



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

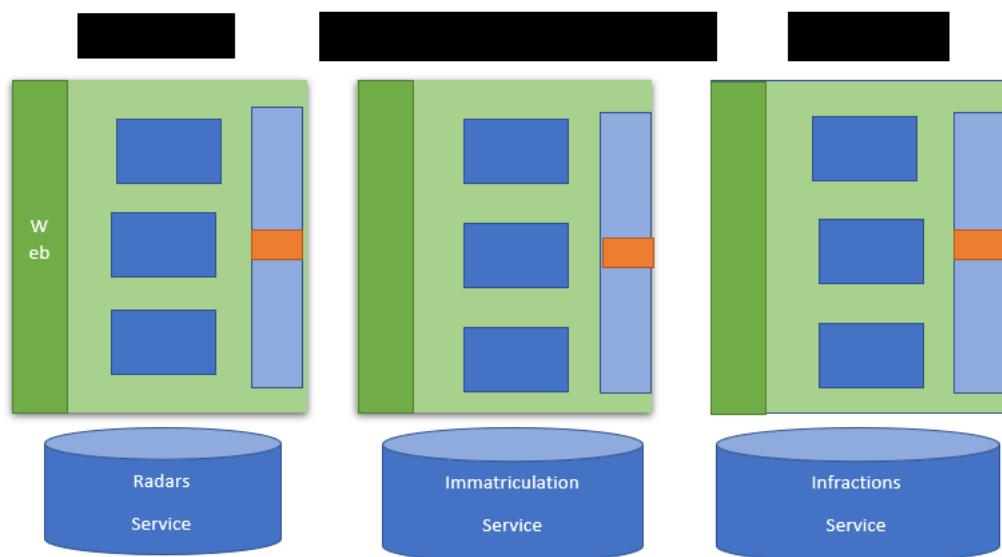
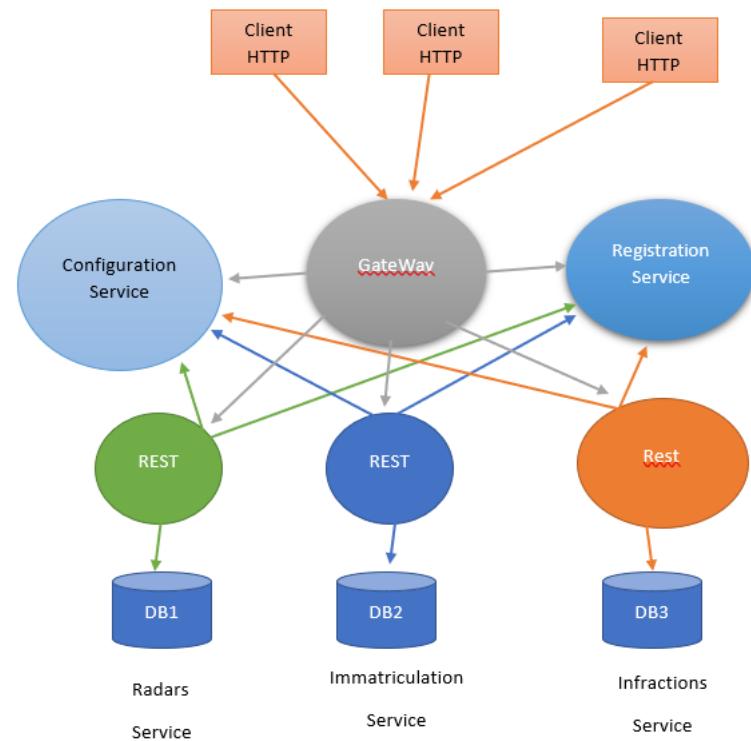
1. Introduction

