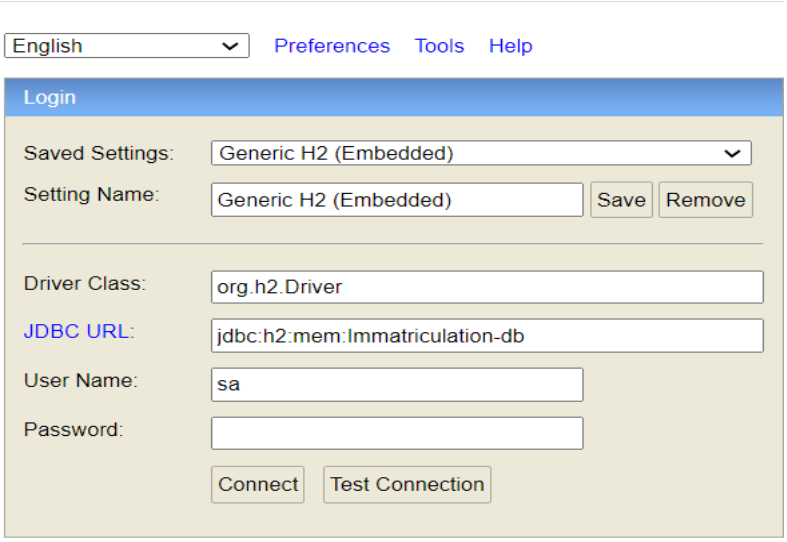
Pour tester les deux entités j'ai créé deux propriétaires et trois véhicule, j'ais associer les deux premiers véhicules et la troisième a le deuxième propriétaire :

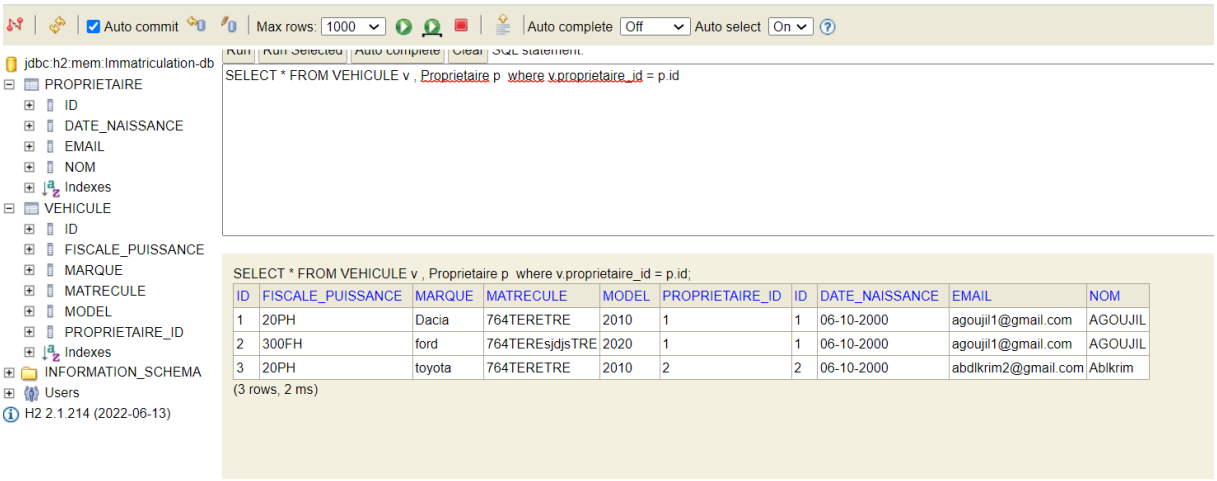
```
package ma.enset.immatriculation_service;

import ma.enset.immatriculation_service.entities.Proprietaire;
import ma.enset.immatriculation_service.entities.Vehicule;
import
ma.enset.immatriculation_service.repositories.ProprietaireRepository;
import ma.enset.immatriculation_service.repositories.VehiculeRepository;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import
org.springframework.data.rest.core.config.RepositoryRestConfiguration;

import java.util.ArrayList;

@SpringBootApplication
public class ImmatriculationServiceApplication {

    public static void main(String[] args) {
        SpringApplication.run(ImmatriculationServiceApplication.class,
args);
    }
    @Bean
    CommandLineRunner start(ProprietaireRepository proprietaireRepository,
VehiculeRepository vehiculeRepository, RepositoryRestConfiguration
repositoryRestConfiguration){
        repositoryRestConfiguration.exposeIdsFor(Proprietaire.class);
        repositoryRestConfiguration.exposeIdsFor(Vehicule.class);
        return args -> {
            Proprietaire agoujil = new Proprietaire(null, "AGOUJIL", "06-
10-2000", "agoujil1@gmail.com",null);
            Proprietaire ablkrim = new Proprietaire(null, "Ablkrim", "06-
10-2000", "abdlkrim2@gmail.com",null);
            proprietaireRepository.save(agoujil);
            proprietaireRepository.save(ablkrim);
            Vehicule vehicule1 = new
Vehicule(null,"764TERETRE","Dacia","20PH","2010",agoujil);
            Vehicule vehicule2 = new
Vehicule(null,"764TEREsjdjsTRE","ford","300FH","2020",agoujil);
```

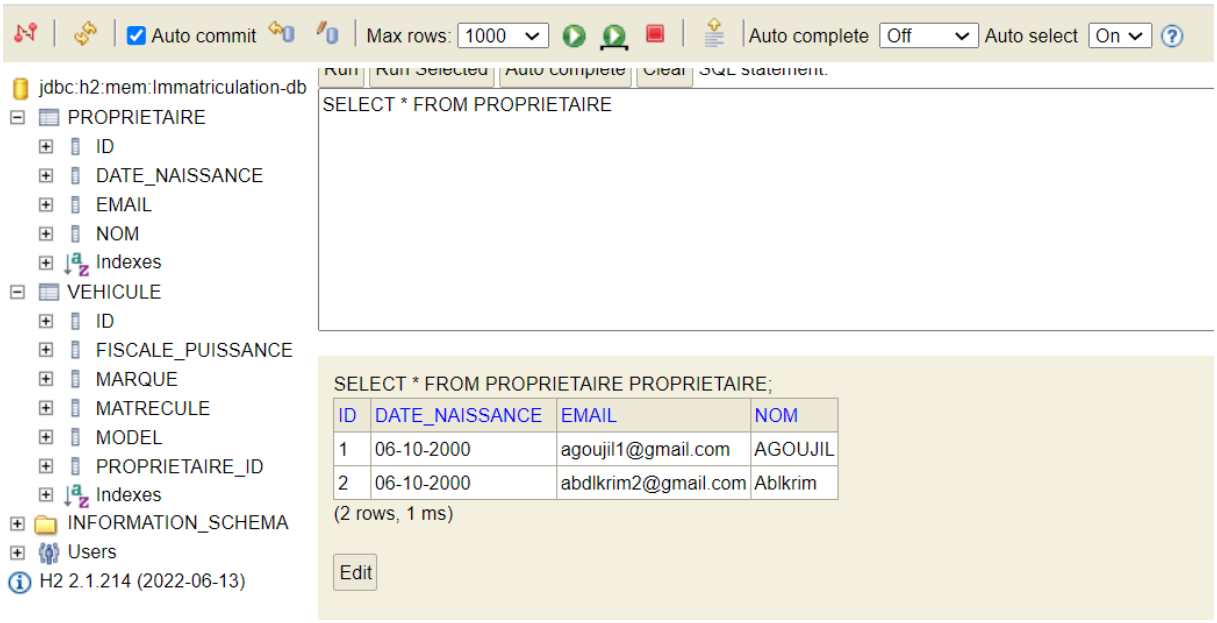
```
            vehiculeRepository.save(vehicule1);
            vehiculeRepository.save(vehicule2);
            vehiculeRepository.save(new
Vehicule(null,"764TERETRE","toyota","20PH","2010",ablkrim));

        };
    }

}
```

- **Teste l'application :**

Voire les résultats dans la base de données H2 console :







b. Les 4 web services REST, GraphQL, SOAP et GRPC

- **Web Service REST**
  - **ProprietaireRestController**

```
o  package ma.enset.immatriculation_service.web;

   import ma.enset.immatriculation_service.entities.Proprietaire;
   import
   ma.enset.immatriculation_service.repositories.ProprietaireRepos
   itory;
   import org.springframework.web.bind.annotation.*;

   import java.util.List;

   @RestController
   @RequestMapping("/api")
   public class ProprietaireRestController {
       private ProprietaireRepository proprietaireRepository;


       public ProprietaireRestController(ProprietaireRepository
   proprietaireRepository) {
           this.proprietaireRepository = proprietaireRepository;
       }
       @GetMapping("/proprietaires")
       public List<Proprietaire> proprietaires(){
           return proprietaireRepository.findAll();

       }
       @GetMapping("/proprietaires/{id}")
       public Proprietaire proprietaire(@PathVariable String id){
           return
   proprietaireRepository.findById(Long.valueOf(id))
                   .orElseThrow(()->new
   RuntimeException(String.format("Proprietaire % not
   found",id)));
       }
       @PostMapping("/proprietaires")
       public Proprietaire save(@RequestBody Proprietaire
   proprietaire){
           return proprietaireRepository.save(proprietaire);
```

```
        }
        @PutMapping("/proprietaires/{id}")
        public Proprietaire update(@PathVariable String
    id,@RequestBody Proprietaire proprietaire){
            Proprietaire
    proprietaire1=proprietaireRepository.findById(Long.valueOf(id))
    .orElseThrow();
            if(proprietaire.getNom()!=null)
    proprietaire1.setNom(proprietaire.getNom());
            if(proprietaire.getDateNaissance()!=null)
    proprietaire1.setDateNaissance
    (proprietaire.getDateNaissance());
            if(proprietaire.getEmail()!=null)
    proprietaire1.setEmail (proprietaire.getEmail());
            if(proprietaire.getVehicules()!=null)
    proprietaire1.setVehicules(proprietaire.getVehicules());
            return proprietaireRepository.save(proprietaire1);
        }
        @DeleteMapping("/proprietaires/{id}")
        public void delete(@PathVariable String id){
            proprietaireRepository.deleteById(Long.valueOf(id));
        }
    }
```

- **VéhiculeRestController**

```
package ma.enset.immatriculation_service.web;

import ma.enset.immatriculation_service.dto.VehiculeDto;
import ma.enset.immatriculation_service.entities.Proprietaire;
import ma.enset.immatriculation_service.entities.Vehicule;
import
ma.enset.immatriculation_service.repositories.ProprietaireRepos
itory;
import
ma.enset.immatriculation_service.repositories.VehiculeRepositor
y;
import org.springframework.web.bind.annotation.*;

import java.util.List;


@RestController
@RequestMapping("/api")
public class VehiculeRestController {
    private VehiculeRepository vehiculeRepository;
    private ProprietaireRepository proprietaireRepository;
    private VehiculeDto vehiculeDto;

    public VehiculeRestController(VehiculeRepository
vehiculeRepository, ProprietaireRepository
proprietaireRepository) {
        this.vehiculeRepository = vehiculeRepository;
        this.proprietaireRepository = proprietaireRepository;
    }

    @GetMapping("/vehicules")
    public List<Vehicule> vehicules(){
        return vehiculeRepository.findAll();

    }
    @GetMapping("/vehicules/{id}")
    public Vehicule vehicule(@PathVariable String id){
        return vehiculeRepository.findById(Long.valueOf(id))
```

```
                    .orElseThrow(()->new
RuntimeException(String.format("Vehicule % not found",id)));
    }
    @PostMapping("/vehicules")
    public Vehicule save(@RequestBody VehiculeDto vehiculeDto){
        Proprietaire
proprietaire=proprietaireRepository.findByNom(vehiculeDto.getNo
m()).get();
        Vehicule vehicule=new Vehicule().builder()
                .model(vehiculeDto.getModel())

.fiscalePuissance(vehiculeDto.getFiscalePuissance())
                .proprietaire(proprietaire)
                .matrecule(vehiculeDto.getMatrecule())
                .marque(vehiculeDto.getMarque())
                .build();
        return vehiculeRepository.save(vehicule);
    }
    @PutMapping("/vehicules/{id}")
    public Vehicule update(@PathVariable String id,@RequestBody
Vehicule vehicule){
        Vehicule
vehicule1=vehiculeRepository.findById(Long.valueOf(id)).orElseT
hrow();
        if(vehicule.getMarque()!=null)
vehicule1.setMarque(vehicule.getMarque());
        if(vehicule.getModel()!=null) vehicule1.setModel
(vehicule.getModel());
        if(vehicule.getMatrecule()!=null)
vehicule1.setMatrecule (vehicule.getMatrecule());
        if(vehicule.getProprietaire()!=null)
vehicule1.setProprietaire(vehicule.getProprietaire());
        if(vehicule.getFiscalePuissance()!=null)
vehicule1.setFiscalePuissance(vehicule.getFiscalePuissance());
        return vehiculeRepository.save(vehicule1);
    }
    @DeleteMapping("/vehicules/{id}")
    public void delete(@PathVariable String id){
        vehiculeRepository.deleteById(Long.valueOf(id));
    }
}
```

- **VehiculeDto**

Pour récupérer le propriétaire à partir du nom

```
package ma.enset.immatriculation_service.dto;

import lombok.Data;

@Data
public class VehiculeDto {
    String nom;
    private String matrecule;
    private String marque;
    private String fiscalePuissance;
    private String model;

}
```

- **Web Service SOAP**
    - **SoapWebServiceConfig :**

```
package ma.enset.immatriculation_service.web.soap;


import org.apache.cxf.Bus;
import org.apache.cxf.jaxws.EndpointImpl;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class CXFSoapWebServiceConfig {
    @Autowired
    private Bus bus;
    @Autowired
    private ImmatriculationSoapService
immatriculationSoapService;

    @Bean
    public EndpointImpl endpoint() {
        EndpointImpl endpoint = new EndpointImpl(bus,
immatriculationSoapService);
        endpoint.publish("/ImmatriculationService");
        return endpoint;
    }
}
```

- **ImmatriculationSoapService :**

```
package ma.enset.immatriculation_service.web.soap;

import jakarta.jws.WebMethod;
import jakarta.jws.WebParam;
import jakarta.jws.WebService;
import ma.enset.immatriculation_service.entities.Proprietaire;
import ma.enset.immatriculation_service.entities.Vehicule;
import
ma.enset.immatriculation_service.repositories.ProprietaireRepos
itory;
import
ma.enset.immatriculation_service.repositories.VehiculeRepositor
y;
import org.springframework.stereotype.Component;

import java.util.List;

@Component
@WebService(serviceName = "ImmatriculationWS")
//@AllArgsConstructor
public class ImmatriculationSoapService {
    private ProprietaireRepository proprietaireRepository;
    private VehiculeRepository vehiculeRepository;

    public ImmatriculationSoapService(ProprietaireRepository
proprietaireRepository, VehiculeRepository vehiculeRepository)
{
        this.proprietaireRepository = proprietaireRepository;
        this.vehiculeRepository = vehiculeRepository;
    }

    @WebMethod
    public List<Vehicule> vehicules(){
        return vehiculeRepository.findAll();
    }

    @WebMethod
    public Vehicule vehiculeById(@WebParam(name = "id") Long
```

```
    id){
            Vehicule vehicule =
    vehiculeRepository.findById(id).get();
            return vehicule;
        }

        @WebMethod
        public List<Proprietaire> proprietaires(){
            return proprietaireRepository.findAll();
        }

        @WebMethod
        public Proprietaire proprietaireById(@WebParam(name = "id")
    Long id){
            Proprietaire proprietaire =
    proprietaireRepository.findById(id).get();
            return proprietaire;
        }
    }
```

- **Web Service GRAPHQL**
  - **ProprietaireGraphQlService :**

```
o   package ma.enset.immatriculation_service.web.graphQL;

    import lombok.AllArgsConstructor;
    import ma.enset.immatriculation_service.entities.Proprietaire;
    import
    ma.enset.immatriculation_service.repositories.ProprietaireRepos
    itory;
    import
    org.springframework.graphql.data.method.annotation.Argument;
    import
    org.springframework.graphql.data.method.annotation.MutationMapp
    ing;
    import
    org.springframework.graphql.data.method.annotation.QueryMapping
    ;
    import org.springframework.stereotype.Controller;

    import java.util.List;

    @Controller
    @AllArgsConstructor
    public class ProprietaireGraphQlService {
        private ProprietaireRepository proprietaireRepository;

        @QueryMapping
        public List<Proprietaire> proprietaires(){
            return proprietaireRepository.findAll();
        }
        @QueryMapping
        public Proprietaire proprietaireRequestById(@Argument Long
    id){
            return proprietaireRepository.findById(id).get();
        }
        @MutationMapping
        public Proprietaire saveProprietaire(@Argument Proprietaire
    proprietaire){
            return proprietaireRepository.save(proprietaire);
        }
    }
```

- **VehiculeGraphQlService :**

```
package ma.enset.immatriculation_service.web.graphQL;

import lombok.AllArgsConstructor;
import ma.enset.immatriculation_service.entities.Vehicule;
import
ma.enset.immatriculation_service.repositories.VehiculeRepositor
y;
import
org.springframework.graphql.data.method.annotation.Argument;
import
org.springframework.graphql.data.method.annotation.MutationMapp
ing;
import
org.springframework.graphql.data.method.annotation.QueryMapping
;
import org.springframework.stereotype.Controller;

import java.util.List;

@Controller
@AllArgsConstructor
public class VehiculeGraphQlService {
    private VehiculeRepository vehiculeRepository;

    @QueryMapping
    public List<Vehicule> vehicules(){
        return vehiculeRepository.findAll();
    }
    @QueryMapping
    public Vehicule vehiculeRequestById(@Argument Long id){
        return vehiculeRepository.findById(id).get();
    }
    @MutationMapping
    public Vehicule saveVehicule(@Argument Vehicule vehicule){
        return vehiculeRepository.save(vehicule);
    }
}
```

- **schemat.graphqls :**

```
type Query {
    proprietaires:[Proprietaire],
    proprietaireRequestById(id:Int) : Proprietaire,
    vehicules:[Vehicule],
    vehiculeRequestById(id:Int) : Vehicule
}
type Mutation {

saveProprietaire(proprietaire:ProprietaireRequest):Proprietaire
,

updateProprietaire(id:Int,proprietaire:ProprietaireRequest):Pro
prietaire,
    deleteProprietaire(id:Int):String,
    saveVehicule(vehicule:VehiculeRequest):Vehicule,
    updateVehicule(id:Int,vehicule:VehiculeRequest):Vehicule,
    deleteVehicule(id:Int):String

}
type Proprietaire {
    id : Int,
    nom : String,
    dateNaissance : String,
    email : String
```

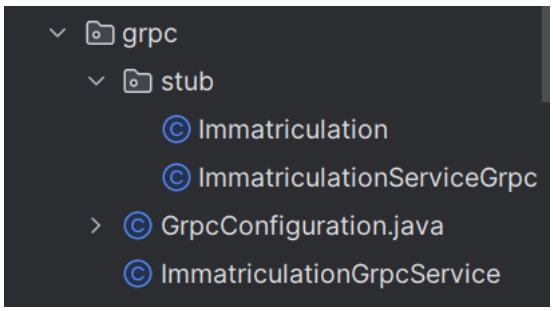
```
}
input ProprietaireRequest {
    nom : String,
    dateNaissance : String,
    email : String,
    #vehicules:[Vehicule]
}
type Vehicule{
    id:Int,
    matricule:String,
    model:String,
    marque:String,
    fiscalePuissance:String,
    proprietaire:Proprietaire
}
input VehiculeRequest{
    matricule:String,
    model:String,
    marque:String,
    fiscalePuissance:String,
    proprietaire:ProprietaireRequest
}
```

- **Web Service GRPC :**
  - Le fichier protobuff **immatriculation.proto**

```
o  syntax = "proto3";
   option
   java_package="ma.enset.immatriculation_service.web.grpc.stub";
   message Proprietaire {
     int64 id=1;
     string nom = 2;
     string dateNaissance = 3;
     string email = 4;
     repeated Vehicule vehicules=5;
   }
   message Vehicule {
     int64 id = 1;
     string matricule = 2;
     string marque = 3;
     string fiscalePuissance = 4;
     string modele = 5;
     Proprietaire proprietaire = 6;
   }

   service ImmatriculationService {
     rpc getProprietaire(GetProprietaireRequest) returns
   (GetProprietaireResponse);
     rpc getListProprietaire(GetAllProprietairesRequest) returns (
   GetAllProprietairesResponse);
     rpc saveProprietaire(SaveProprietaireRequest) returns
   (SaveProprietaireResponse);
     rpc getVehicule(GetVehiculeRequest) returns
   (GetVehiculeResponse);
     rpc getListVehicule(GetAllVehiculesRequest) returns (
   GetAllVehiculesResponse);
     rpc saveVehicule(SaveVehiculeRequest) returns
   (SaveVehiculeResponse);
   }
   message GetProprietaireRequest{
     int64 id=1;
   }
   message GetProprietaireResponse{
```

```
    Proprietaire proprietaire=1;
  }
  message GetAllProprietairesRequest {}
  message GetAllProprietairesResponse{
    repeated  Proprietaire proprietaire = 1;
  }
  message SaveProprietaireRequest {
    string nom = 1;
    string dateNaissance = 2;
    string email = 3;
    repeated Vehicule vehicule=4;
  }
  message SaveProprietaireResponse {
    Proprietaire proprietaire=1;
  }

  message GetVehiculeRequest{
    int64 id=1;
  }
  message GetVehiculeResponse{
    Vehicule vehicule=1;
  }
  message GetAllVehiculesRequest {}
  message GetAllVehiculesResponse{
    repeated  Vehicule vehicule = 1;
  }
  message SaveVehiculeRequest {
    string matricule = 1;
    string marque = 2;
    string fiscalePuissance = 3;
    string modele = 4;
    Proprietaire proprietaire = 5;
  }
  message SaveVehiculeResponse {
    Vehicule vehicule=1;
  }
```

- **stub :**



- **GrpcConfiguration.java :**

```
o  package ma.enset.immatriculation_service.web.grpc;

   import
   org.springframework.boot.autoconfigure.ImportAutoConfiguration;
   import org.springframework.context.annotation.Configuration;

   @Configuration
   @ImportAutoConfiguration({

   net.devh.boot.grpc.client.autoconfigure.GrpcClientAutoConfiguration.c
   lass,

   net.devh.boot.grpc.client.autoconfigure.GrpcClientMetricAutoConfigura
   tion.class,
```

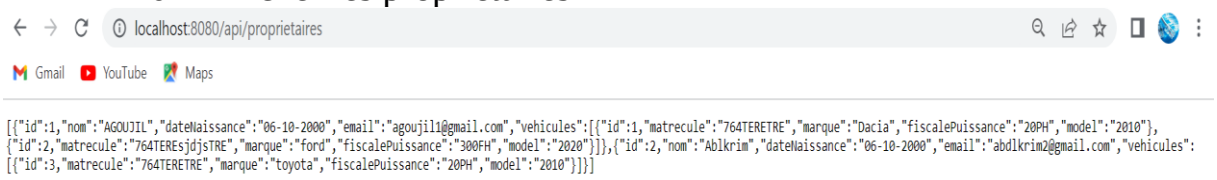
```
    net.devh.boot.grpc.client.autoconfigure.GrpcClientHealthAutoConfigura
    tion.class,

    net.devh.boot.grpc.client.autoconfigure.GrpcClientSecurityAutoConfigu
    ration.class,

    net.devh.boot.grpc.client.autoconfigure.GrpcClientTraceAutoConfigurat
    ion.class,

    net.devh.boot.grpc.client.autoconfigure.GrpcDiscoveryClientAutoConfig
    uration.class,


    net.devh.boot.grpc.common.autoconfigure.GrpcCommonCodecAutoConfigurat
    ion.class,

    net.devh.boot.grpc.common.autoconfigure.GrpcCommonTraceAutoConfigurat
    ion.class,


    net.devh.boot.grpc.server.autoconfigure.GrpcAdviceAutoConfiguration.c
    lass,

    net.devh.boot.grpc.server.autoconfigure.GrpcHealthServiceAutoConfigur
    ation.class,

    net.devh.boot.grpc.server.autoconfigure.GrpcMetadataConsulConfigurati
    on.class,

    net.devh.boot.grpc.server.autoconfigure.GrpcMetadataEurekaConfigurati
    on.class,

    net.devh.boot.grpc.server.autoconfigure.GrpcMetadataNacosConfiguratio
    n.class,

    net.devh.boot.grpc.server.autoconfigure.GrpcMetadataZookeeperConfigur
    ation.class,

    net.devh.boot.grpc.server.autoconfigure.GrpcReflectionServiceAutoConf
    iguration.class,

    net.devh.boot.grpc.server.autoconfigure.GrpcServerAutoConfiguration.c
    lass,

    net.devh.boot.grpc.server.autoconfigure.GrpcServerFactoryAutoConfigur
    ation.class,

    net.devh.boot.grpc.server.autoconfigure.GrpcServerMetricAutoConfigura
    tion.class,

    net.devh.boot.grpc.server.autoconfigure.GrpcServerSecurityAutoConfigu
    ration.class,

    net.devh.boot.grpc.server.autoconfigure.GrpcServerTraceAutoConfigurat
    ion.class
    })
    class GrpcConfig {}
```

- **ImmatriculationGrpcService :**

```
package ma.enset.immatriculation_service.web.grpc;
import io.grpc.stub.StreamObserver;
import lombok.AllArgsConstructor;
import ma.enset.immatriculation_service.entities.Proprietaire;
import ma.enset.immatriculation_service.entities.Vehicule;
import ma.enset.immatriculation_service.mappers.GrpcMapper;
import
ma.enset.immatriculation_service.repositories.ProprietaireRepository;
import ma.enset.immatriculation_service.repositories.VehiculeRepository;
import ma.enset.immatriculation_service.web.grpc.stub.Immatriculation;
import
ma.enset.immatriculation_service.web.grpc.stub.ImmatriculationServiceGrpc;
import net.devh.boot.grpc.server.service.GrpcService;

import java.util.List;
import java.util.stream.Collectors;
@GrpcService
@AllArgsConstructor
public class ImmatriculationGrpcService extends
ImmatriculationServiceGrpc.ImmatriculationServiceImplBase {
    private ProprietaireRepository proprietaireRepository;
    private VehiculeRepository vehiculeRepository;
    private GrpcMapper grpcMapper;

    @Override
    public void getListVehicule(Immatriculation.GetAllVehiculesRequest
request , StreamObserver<Immatriculation.GetAllVehiculesResponse>
responseObserver){
        List<Vehicule> vehicules= this.vehiculeRepository.findAll();
        List<Immatriculation.Vehicule>
vehiculeList=vehicules.stream().map(account-
>grpcMapper.fromVehiculeEntity(account)).collect(Collectors.toList());
        Immatriculation.GetAllVehiculesResponse
vehiculesListResponse=Immatriculation.GetAllVehiculesResponse.newBuilder()
                .addAllVehicule(vehiculeList)
                .build();
        responseObserver.onNext(vehiculesListResponse);
        responseObserver.onCompleted();
    }

    @Override
    public void
getListProprietaire(Immatriculation.GetAllProprietairesRequest request ,
StreamObserver<Immatriculation.GetAllProprietairesResponse>
responseObserver){
        List<Proprietaire> proprietaires=
this.proprietaireRepository.findAll();
        List<Immatriculation.Proprietaire>
proprietaires1=proprietaires.stream().map(grpcMapper::fromProprietaireEntit
y).collect(Collectors.toList());
        Immatriculation.GetAllProprietairesResponse
proprietaireResponse=Immatriculation.GetAllProprietairesResponse.newBuilder
()
                .addAllProprietaire(proprietaires1)
                .build();
        responseObserver.onNext(proprietaireResponse);
        responseObserver.onCompleted();
    }
}
```

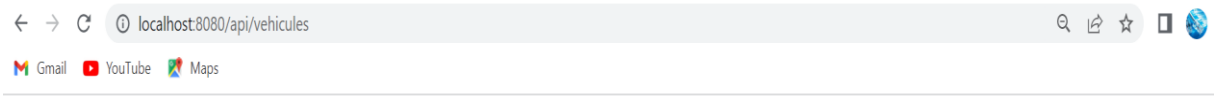
c. Tester les 4 web services

- **Tester REST**
  - Afficher les proprietaires



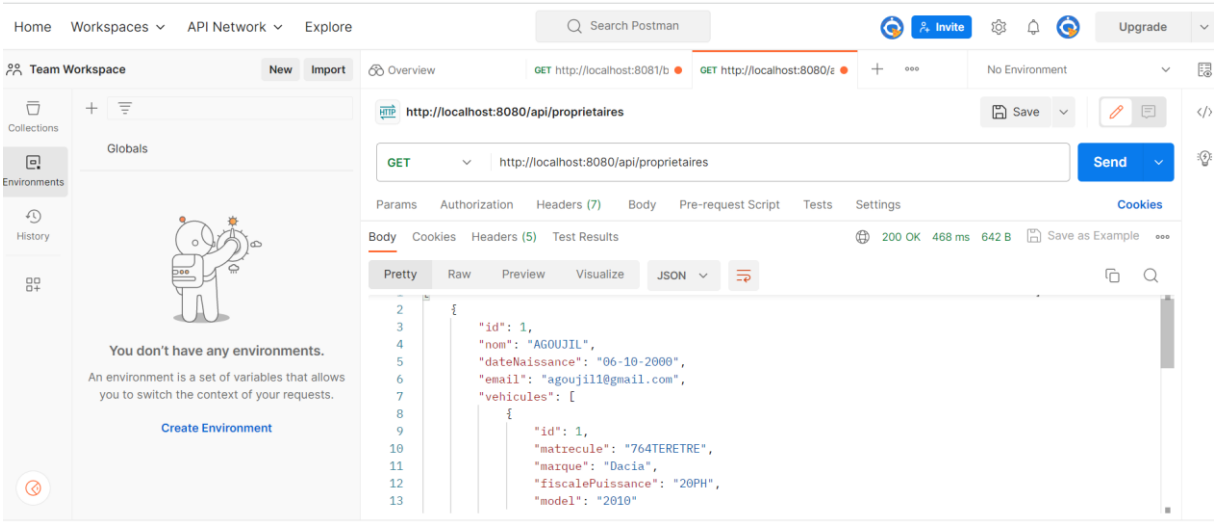

```
[{"id":1,"nom":"AGOUJIL","dateNaissance":"06-10-2000","email":"agoujil1@gmail.com","vehicules":[{"id":1,"matrecule":"764TERETRE","marque":"Dacia","fiscalePuissance":"20PH","model":"2010"},
{"id":2,"matrecule":"764TEREsjdjsTRE","marque":"ford","fiscalePuissance":"300FH","model":"2020"}]},{"id":2,"nom":"Ablkrim","dateNaissance":"06-10-2000","email":"abdlkrim2@gmail.com","vehicules":
[{"id":3,"matrecule":"764TERETRE","marque":"toyota","fiscalePuissance":"20PH","model":"2010"}]}]
```

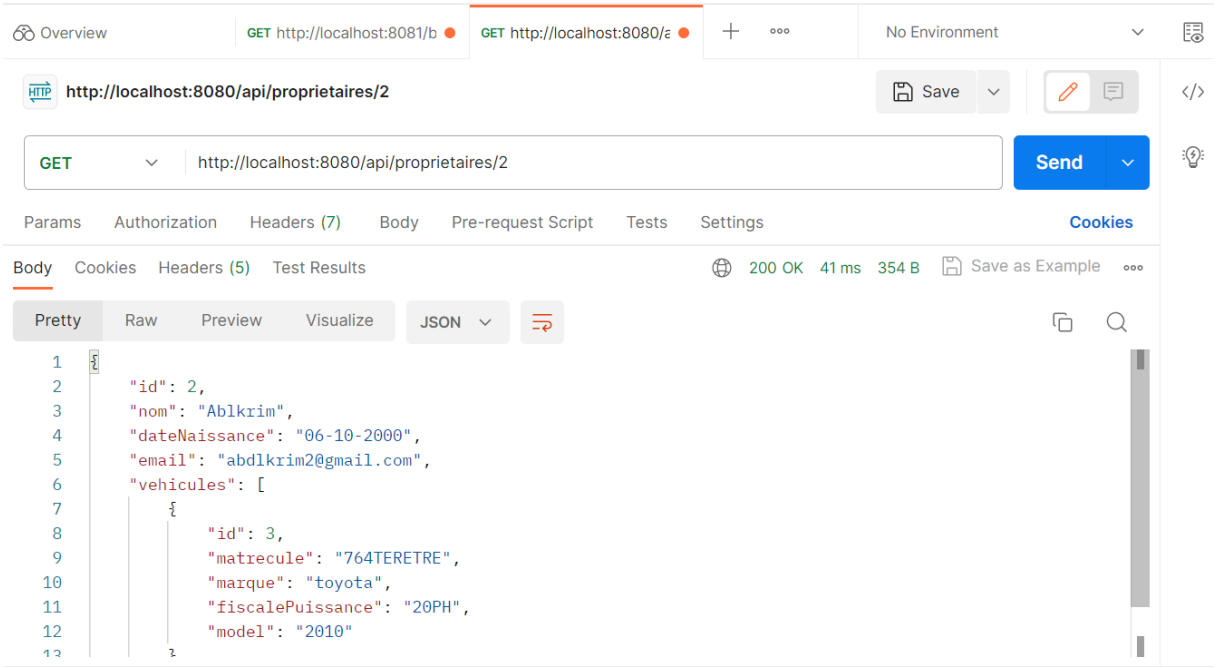

  - Afficher les vehicules



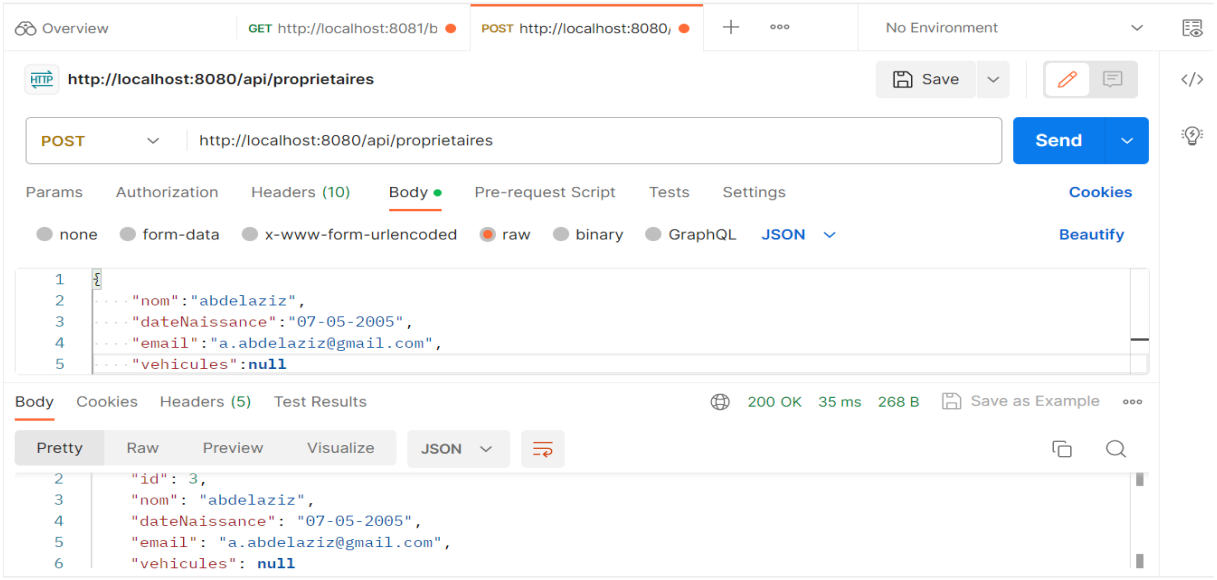

```
[{"id":1,"matrecule":"764TERETRE","marque":"Dacia","fiscalePuissance":"20PH","model":"2010"},{"id":2,"matrecule":"764TEREsjdjsTRE","marque":"ford","fiscalePuissance":"300FH","model":"2020"},
{"id":3,"matrecule":"764TERETRE","marque":"toyota","fiscalePuissance":"20PH","model":"2010"}]
```