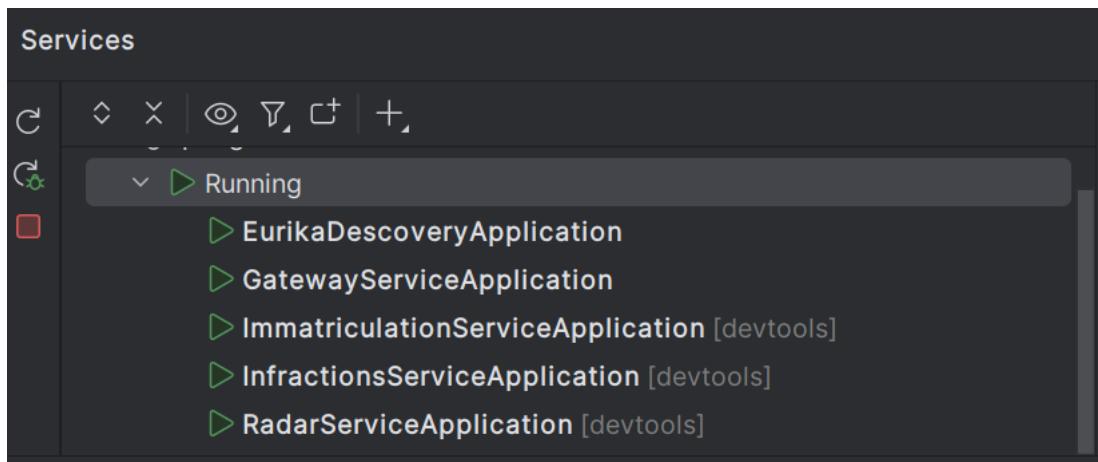
On souhaite créer un système distribué basé sur les micro-services. Cette application devrait permettre de gérer et d'automatiser le processus des infractions concernant des véhicules suites à des dépassement de vitesses détectés par des radars automatiques. Le système se compose de trois micro-services :

- 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.

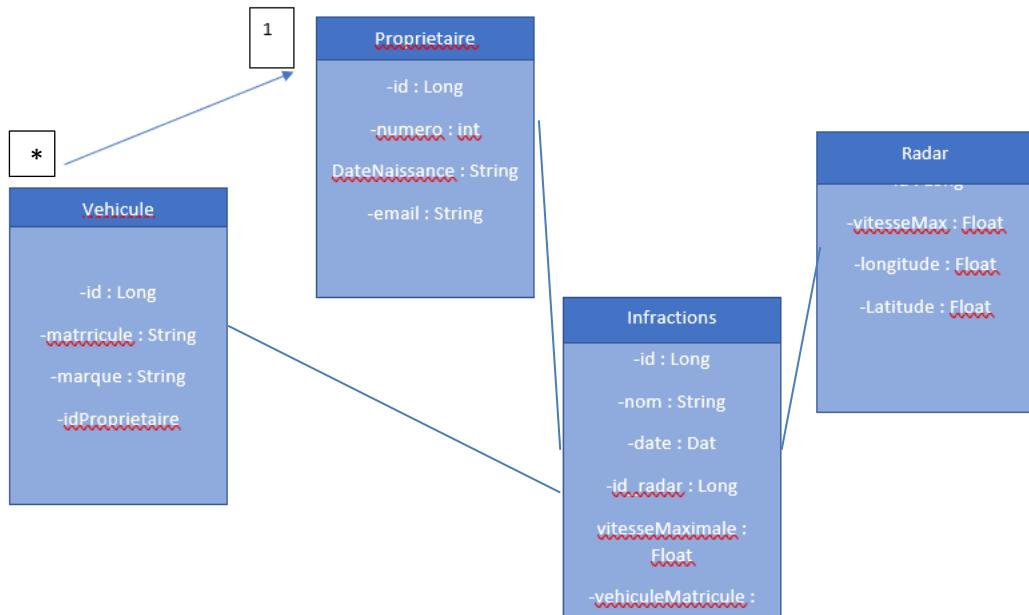
En plus des opérations classiques de consultation et de modifications de données, le système doit permettre de poster un dépassement de vitesse qui va se traduire par une infraction. En plus, il doit permettre à un propriétaire de consulter ses infractions.

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.NoArgsConstructor;
```

```

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

import java.util.List;

@Entity
@Data @NoArgsConstructor @AllArgsConstructor @Builder
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.NoArgsConstructor;
import lombok.Data;
import lombok.NonNull;
import lombok.ToString;

@Entity
@Data @NoArgsConstructor @AllArgsConstructor @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.ensemmatriculation_service.repositories;

import ma.ensemmatriculation_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 véhicules, j'ai associer les deux premiers véhicules et la troisième à le deuxième propriétaire :

```
package ma.ensemmatriculation_service;

import ma.ensemmatriculation_service.entities.Proprietaire;
import ma.ensemmatriculation_service.entities.Vehicule;
import
ma.ensemmatriculation_service.repositories.ProprietaireRepository;
import ma.ensemmatriculation_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", "agoujill@gmail.com", null);
            Proprietaire ablkrim = new Proprietaire(null, "Ablkrim", "06-
10-2000", "abdlkrim2@gmail.com", null);
            proprietaireRepository.save(agoujil);
            proprietaireRepository.save(blkrim);
            Vehicule vehicule1 = new
Vehicule(null, "764TERETRE", "Dacia", "20PH", "2010", agoujil);
            Vehicule vehicule2 = new
Vehicule(null, "764TEREsjdjsTRE", "ford", "300FH", "2020", agoujil);
        
```