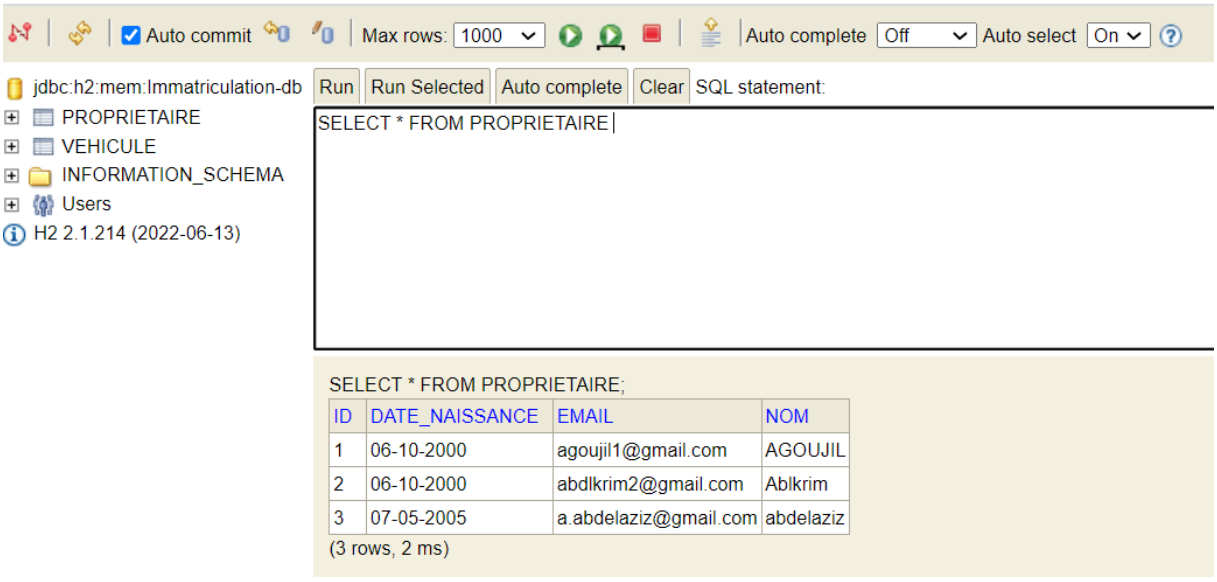

  - Tester les méthodes avec postman
  - Table Proprietaire :
    - Méthode GET



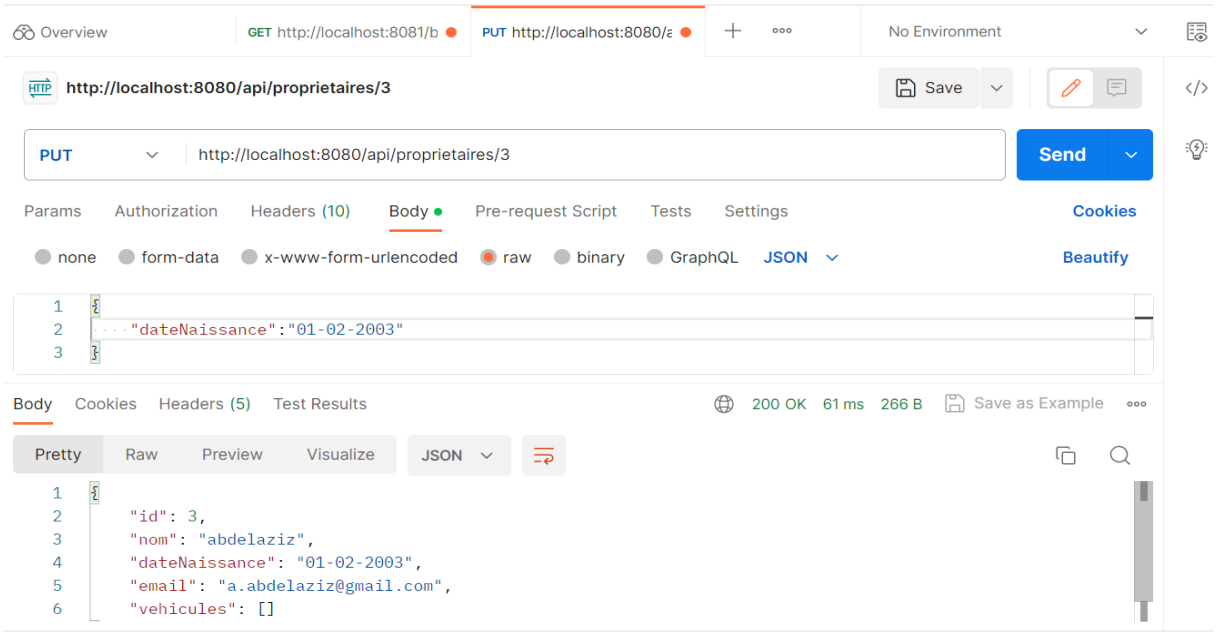


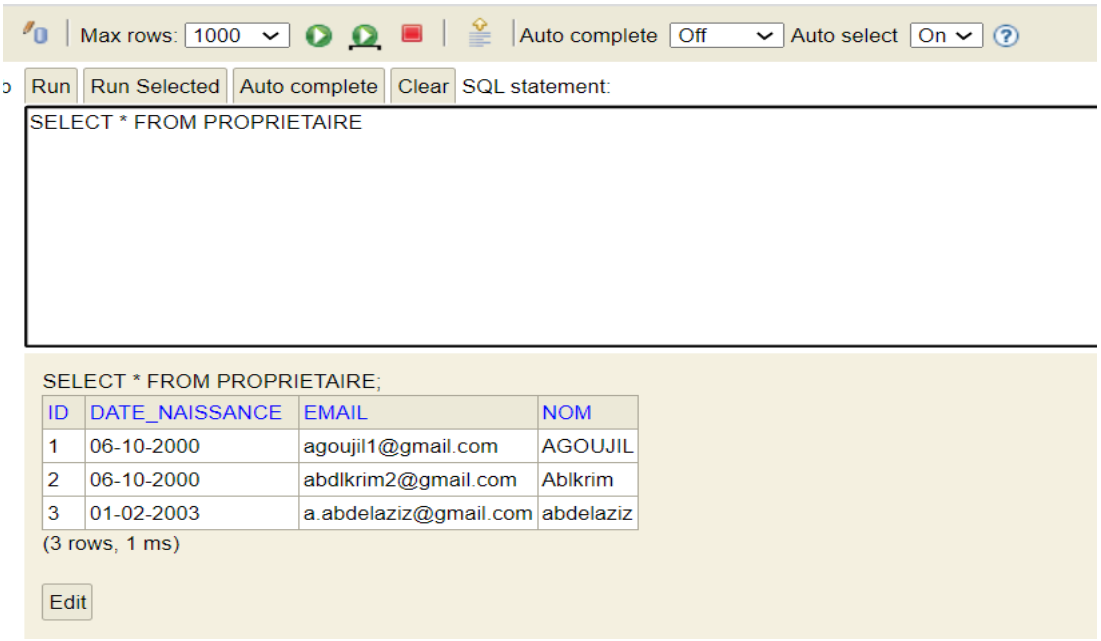
➢ Méthode POST



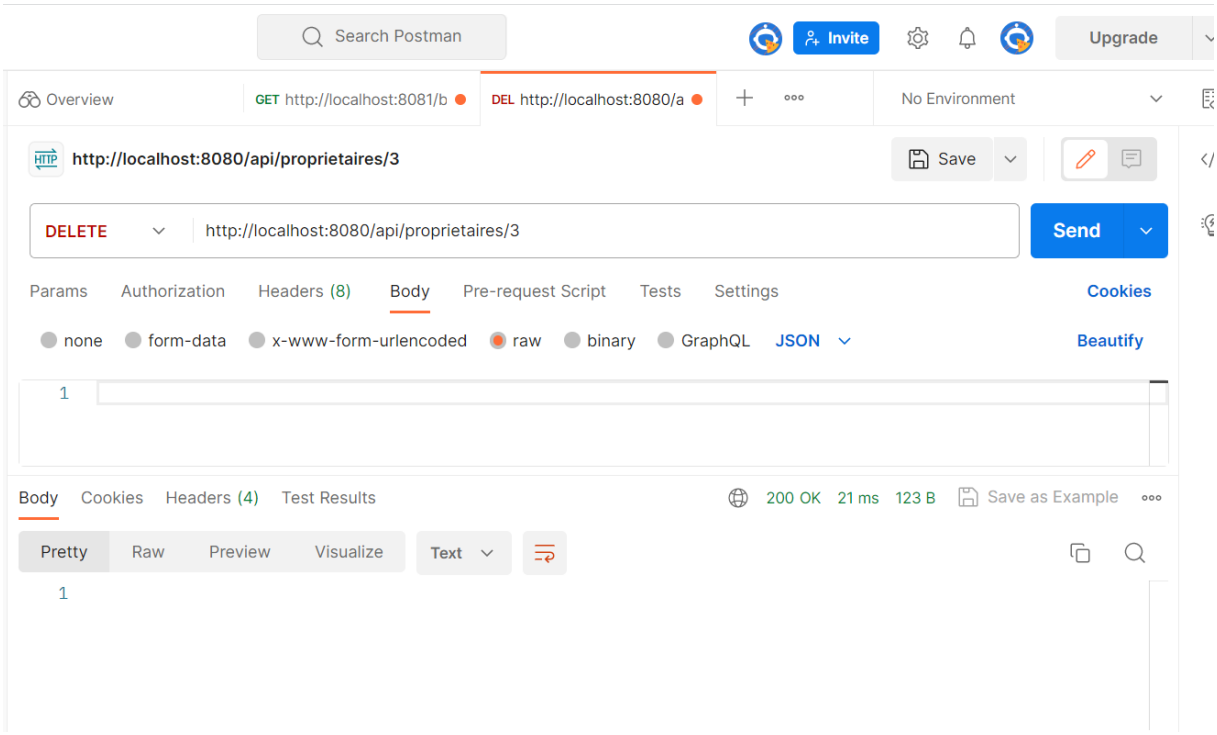


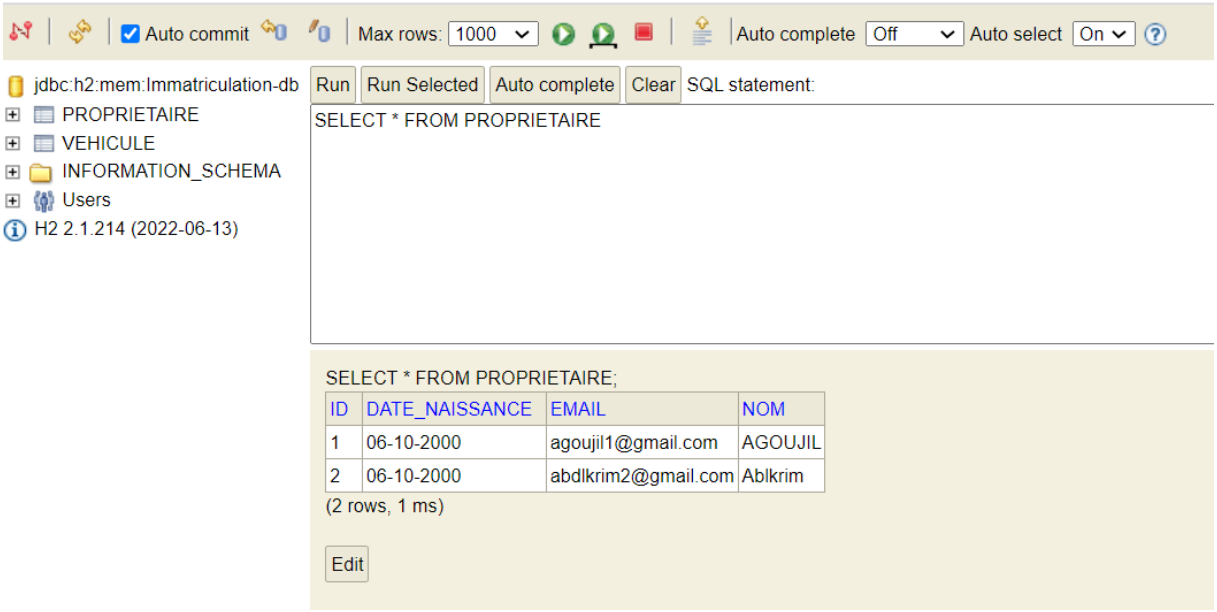
➢ Méthode PUT



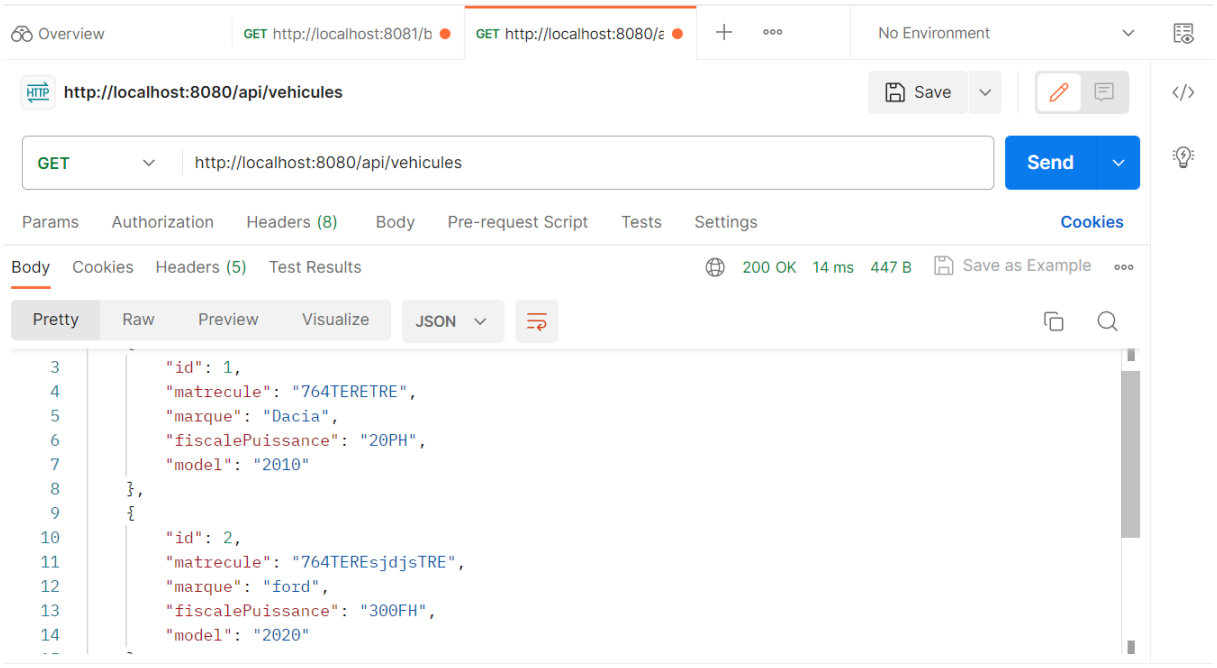


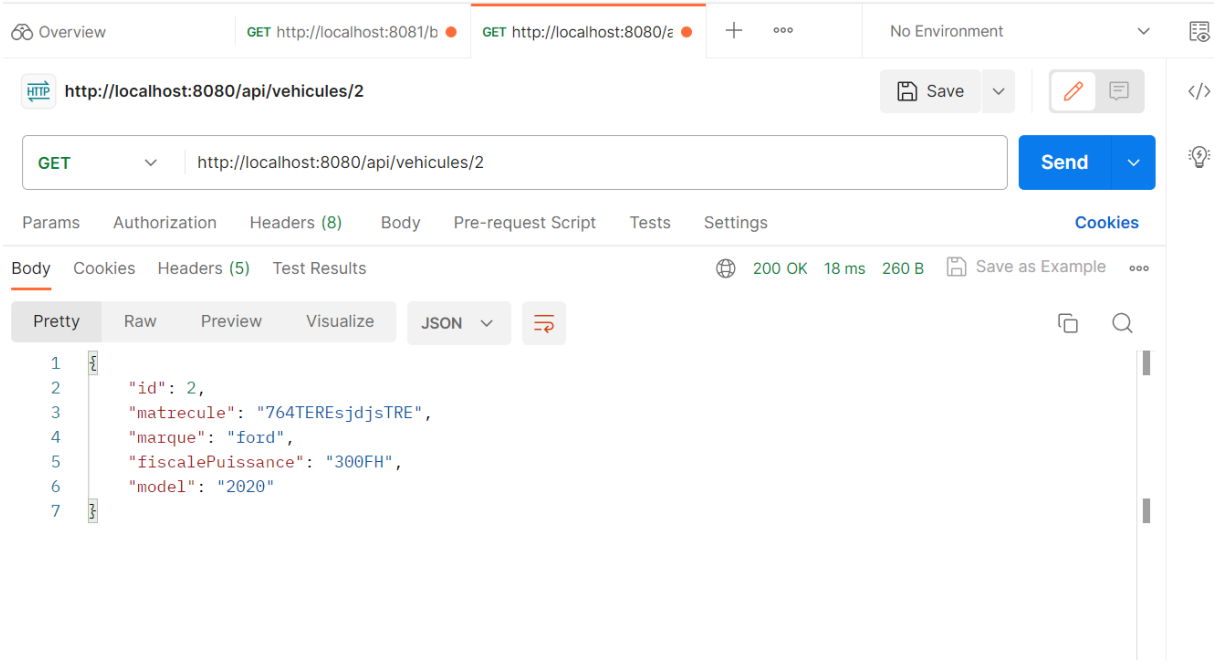
➢ Méthode DELETE



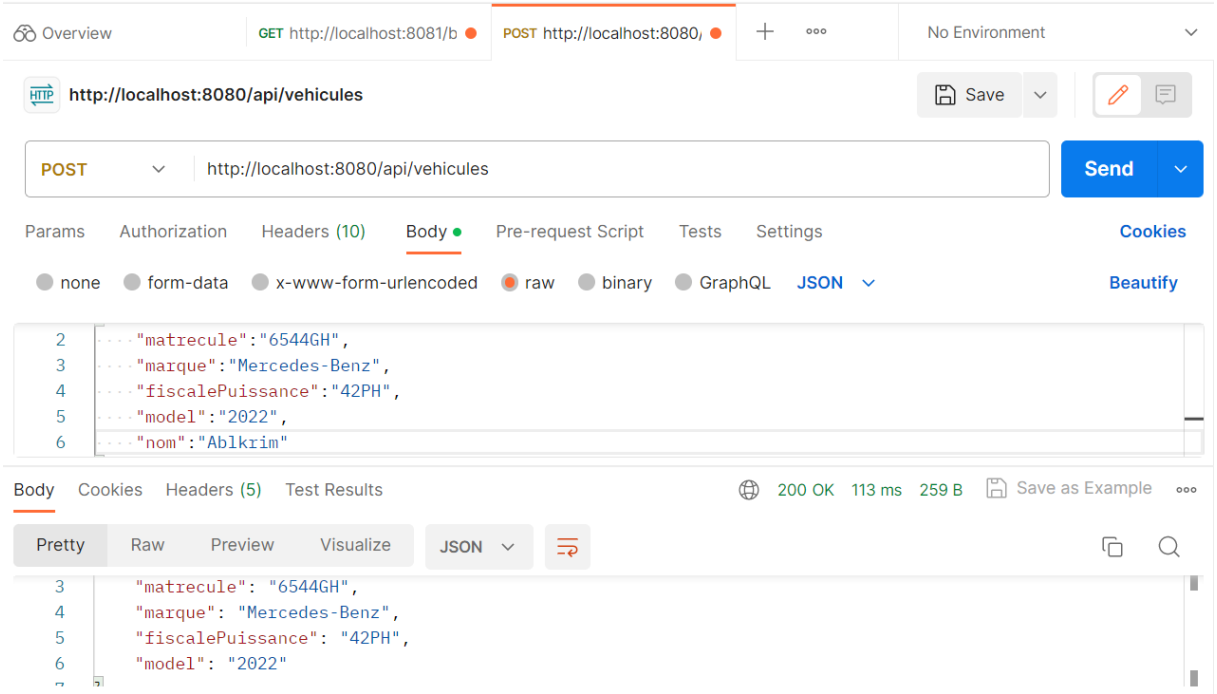


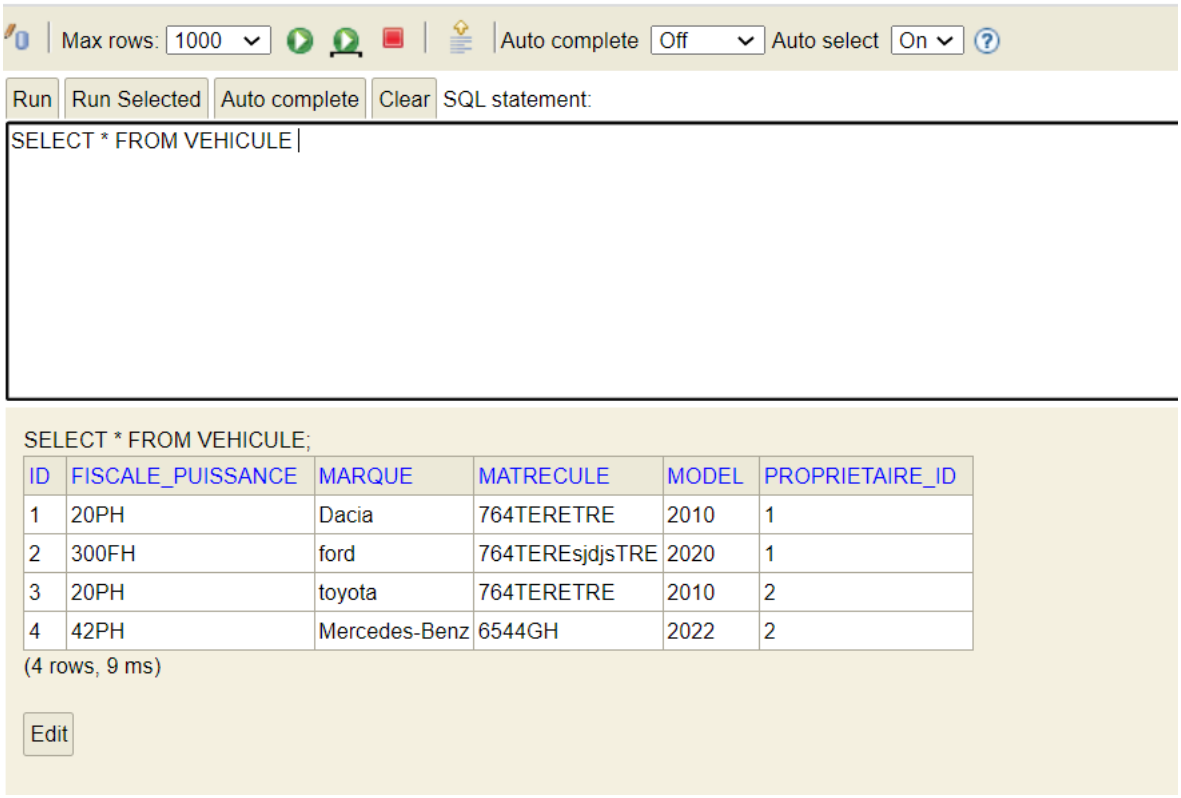
- Table Véhicule :
  - Méthode GET





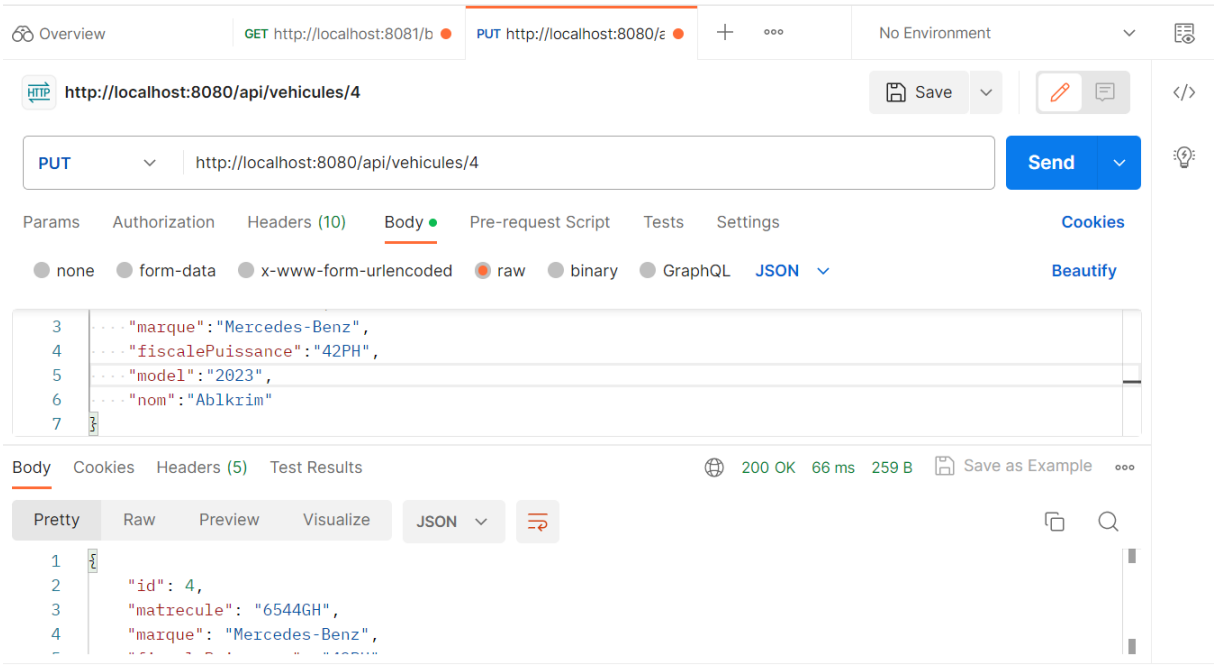
➢ Méthode POST

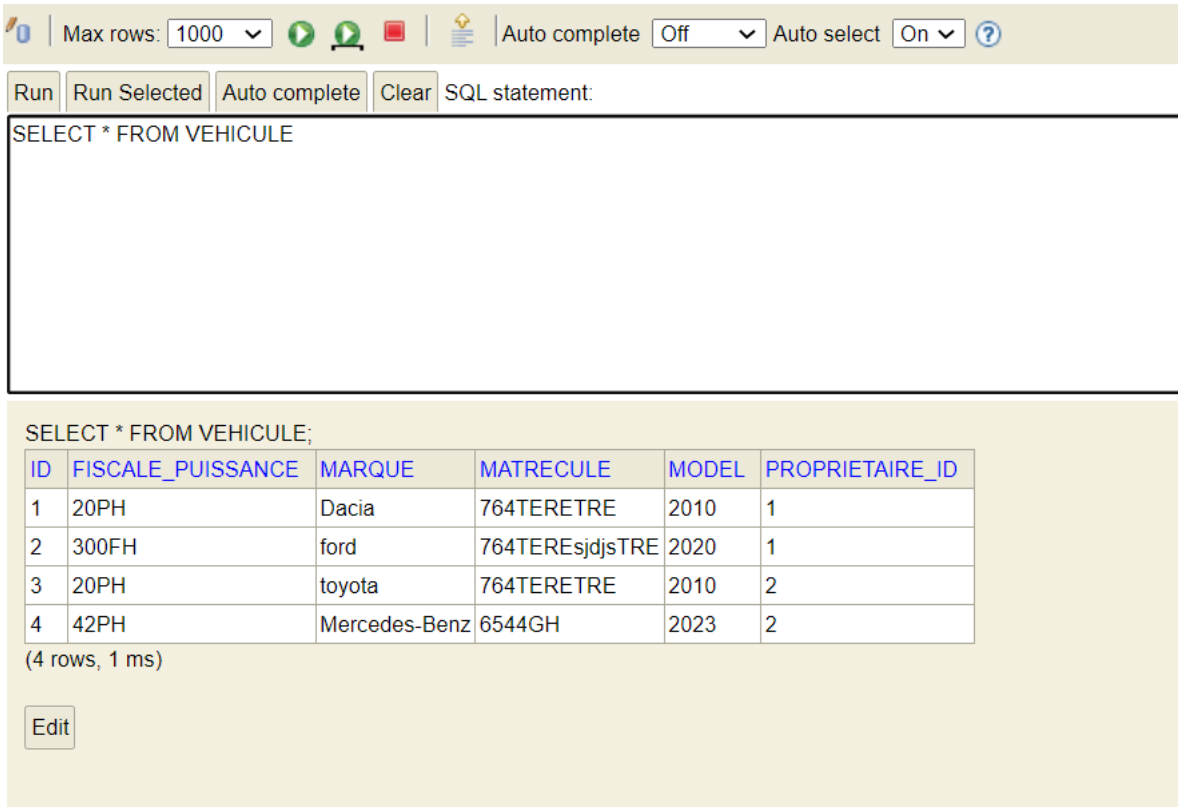




- Méthode POST

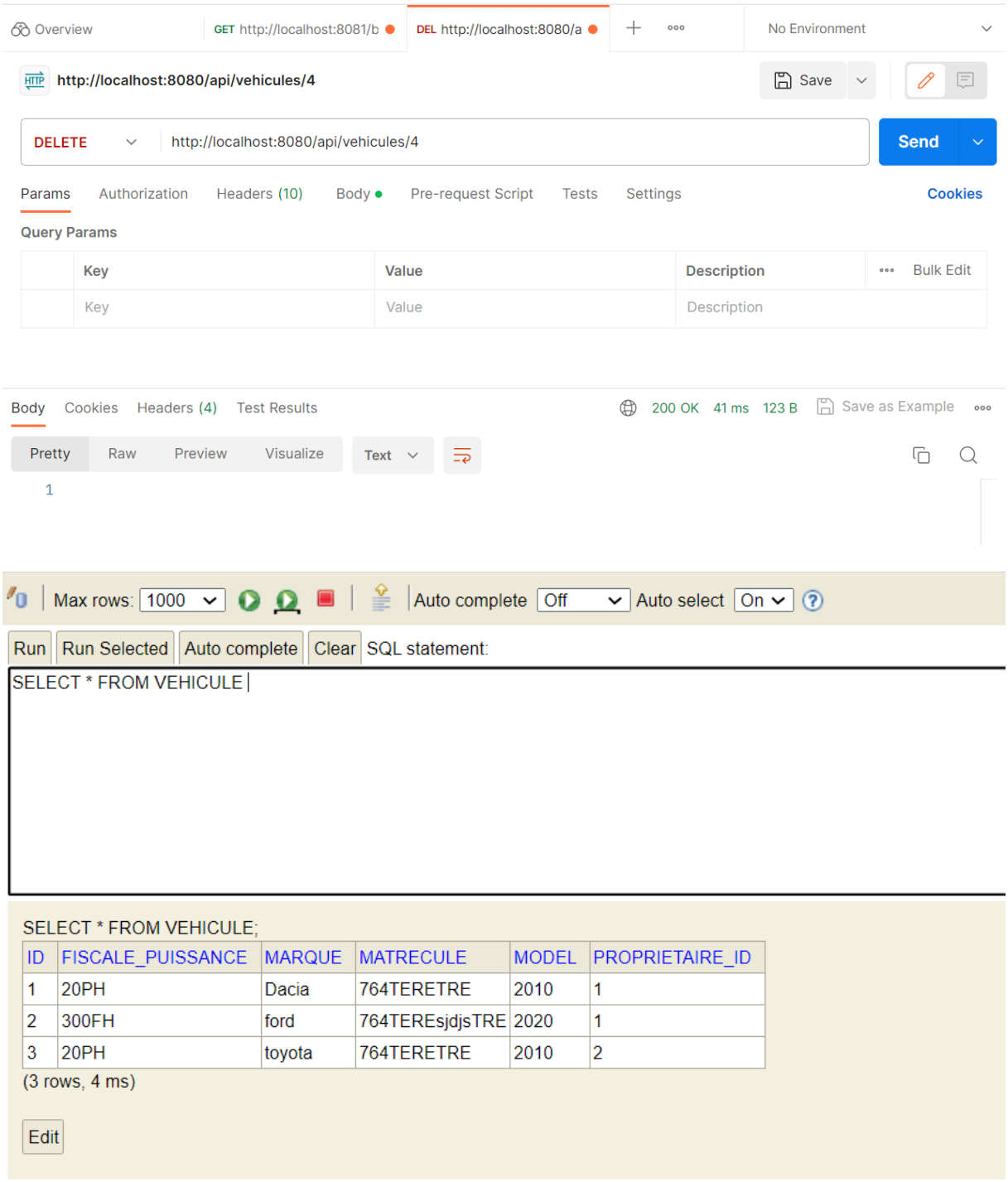
Changer le model





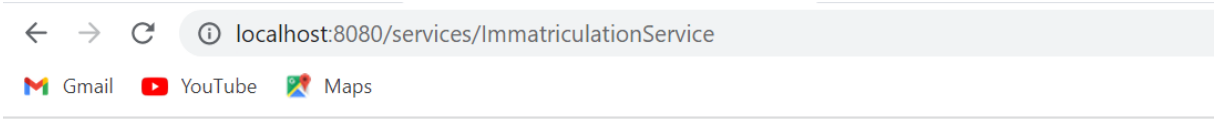
- Méthode DELETE

Supprimer le dernier véhicule ajouter :



- **Tester SOAP**

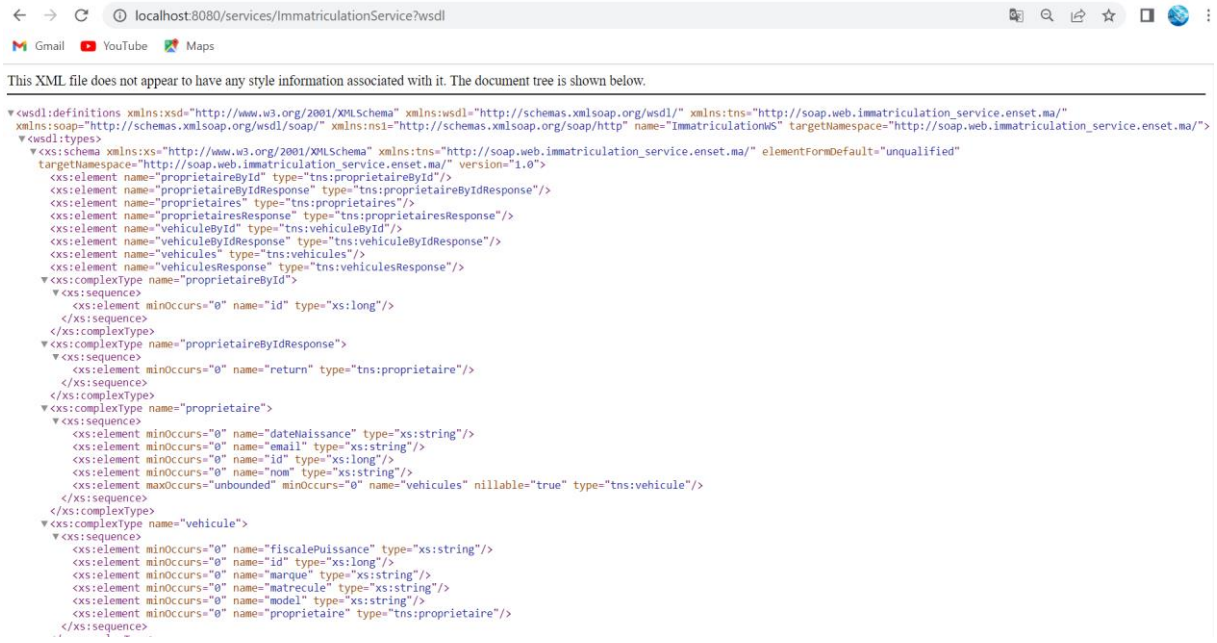
  - Afficher le service Soap :



This XML file does not appear to have any style information associated with it. The document tree is shown below.

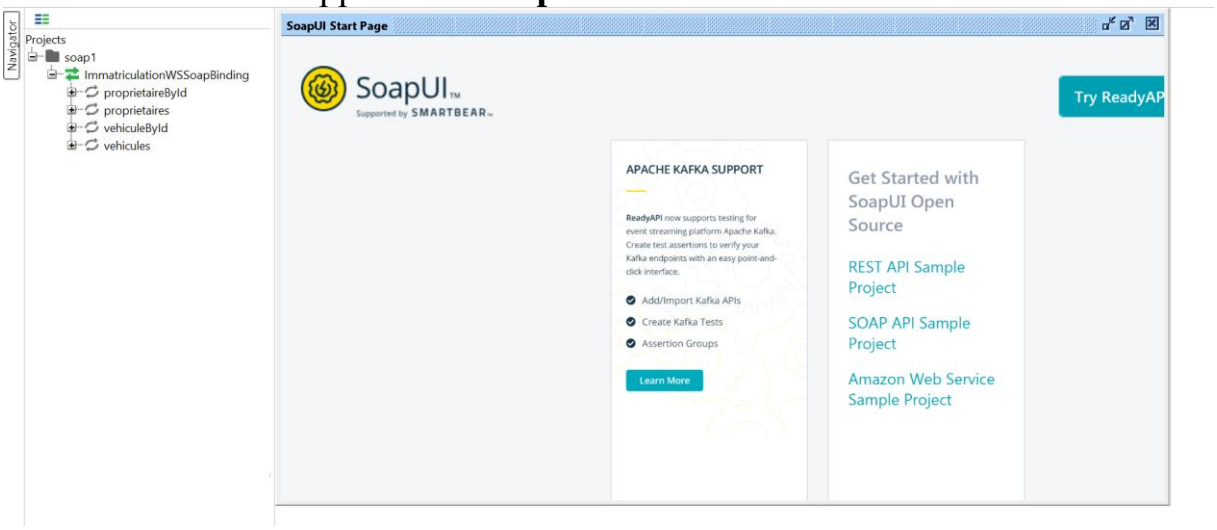
```
▼<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  ▼<soap:Body>
    ▼<soap:Fault>
       <faultcode>soap:Server</faultcode>
       <faultstring>No binding operation info while invoking unknown method with params unknown.</faultstring>
     </soap:Fault>
   </soap:Body>
 </soap:Envelope>
```