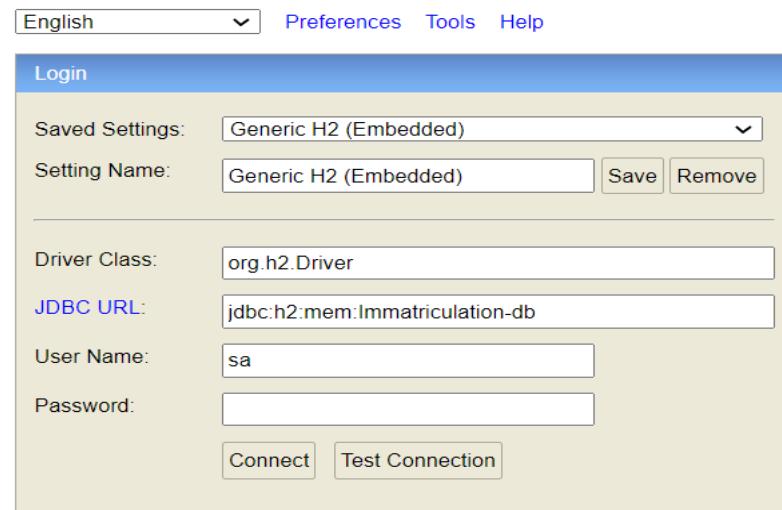
```

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

```

- **Teste l'application :**

Voir les résultats dans la base de données H2 console :



ID	FISCALE_PUISSANCE	MARQUE	MATRECULE	MODEL	PROPRIETAIRE_ID	ID	DATE_NAISSANCE	EMAIL	NOM
1	20PH	Dacia	764TERETRE	2010	1	1	06-10-2000	agoujil1@gmail.com	AGOUJIL
2	300FH	ford	764TEREsdjsTRE	2020	1	1	06-10-2000	agoujil1@gmail.com	AGOUJIL
3	20PH	toyota	764TERETRE	2010	2	2	06-10-2000	abdkrim2@gmail.com	Abdkrim

The screenshot shows the H2 Database Browser interface. On the left, there's a tree view of database objects:

- PROPRIETAIRE** table with columns: ID, DATE_NAISSANCE, EMAIL, NOM.
- VEHICULE** table with columns: ID, FISCALE_PUISSANCE, MARQUE, MATRECULE, MODEL, PROPRIETAIRE_ID.
- INFORMATION_SCHEMA** and **Users** are also listed.
- H2 2.1.214 (2022-06-13)** is at the bottom.

In the main area, a SQL query is run:

```
SELECT * FROM PROPRIETAIRE
```

The results are displayed in a table:

ID	DATE_NAISSANCE	EMAIL	NOM
1	06-10-2000	agoujil1@gmail.com	AGOUJIL
2	06-10-2000	abdkrim2@gmail.com	Abdkrim

(2 rows, 1 ms)

Edit

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

- **Web Service REST**

- **ProprietaireRestController**

```
○ package ma.ensemt.immatriculation_service.web;

import ma.ensemt.immatriculation_service.entities.Proprietaire;
import ma.ensemt.immatriculation_service.repositories.ProprietaireRepository;
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
        proprietairel=proprietaireRepository.findById(Long.valueOf(id))
        .orElseThrow();
        if(proprietaire.getNom()!=null)
        proprietairel.setNom(proprietaire.getNom());
        if(proprietaire.getDateNaissance()!=null)
        proprietairel.setDateNaissance
        (proprietaire.getDateNaissance());
        if(proprietaire.getEmail()!=null)
        proprietairel.setEmail (proprietaire.getEmail());
        if(proprietaire.getVehicules()!=null)
        proprietairel.setVehicules(proprietaire.getVehicules());
        return proprietaireRepository.save(proprietairel);
    }
    @DeleteMapping("/proprietaires/{id}")
    public void delete(@PathVariable String id) {
        proprietaireRepository.deleteById(Long.valueOf(id));
    }
}

```

o VéhiculeRestController

```

o 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.findById(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 :**

- o 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;
    }
}

```

- o **ImmatriculationSoapService :**

- o 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
)
{
        return vehiculeRepository.findById(id);
    }
}

```

```

    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 :**

```

○ package ma.ensemmatriculation_service.web.graphQL;

import lombok.AllArgsConstructor;
import ma.ensemmatriculation_service.entities.Proprietaire;
import
ma.ensemmatriculation_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.enst.immatriculation_service.web.graphQL;

import lombok.AllArgsConstructor;
import ma.enst.immatriculation_service.entities.Vehicule;
import
ma.enst.immatriculation_service.repositories.VehiculeRepository
y;
import
org.springframework.graphql.data.method.annotation.Argument;
import
org.springframework.graphql.data.method.annotation.MutationMapping;
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> vehicles(){
        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) : Proprietaire,
    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**