- Afficher la documentation **WSDL**

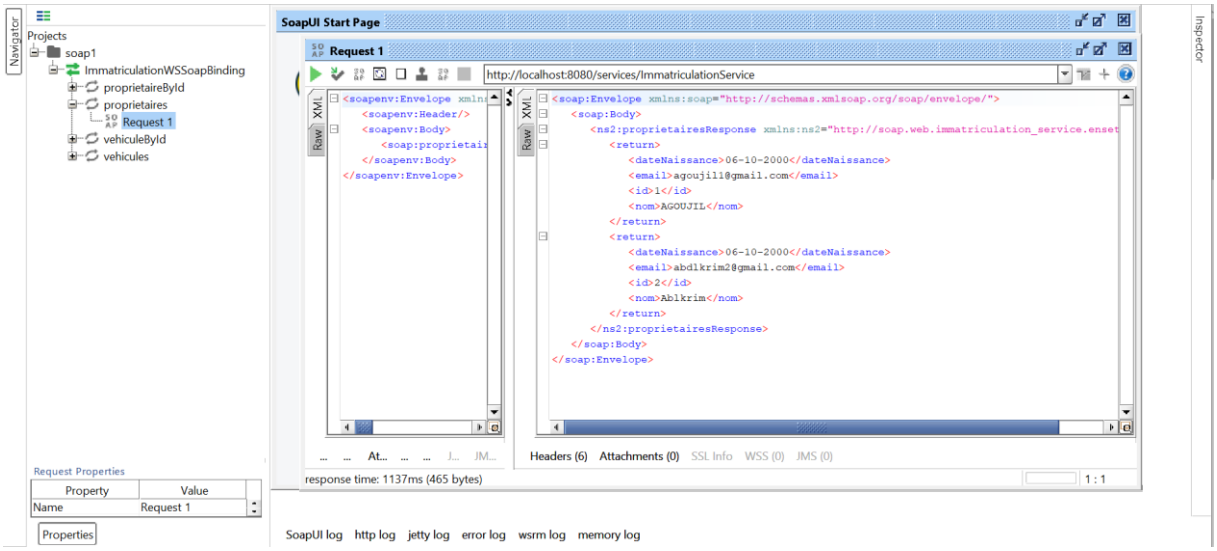



```
▼<wsdl:definitions xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="http://soap.web.immatriculation_service.enset.ma/"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:ns1="http://schemas.xmlsoap.org/soap/http" name="ImmatriculationWS" targetNamespace="http://soap.web.immatriculation_service.enset.ma/">
  ▼<wsdl:types>
    ▼<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://soap.web.immatriculation_service.enset.ma/" elementFormDefault="unqualified"
     targetNamespace="http://soap.web.immatriculation_service.enset.ma/" version="1.0">
       <xs:element name="proprietaireById" type="tns:proprietaireById"/>
       <xs:element name="proprietaireByIdResponse" type="tns:proprietaireByIdResponse"/>
       <xs:element name="proprietaires" type="tns:proprietaires"/>
       <xs:element name="proprietairesResponse" type="tns:proprietairesResponse"/>
       <xs:element name="vehiculeById" type="tns:vehiculeById"/>
       <xs:element name="vehiculeByIdResponse" type="tns:vehiculeByIdResponse"/>
       <xs:element name="vehicules" type="tns:vehicules"/>
       <xs:element name="vehiculesResponse" type="tns:vehiculesResponse"/>
      ▼<xs:complexType name="proprietaireById">
        ▼<xs:sequence>
           <xs:element minOccurs="0" name="id" type="xs:long"/>
         </xs:sequence>
       </xs:complexType>
      ▼<xs:complexType name="proprietaireByIdResponse">
        ▼<xs:sequence>
           <xs:element minOccurs="0" name="return" type="tns:proprietaire"/>
         </xs:sequence>
       </xs:complexType>
      ▼<xs:complexType name="proprietaire">
        ▼<xs:sequence>
           <xs:element minOccurs="0" name="dateNaissance" type="xs:string"/>
           <xs:element minOccurs="0" name="email" type="xs:string"/>
           <xs:element minOccurs="0" name="id" type="xs:long"/>
           <xs:element minOccurs="0" name="nom" type="xs:string"/>
           <xs:element maxOccurs="unbounded" minOccurs="0" name="vehicules" nillable="true" type="tns:vehicule"/>
         </xs:sequence>
       </xs:complexType>
      ▼<xs:complexType name="vehicule">
        ▼<xs:sequence>
           <xs:element minOccurs="0" name="fiscalePuissance" type="xs:string"/>
           <xs:element minOccurs="0" name="id" type="xs:long"/>
           <xs:element minOccurs="0" name="marque" type="xs:string"/>
           <xs:element minOccurs="0" name="matrecule" type="xs:string"/>
           <xs:element minOccurs="0" name="model" type="xs:string"/>
           <xs:element minOccurs="0" name="proprietaire" type="tns:proprietaire"/>
         </xs:sequence>
```

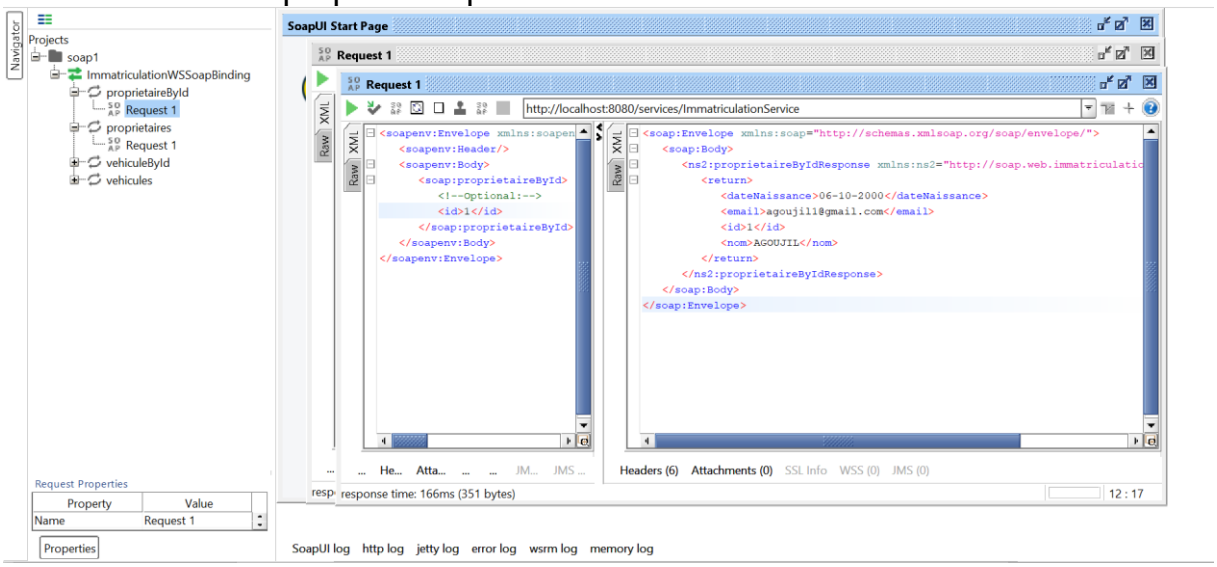
- En utilisant l'application **SoapUI**



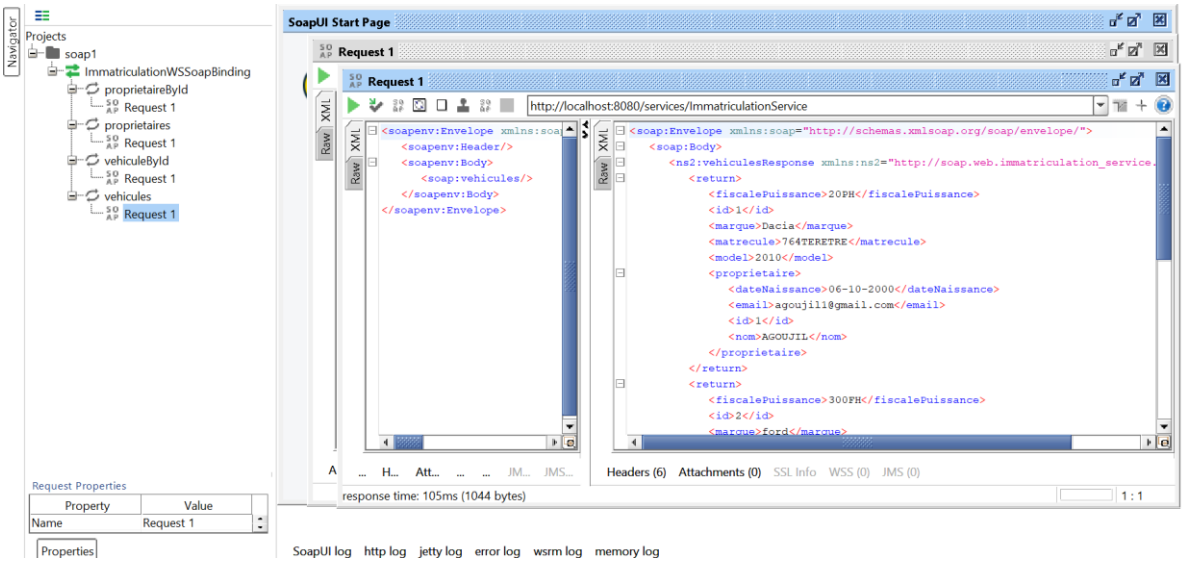
- Afficher les propriétaires :



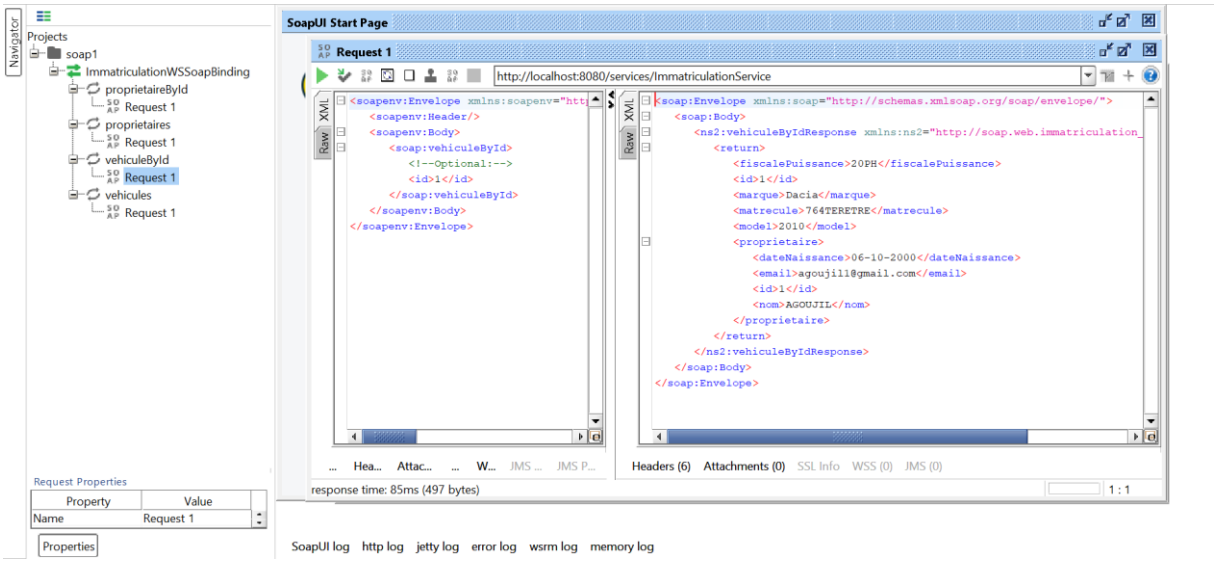
- Afficher un propriétaire par son **id**



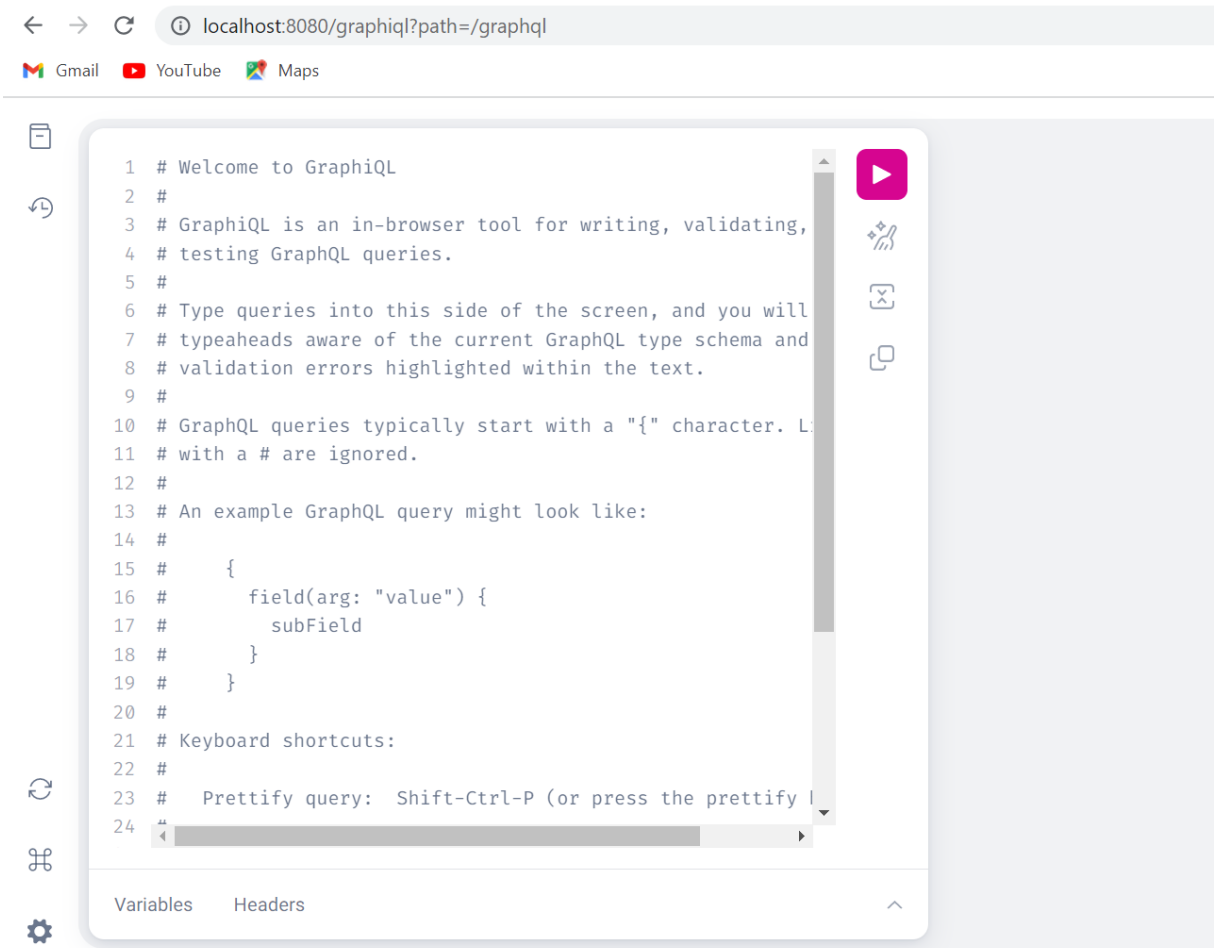
- Afficher les véhicules :



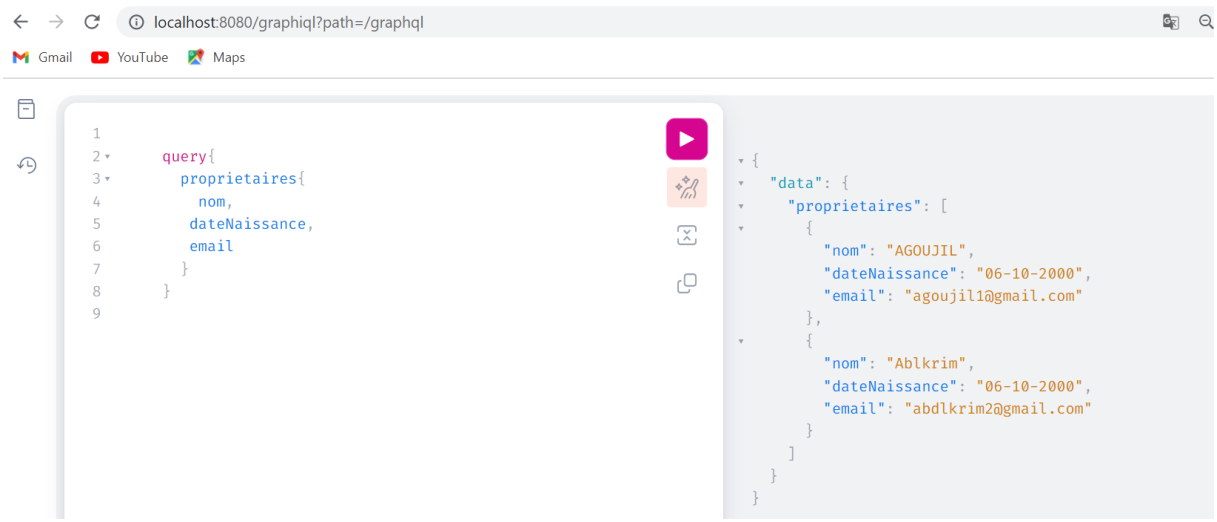
- Afficher les véhicules par leur **id**



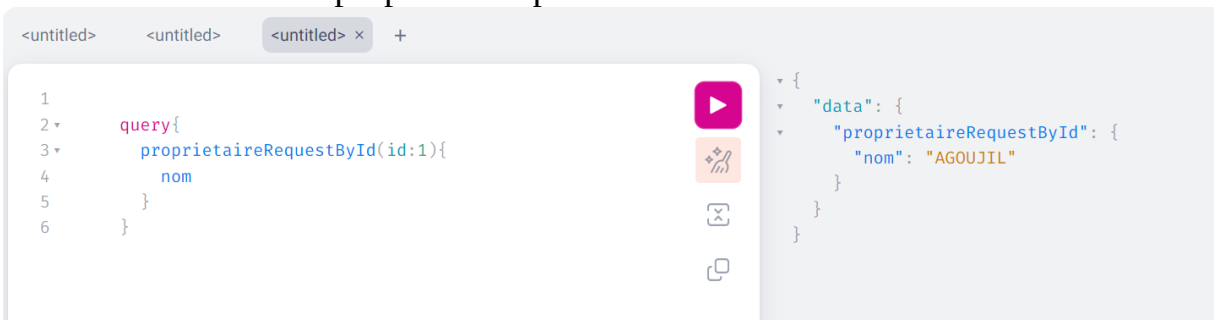
- **Tester GRAPHQL :**



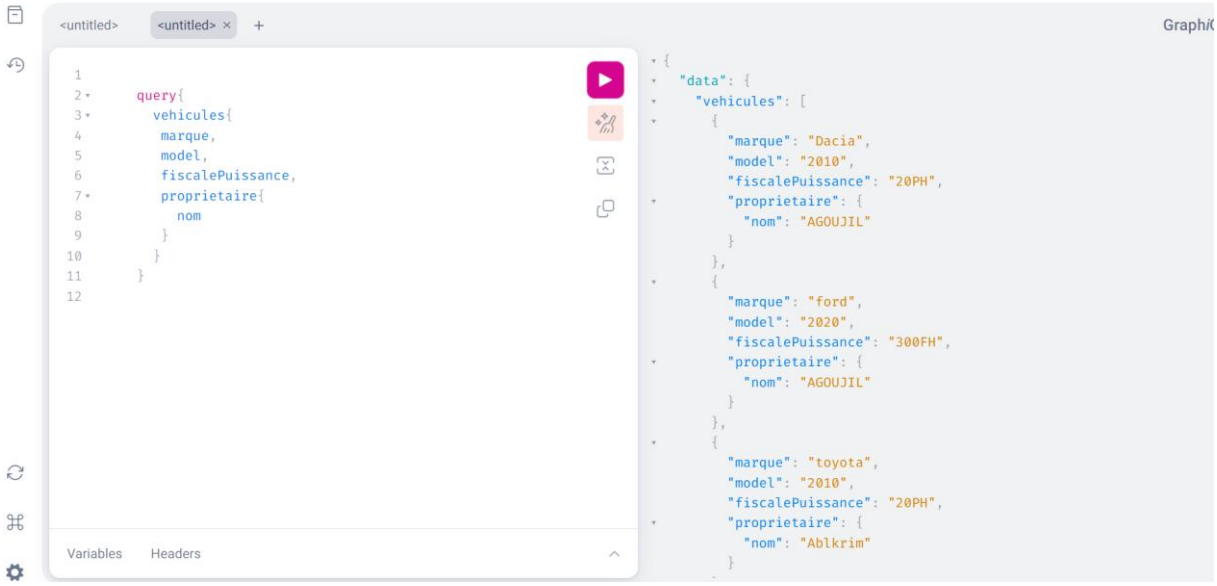
  - Afficher les propriétaires :



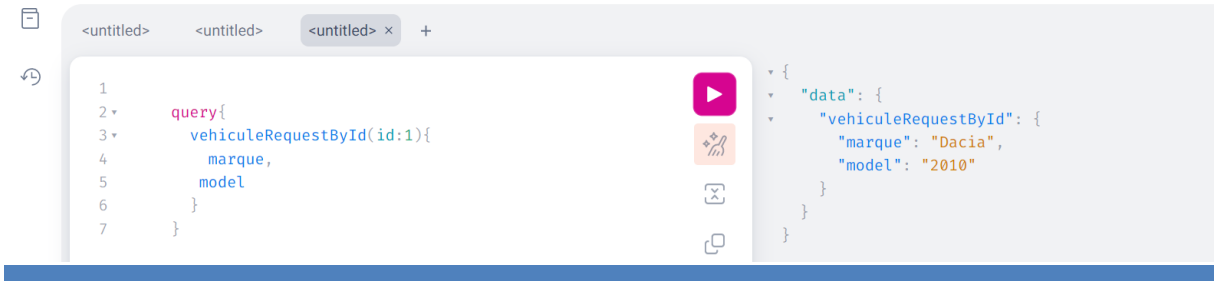
- Afficher les propriétaires par leur **id** :



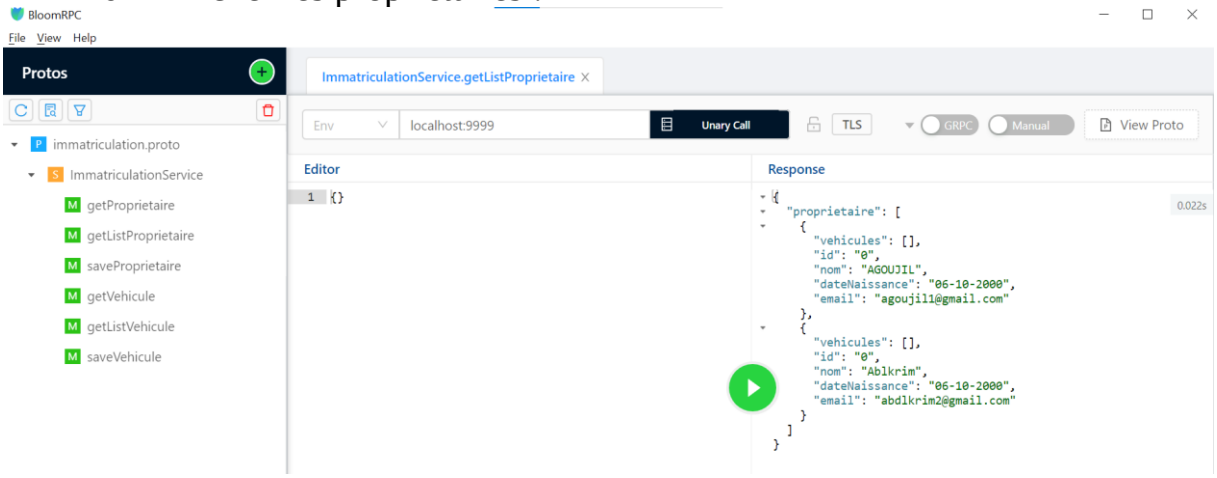
- Afficher les véhicules :



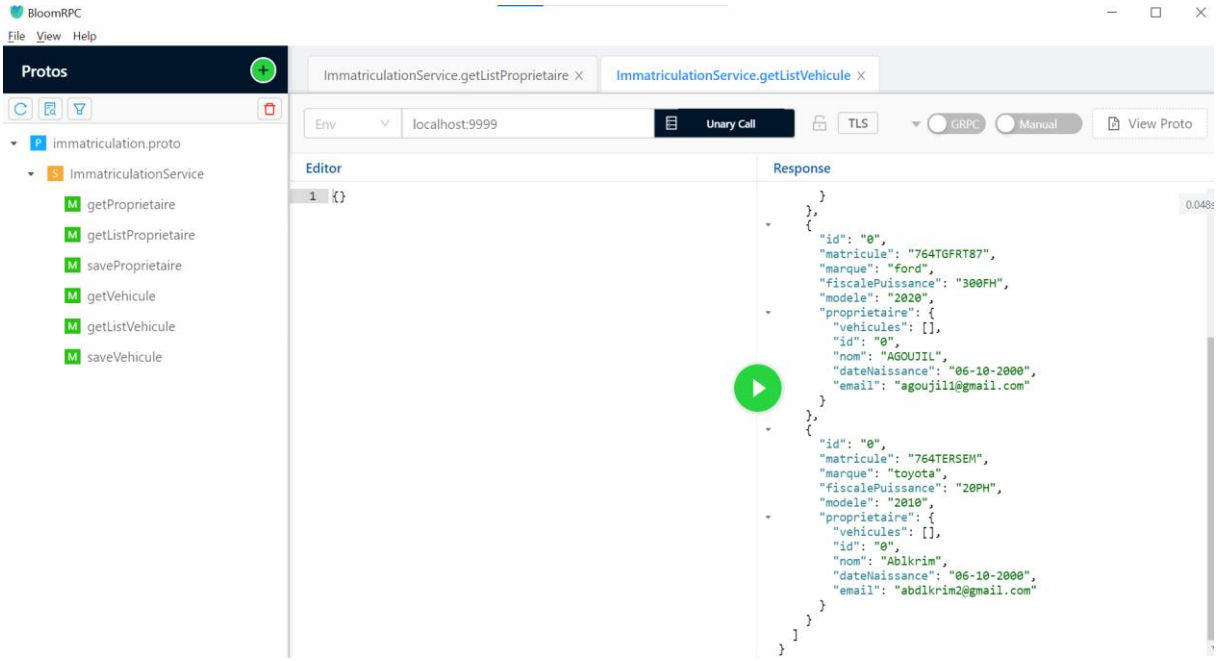
- Afficher les véhicules par leur **id** :



- **Tester GRPC :**
  - Afficher les propriétaires :



  - Afficher les véhicules :



4. Développer le micro-service Infractions:

a. Entités JPA et Interface JpaRepository basées sur Spring data

- **Entité Infraction**

```
package ma.enset.infractionsservice.entities;


import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
```

```
import java.util.Date;

@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor @Builder
public class Infraction {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String matricule;
    private double vitesseVehicle;
    private Date dateInfraction;
    private Long radarId;
    private Long vehiculeId;
    private double maxVitesseAccepte;
    private double montant;
}
```

- **InfractionRepository**

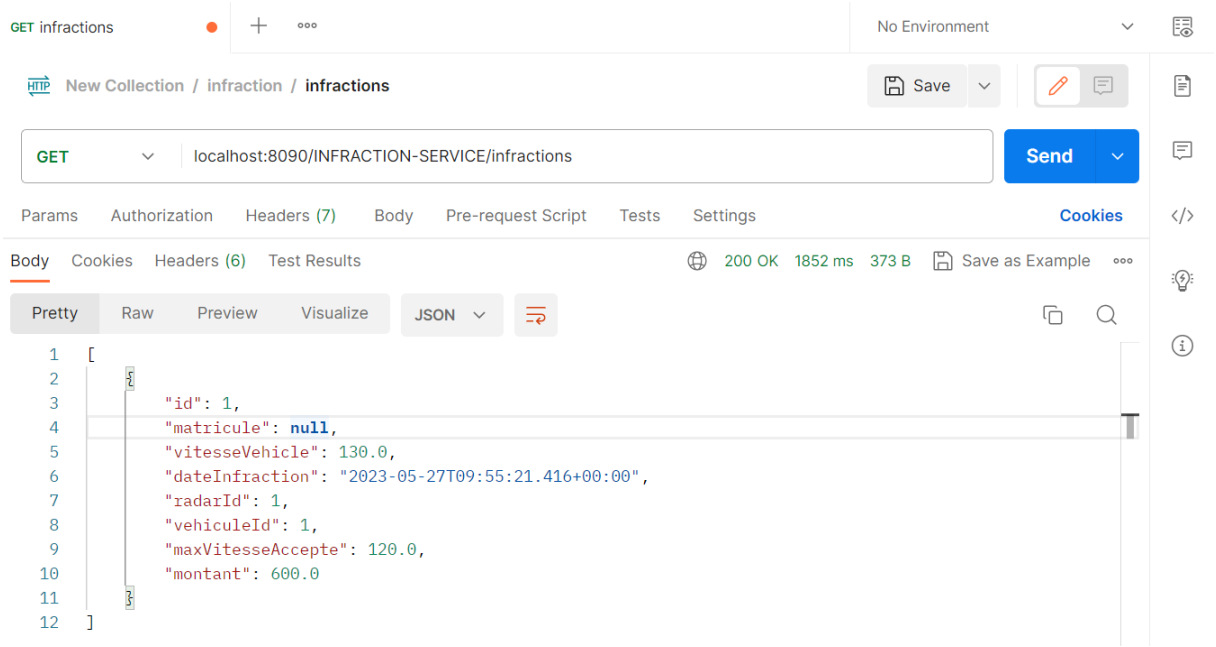
```
package ma.enset.infractionsservice.repositories;

import ma.enset.infractionsservice.entities.Infraction;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

public interface InfractionRepository extends
JpaRepository<Infraction,Long> {

}
```

- **InfractionRestController**

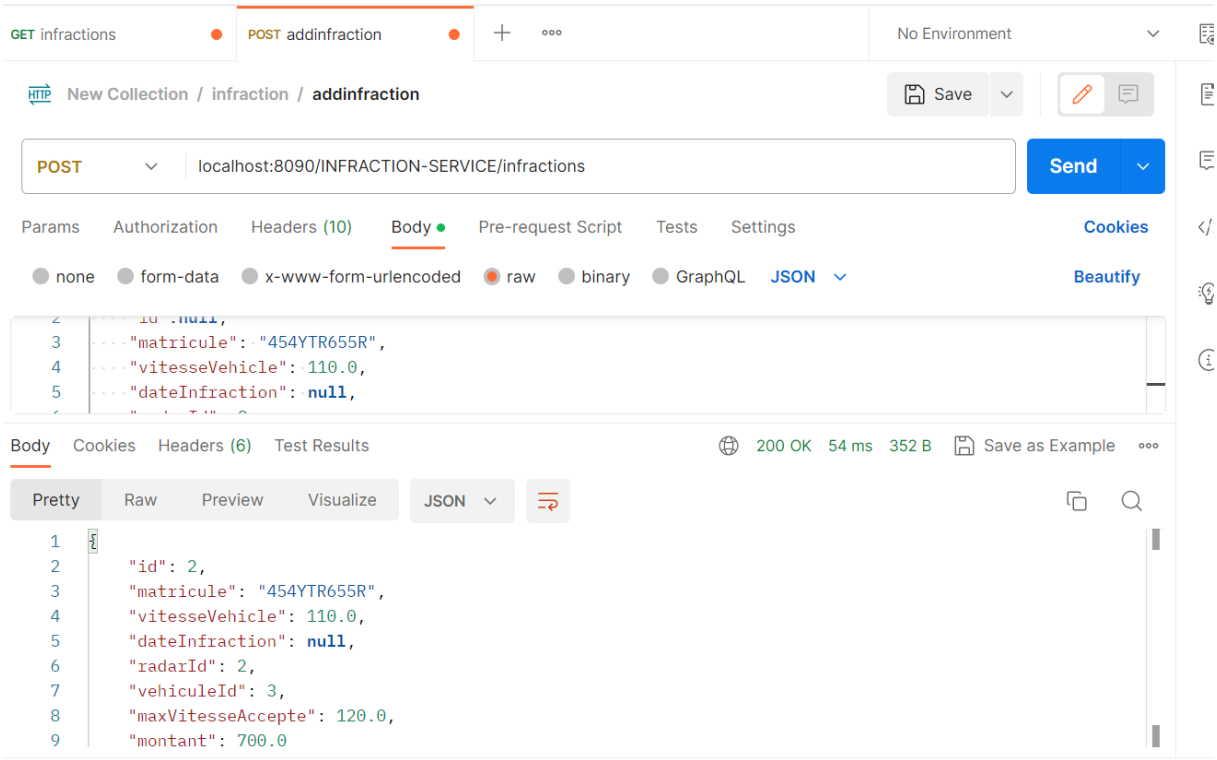
```
package ma.enset.infractionsservice.controller;

import lombok.AllArgsConstructor;
import ma.enset.infractionsservice.entities.Infraction;
import ma.enset.infractionsservice.repositories.InfractionRepository;
import org.springframework.stereotype.Service;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@AllArgsConstructor
public class InfractionRestController {
    private InfractionRepository infractionRepository;

    @GetMapping("/infractions")
        public List<Infraction> infractions(){
            return infractionRepository.findAll();

        }
        @GetMapping("/infractions/{id}")
        public Infraction infraction(@PathVariable String id){
            return infractionRepository.findById(Long.valueOf(id))
                    .orElseThrow(()->new
RuntimeException(String.format("infraction % not found",id)));
        }
        @PostMapping("/infractions")
        public Infraction save(@RequestBody Infraction infraction){
            return infractionRepository.save(infraction);
        }
```

```
        @PutMapping("/infractions/{id}")
        public Infraction update(@PathVariable String id,@RequestBody
Infraction infraction){
            Infraction
infraction1=infractionRepository.findById(Long.valueOf(id)).orElseThr
ow();
            if(infraction.getMatricule()!=null)
infraction1.setMatricule(infraction.getMatricule());
            if(infraction.getDateInfraction()!=null)
infraction1.setDateInfraction (infraction.getDateInfraction());
            if(infraction.getVitesseVehicle()!=0.0)
infraction1.setVitesseVehicle (infraction.getVitesseVehicle());
            if(infraction.getRadarId()!=null)
infraction1.setRadarId(infraction.getRadarId());
            if(infraction.getMontant()!=0.0)
infraction1.setMontant(infraction.getMontant());
            if(infraction.getMaxVitesseAccepte()!=0.0)
infraction1.setMaxVitesseAccepte(infraction.getMaxVitesseAccepte());
            return infractionRepository.save(infraction1);
        }
        @DeleteMapping("/infractions/{id}")
        public void delete(@PathVariable String id){
            infractionRepository.deleteById(Long.valueOf(id));
        }
    }
```

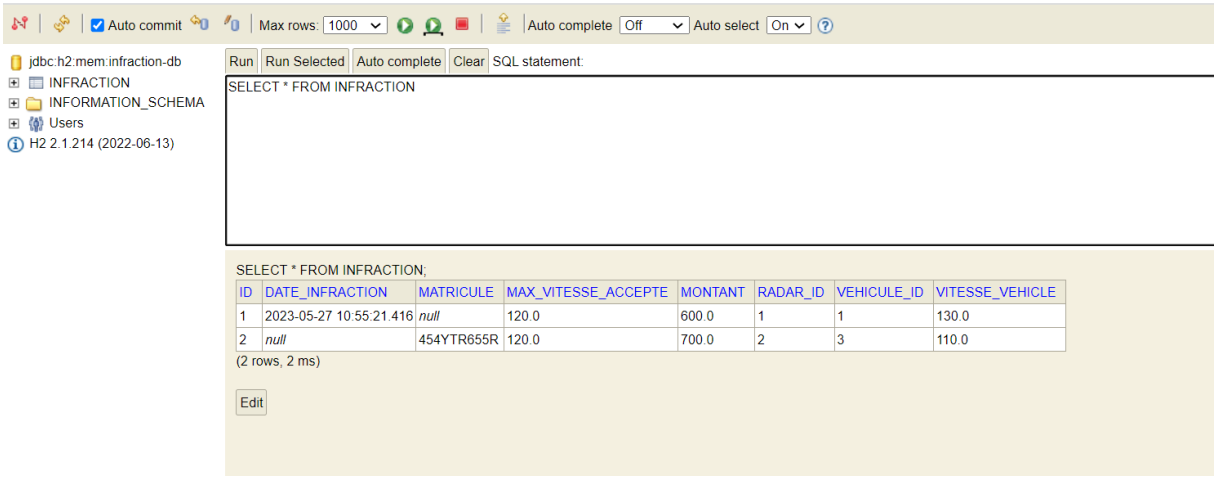
- **Tester l'Infraction :**
  - Méthode GET



  - Méthode POST

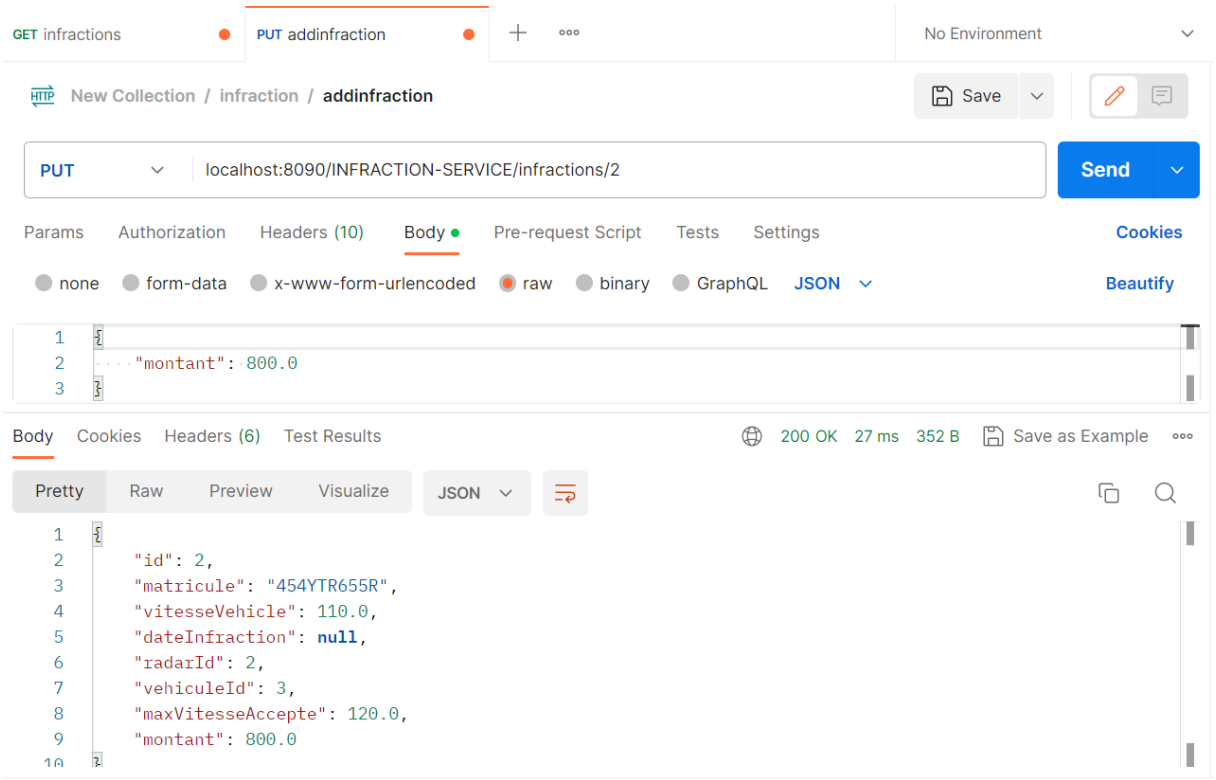


Vérifier dans la base H2 console :

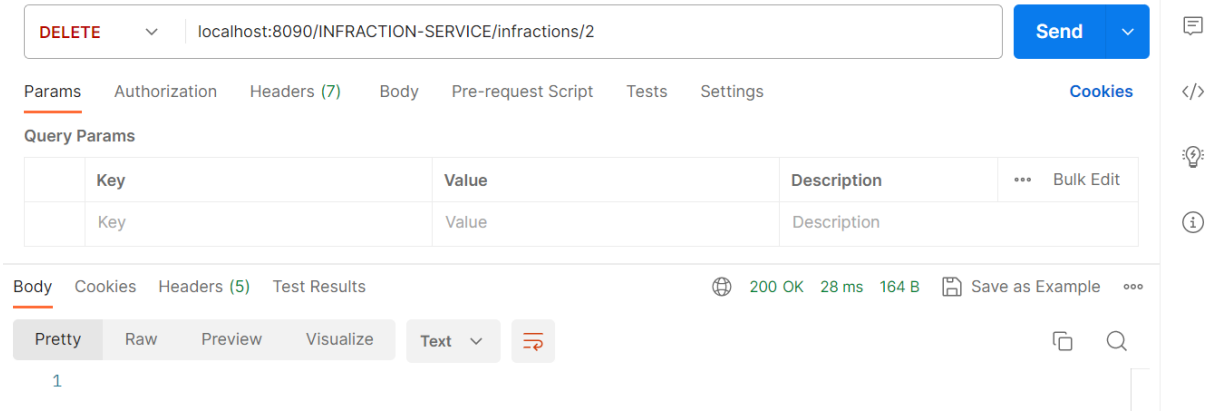


- Méthode PUT

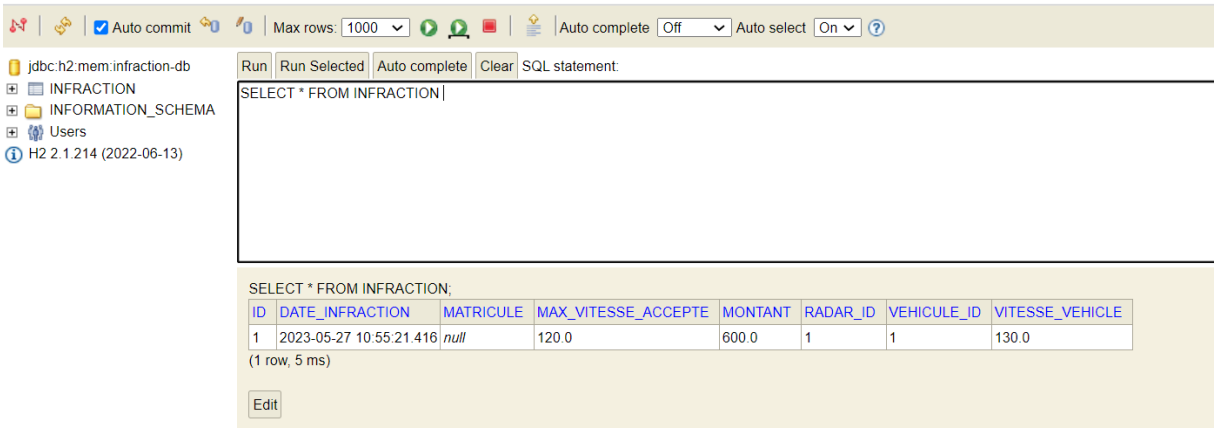
Changer le montant de la deuxième infraction :



- Méthode DELET



Vérifier dans la base de données H2 console :



5. Développer le micro-service Radar.

Chaque dépassement de vitesse, ce service devrait consulter le service d'immatriculation pour récupérer les informations sur le propriétaire du véhicule. Ensuite il fait appel au service

Infraction pour générer une nouvelle infraction. La communication entre les services peut se faire au choix entre REST, SOAP, GRPC ou GraphQL.

a. Entités JPA et Interface JpaRepository basées sur Spring data

- **Entité Radar**

```
package ma.enset.radarservice.entities;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
@Builder
public class Radar {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private double maxVitesseAccepte;
    private double longitude;
    private double latitude;
}
```

- **RadarRepository**

```
package ma.enset.radarservice.repositories;

import ma.enset.radarservice.entities.Radar;
import org.springframework.data.jpa.repository.JpaRepository;

public interface RadarRepository extends JpaRepository <Radar,Long>{
}
```

- **RadarRestController**

```
package ma.enset.radarservice.web.rest;

import ma.enset.radarservice.entities.Radar;
import ma.enset.radarservice.repositories.RadarRepository;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
public class RadarRestController {

    RadarRepository radarRepository;

    public RadarRestController(RadarRepository radarRepository) {
        this.radarRepository = radarRepository;
```

```
    }

    @GetMapping("/radars")
    public List<Radar> getRadars(){
        return radarRepository.findAll();
    }

    //  *********
    @GetMapping("/radars/{id}")
    public Radar getRadarById(@PathVariable Long id){
        return radarRepository.findById(id).get();
    }

    @PostMapping("/radars")
    public Radar addVehicule(@RequestBody Radar radar){
        return radarRepository.save(radar);
    }

}
```

- **Models:**
  - Infraction

```
package ma.enset.radarservice.models;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
import ma.enset.radarservice.entities.Radar;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;

import java.util.Date;

@Data
@AllArgsConstructor
@NoArgsConstructor
@Builder
public class Infraction {

    private Long id;
    private Date dateInfraction;
    private Long radarId;
    private Long vehiculeId;
    private double vitesseVehicule;
    private double maxVitesseAccepte;
    private double montant;
    private boolean state;
    private Vehicule vehicule;
    private Radar radar;
}
```

  - Proprietaire

```
package ma.enset.radarservice.models;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
```

```
@AllArgsConstructor @NoArgsConstructor @Builder
public class Proprietaire {
    private Long id;
    private String nom;
    private String DateNaissance;
    private String email;
}
```

- Vehicule

```
package ma.enset.radarservice.models;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor @AllArgsConstructor @Builder
public class Vehicule {
    private Long id;
    private String matricule;
    private String marque;
    private int fiscalePuissance;
    private String model;
    private Proprietaire proprietaire;
}
```

- **feign :**
  - InfractionRestClient

```
package ma.enset.radarservice.feign;

import ma.enset.radarservice.models.Infraction;
import org.springframework.cloud.openfeign.FeignClient;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
@FeignClient(name = "INFRACTION-SERVICE")
public interface InfractionRestClient {
    @PostMapping("/infractions")
    Infraction save(@RequestBody Infraction infraction);
}
```

  - VehiculeRestClient

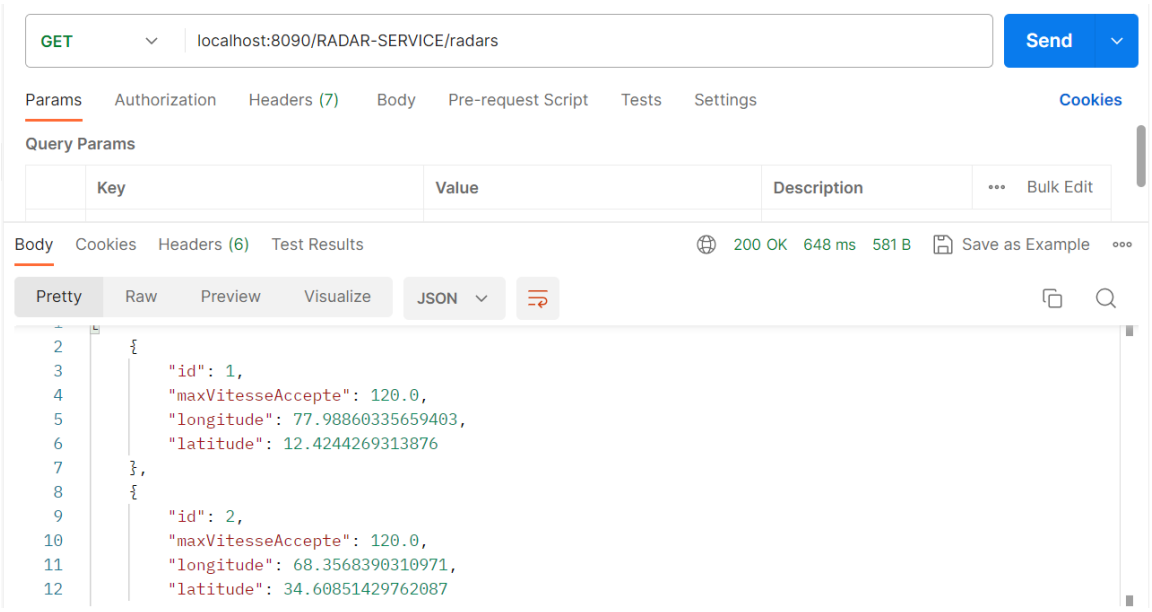
```
package ma.enset.radarservice.feign;

import ma.enset.radarservice.models.Vehicule;
import org.springframework.cloud.openfeign.FeignClient;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;

import java.util.List;
@FeignClient(name = "IMMATRICULATION-SERVICE")
public interface VehiculeRestClient {
    @GetMapping("/vehicules")
    List<Vehicule> getVehicules();
```

```
    @GetMapping("/vehicule")
    Vehicule getVehiculeByMatricule(@RequestParam String mat);
}
```

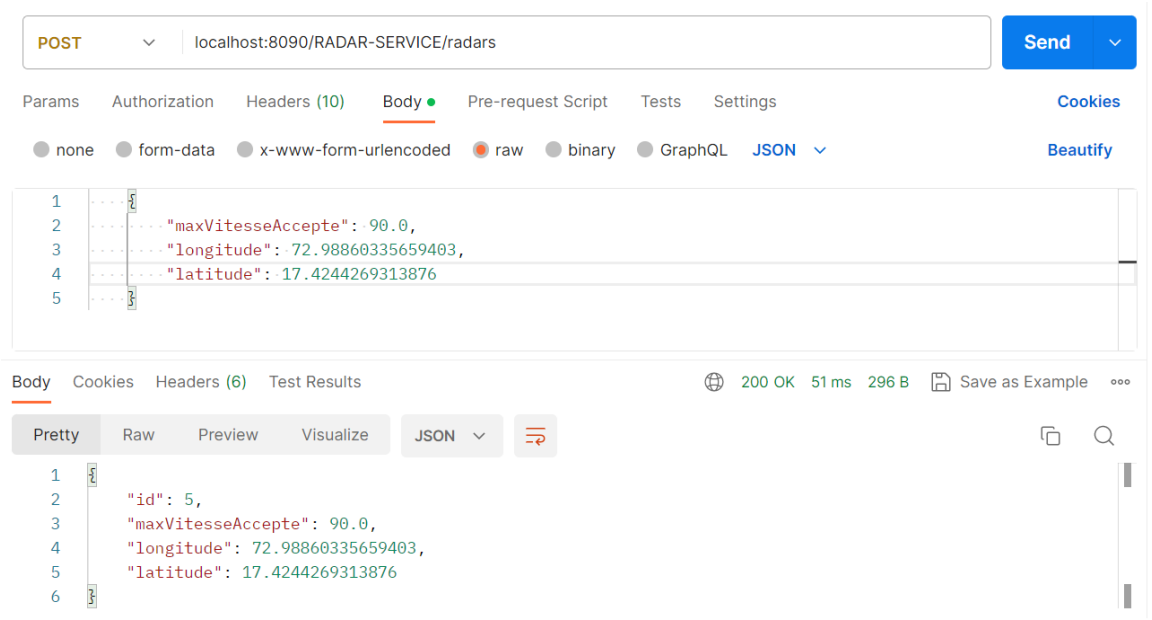
- **Tester l'Infraction :**
  - **RadarServiceApplication :**

```
o package ma.enset.radarservice;

   import ma.enset.radarservice.entities.Radar;
   import ma.enset.radarservice.repositories.RadarRepository;
   import org.springframework.boot.CommandLineRunner;
   import org.springframework.boot.SpringApplication;
   import
   org.springframework.boot.autoconfigure.SpringBootApplication;
   import org.springframework.cloud.openfeign.EnableFeignClients;
   import org.springframework.context.annotation.Bean;

   import java.text.DecimalFormat;

   @SpringBootApplication
   @EnableFeignClients
   public class RadarServiceApplication {

      public static void main(String[] args) {
         SpringApplication.run(RadarServiceApplication.class, args);
      }

      @Bean
      CommandLineRunner start(RadarRepository radarRepository){
         return args -> {
            DecimalFormat numberFormat = new DecimalFormat("#.0000");
            for (int i = 0; i < 4; i++) {
               radarRepository.save(
                     new Radar().builder()
                           .id(null)
                           .maxVitesseAccepte(120)
                           .longitude(Math.random()*100)
                           .latitude(Math.random()*100)
                           .build()
               );
            }
         };
      }
   }
```

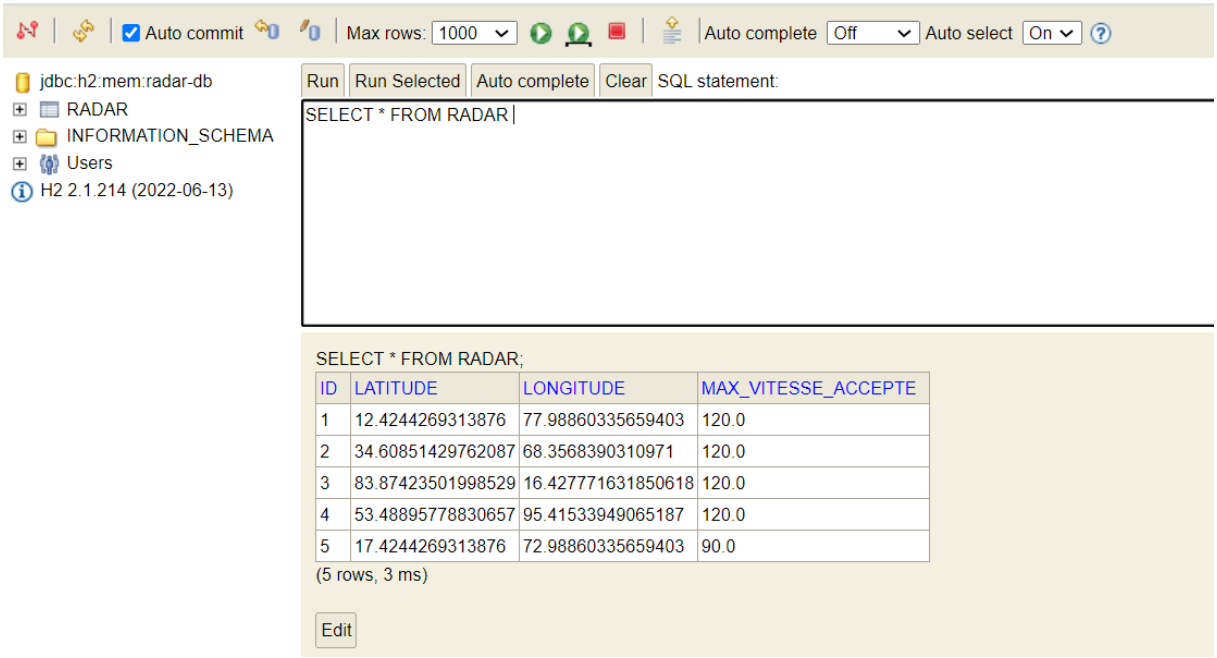
  - Méthode GET



- Méthode POST

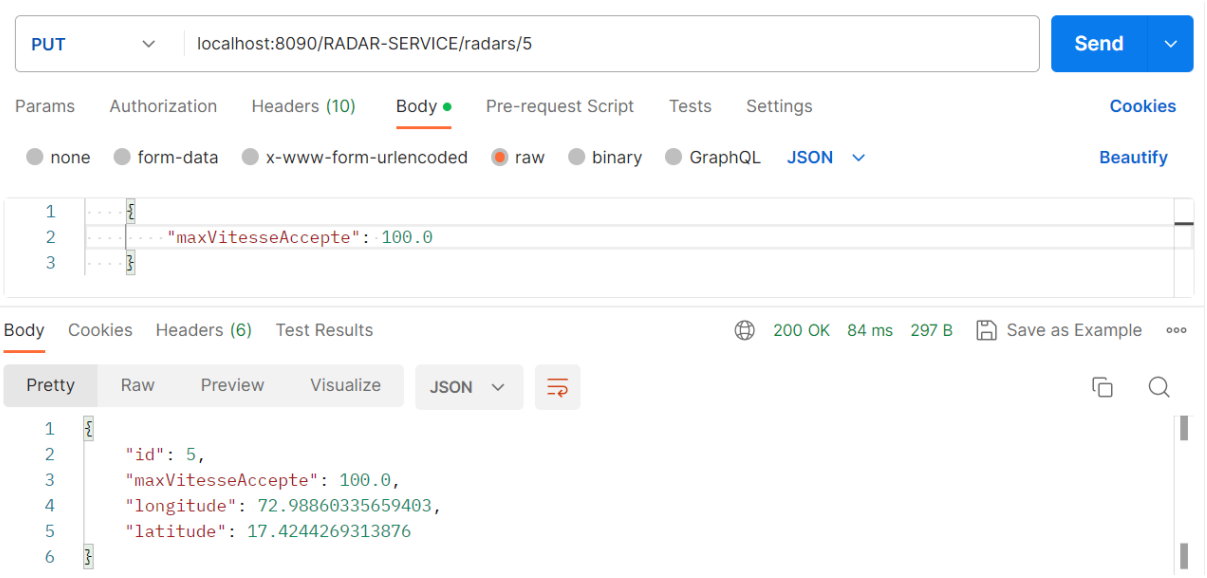


Vérifier dans la base H2 console :

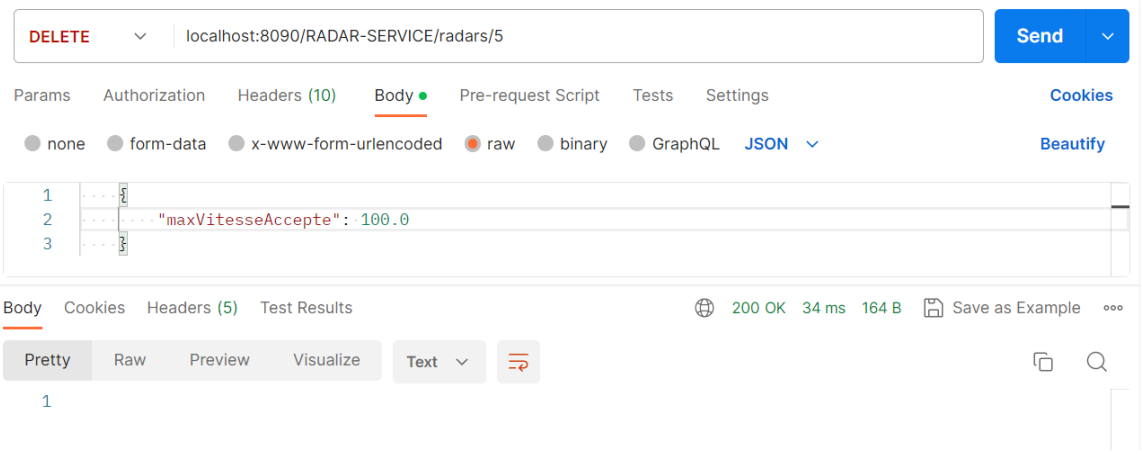


- Méthode PUT

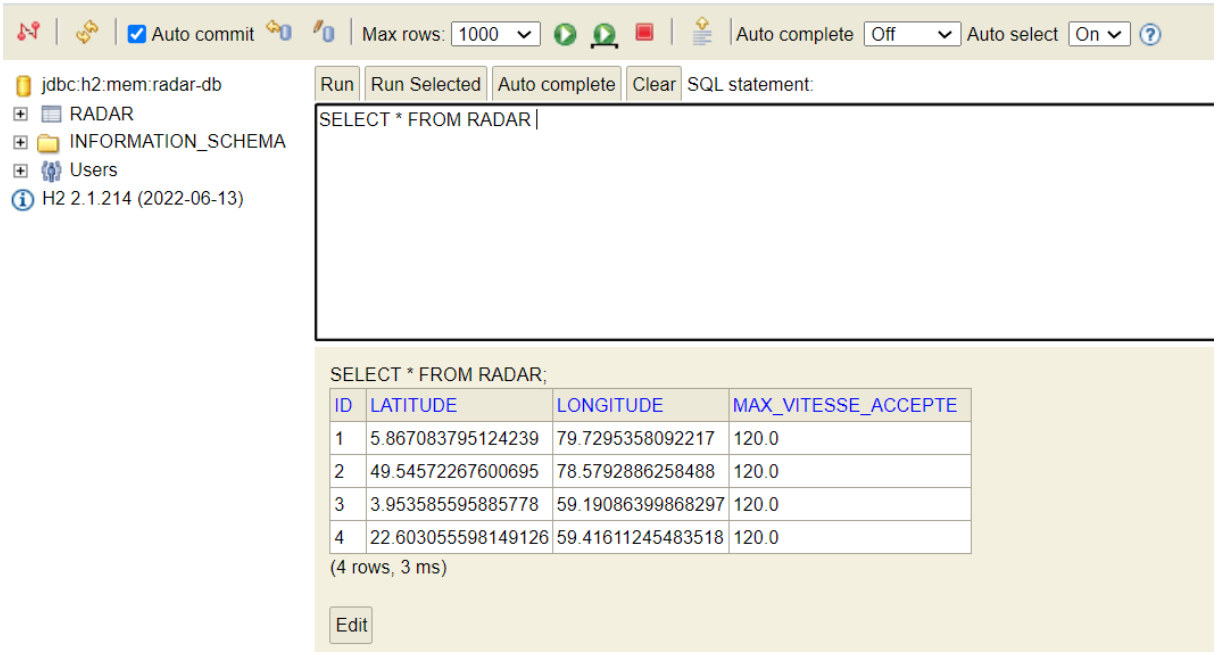
Changer la vitesse du dernier par 100 :



- Méthode DELET



Vérifier dans la base de données H2 console :



7 .Créer un application java qui permet de simuler un radar :

```
package ma.enset.radarsimulation;


import io.grpc.ManagedChannel;
import io.grpc.ManagedChannelBuilder;
import io.grpc.stub.StreamObserver;
import lombok.AllArgsConstructor;
import ma.enset.radarsimulation.feign.VehiculeClientRepository;

import ma.enset.radarsimulation.grpc.stub.RadarGrpcServiceGrpc;
import ma.enset.radarsimulation.grpc.stub.RadarOuterClass;
import ma.enset.radarsimulation.models.Vehicule;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.openfeign.EnableFeignClients;
import org.springframework.context.annotation.Bean;

import java.util.List;
import java.util.Random;


@SpringBootApplication
@AllArgsConstructor
@EnableFeignClients
public class RadarSimulationApplication {

    VehiculeClientRepository vehiculeClientRepository;
    public static void main(String[] args) {
        SpringApplication.run(RadarSimulationApplication.class, args);
    }
```

```
    @Bean
    CommandLineRunner start() {
        return args -> {
            ManagedChannel managedChannel =
ManagedChannelBuilder.forAddress("localhost", 8084)
                    .usePlaintext().build();
            RadarGrpcServiceGrpc.RadarGrpcServiceStub radarServiceStub =
RadarGrpcServiceGrpc.newStub(managedChannel);
            StreamObserver<RadarOuterClass.SaveRadarRequest>
RadarStreamObserver = radarServiceStub.radarControl(new
StreamObserver<RadarOuterClass.DetectOverSpeed>() {
                @Override
                public void onNext(RadarOuterClass.DetectOverSpeed
detectOverSpeed) {

                }

                @Override
                public void onError(Throwable throwable) {

                }

                @Override
                public void onCompleted() {

                }
            });
            int counter = 0;
            List<Vehicule> vehiculeList =
vehiculeClientRepository.getVehicules();


            for (Vehicule v : vehiculeList) {
                int randomNumber = generateNumberInRange(40, 260);
                System.out.println("Vitesse aleatoire: " + (randomNumber));
                RadarOuterClass.SaveRadarRequest simulate =
RadarOuterClass.SaveRadarRequest.newBuilder()
                        .setVitesseMax(120)
                        .setVitesseVehicule(randomNumber)
                        .setMatricule(v.getMatricule())
                        .setRadarId(1L)
                        .build();
                RadarStreamObserver.onNext(simulate);
            }

            RadarStreamObserver.onCompleted();

        };
    }
    public int generateNumberInRange(int min, int max) {
        if (min >= max) {
            throw new IllegalArgumentException("Attention!! 'min' doit être
inférieur à 'max'.");
        }

        Random random = new Random();
        return random.nextInt(max - min + 1) + min;
    }
}
```

Cette application génère aléatoirement des dépassements de vitesses et de les envoyer, via GRPC, au service Radar-Service

6. Mettre en place les services techniques de l'architecture micro-service (Gateway,Eureka Discovery service)

- **Gateway**

```
package ma.enset.gatewayservice;

import org.junit.Test;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.cloud.client.discovery.ReactiveDiscoveryClient;
import
org.springframework.cloud.gateway.discovery.DiscoveryClientRouteDefin
itionLocator;
import
org.springframework.cloud.gateway.discovery.DiscoveryLocatorPropertie
s;
import org.springframework.context.annotation.Bean;

@SpringBootApplication
public class GatewayServiceApplication {

   public static void main(String[] args) {
      SpringApplication.run(GatewayServiceApplication.class, args);
   }
   @Bean
   DiscoveryClientRouteDefinitionLocator
definitionLocator(ReactiveDiscoveryClient rdc,
DiscoveryLocatorProperties properties){
      return new DiscoveryClientRouteDefinitionLocator(rdc,
properties);
   }
   @Test
   void contextLoads() {
   }
}
```

- **Eureka Discovery service**

```
package ma.enset.eurikadescovery;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

@SpringBootApplication
@EnableEurekaServer
public class EurikaDescoveryApplication {

   public static void main(String[] args) {
      SpringApplication.run(EurikaDescoveryApplication.class, args);
   }

}
```

spring Eureka

HOME LAST 1000 SINCE STARTUP

System Status

Environment	test	Current time	2023-05-29T02:34:37 +0100
Data center	default	Uptime	00:04
		Lease expiration enabled	true
		Renews threshold	8
		Renews (last min)	12

DS Replicas

localhost

Instances currently registered with Eureka

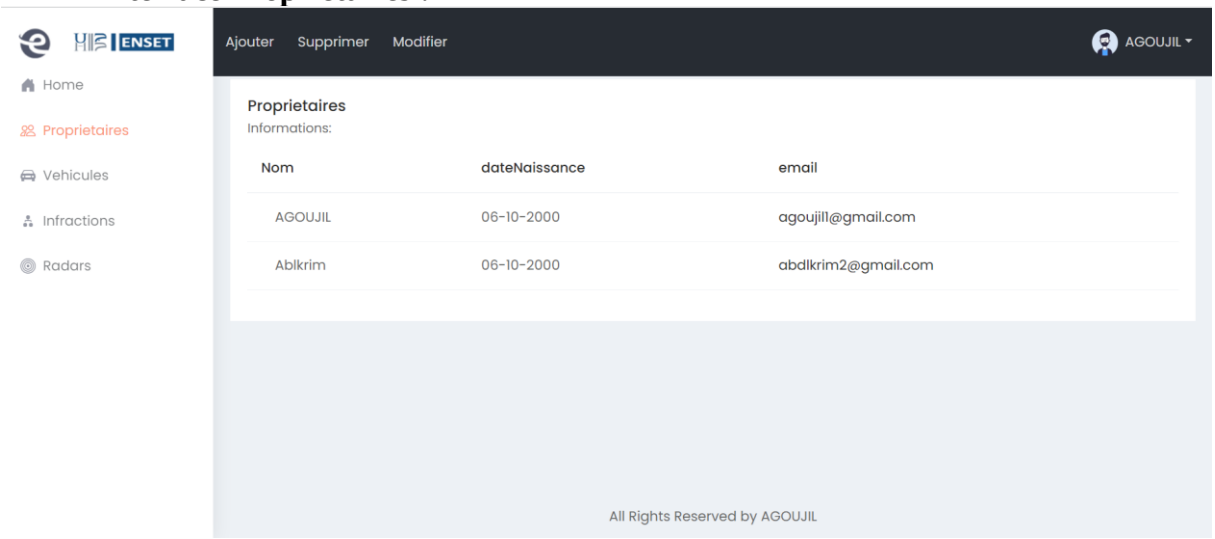
Application	AMIs	Availability Zones	Status
GATEWAY-SERVICE	n/a (1)	(1)	UP (1) - AGOUJIL:gateway-service:8090
IMMATRICULATION-SERVICE	n/a (1)	(1)	UP (1) - AGOUJIL:Immatriculation-service:8080
INFRACTION-SERVICE	n/a (1)	(1)	UP (1) - AGOUJIL:infraction-service:8081
RADAR-SERVICE	n/a (1)	(1)	UP (1) - AGOUJIL:RADAR-SERVICE:8082

General Info

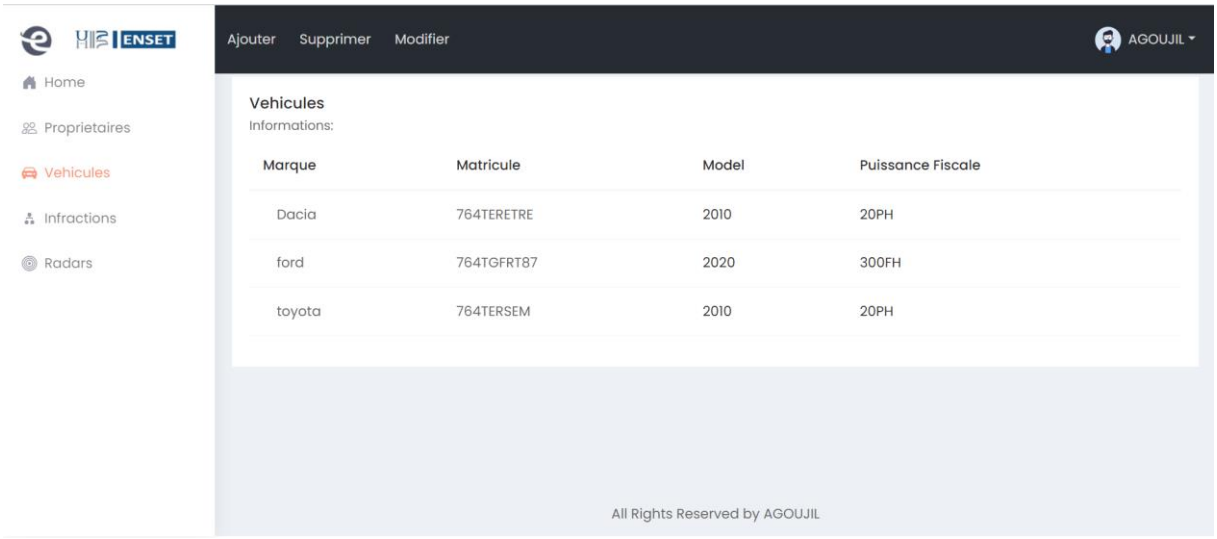
Name	Value
total-avail-memory	88mb
num-of-cpus	4
current-memory-usage	59mb (67%)
server-uptime	00:06
registered-replicas	http://localhost:8761/eureka/

7. Développer votre application Frontend avec Angular :

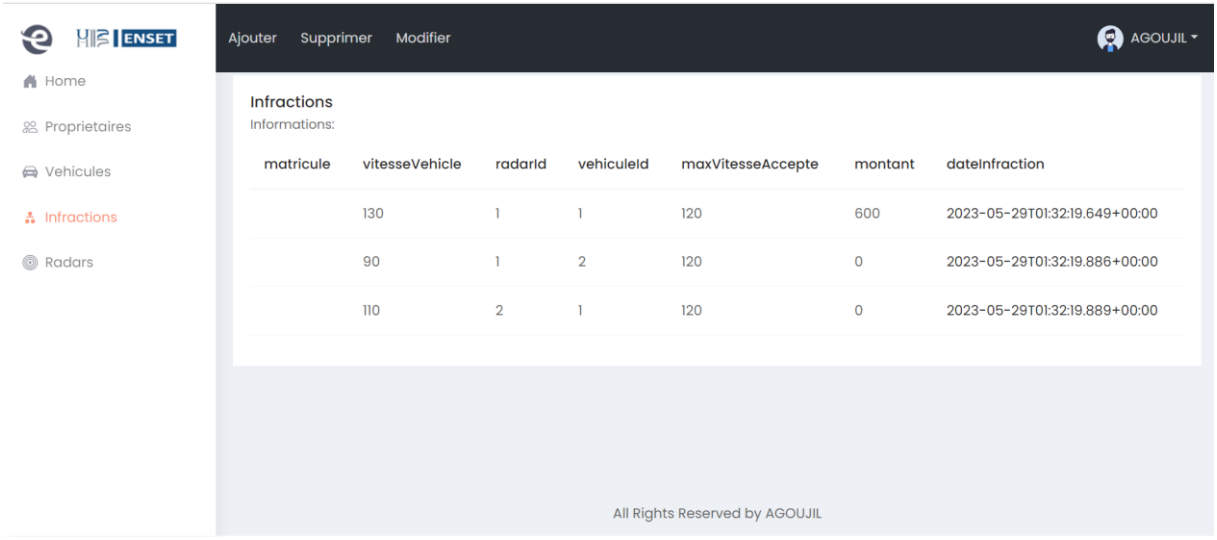
- **Interface Propriétaires :**



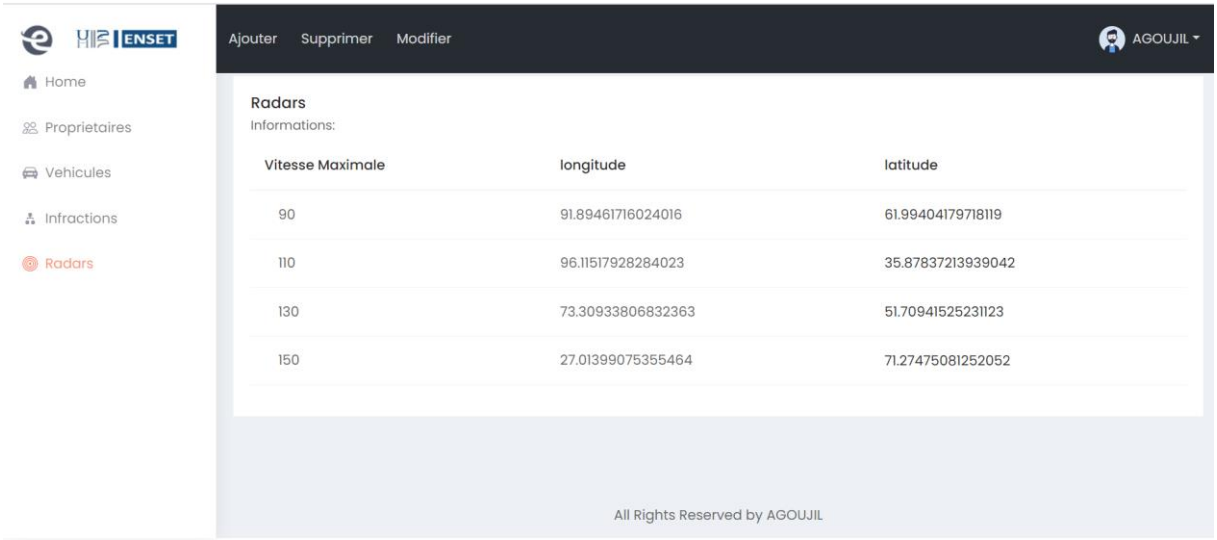
- **Interface Vehicules :**



- **Interface Infractions :**



- **Interface Radars :**



9. Sécuriser votre système avec un système de d'authentification OAuth2 comme Keycloak

10. Écrire un script docker-compose.yml pour le déploiement de ce système distribué dans des conteneurs docker.

Conclusion :

En conclusion, le travail réalisé dans ce projet consistait à créer un système distribué basé sur des micro-services pour gérer les infractions liées aux dépassements de vitesse détectés par des radars automatiques. Nous avons établi une architecture technique, développé des micro-services pour gérer les radars, les véhicules et les infractions, et mis en place différentes méthodes de communication entre ces services (REST, GraphQL, SOAP, GRPC). Nous avons également simulé un radar, sécurisé le système avec OAuth2, développé une interface utilisateur et déployé l'ensemble du système dans des conteneurs Docker. Ce projet nous a permis d'acquérir une expérience pratique en programmation distribuée, en utilisant des technologies et des concepts tels que les micro-services, les web services, les bases de données, la sécurité et le déploiement dans des conteneurs.