```

○ 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 (
GetAllVehiclesResponse);
    rpc saveVehicule(SaveVehiculeRequest) returns
(SaveVehiculeResponse);
}
message GetProprietaireRequest{
    int64 id=1;
}
message GetProprietaireResponse{

```

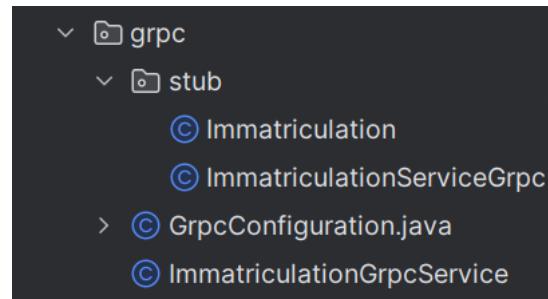
```

        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 :**

```

○ 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.GrpcClientHealthAutoConfiguration.class,
net.devh.boot.grpc.client.autoconfigure.GrpcClientSecurityAutoConfiguration.class,
net.devh.boot.grpc.client.autoconfigure.GrpcClientTraceAutoConfiguration.class,
net.devh.boot.grpc.client.autoconfigure.GrpcDiscoveryClientAutoConfiguration.class,
net.devh.boot.grpc.common.autoconfigure.GrpcCommonCodecAutoConfiguration.class,
net.devh.boot.grpc.common.autoconfigure.GrpcCommonTraceAutoConfiguration.class,
net.devh.boot.grpc.server.autoconfigure.GrpcAdviceAutoConfiguration.class,
net.devh.boot.grpc.server.autoconfigure.GrpcHealthServiceAutoConfiguration.class,
net.devh.boot.grpc.server.autoconfigure.GrpcMetadataConsulConfiguration.class,
net.devh.boot.grpc.server.autoconfigure.GrpcMetadataEurekaConfiguration.class,
net.devh.boot.grpc.server.autoconfigure.GrpcMetadataNacosConfiguration.class,
net.devh.boot.grpc.server.autoconfigure.GrpcMetadataZookeeperConfiguration.class,
net.devh.boot.grpc.server.autoconfigure.GrpcReflectionServiceAutoConfiguration.class,
net.devh.boot.grpc.server.autoconfigure.GrpcServerAutoConfiguration.class,
net.devh.boot.grpc.server.autoconfigure.GrpcServerFactoryAutoConfiguration.class,
net.devh.boot.grpc.server.autoconfigure.GrpcServerMetricAutoConfiguration.class,
net.devh.boot.grpc.server.autoconfigure.GrpcServerSecurityAutoConfiguration.class,
net.devh.boot.grpc.server.autoconfigure.GrpcServerTraceAutoConfiguration.class
})
class GrpcConfig {}

```

- **ImmatriculationGrpcService :**

```

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

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

    @Override
    public void getListVehicule(Immatriculation.GetAllVehiclesRequest
request , StreamObserver<Immatriculation.GetAllVehiclesResponse>
responseObserver) {
        List<Vehicule> vehicles= this.vehiculeRepository.findAll();
        List<Immatriculation.Vehicule>
vehiculeList=vehicles.stream().map(account-
>grpcMapper.fromVehiculeEntity(account)).collect(Collectors.toList());
        Immatriculation.GetAllVehiclesResponse
vehiculesListResponse=Immatriculation.GetAllVehiclesResponse.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

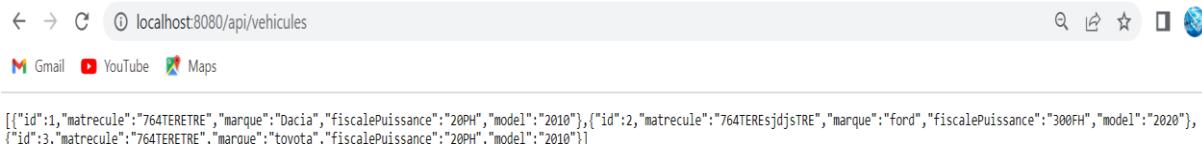
• Tester REST

○ Afficher les propriétaires



```
[{"id":1,"nom":"AGOUJIL","dateNaissance":"06-10-2000","email":"agoujill@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":"Abdkrim","dateNaissance":"06-10-2000","email":"abdkrim2@gmail.com","vehicules": [{"id":3,"matrecule":"764TERETRE","marque":"toyota","fiscalePuissance":"20PH","model":"2010"}]}]
```

○ Afficher les véhicules

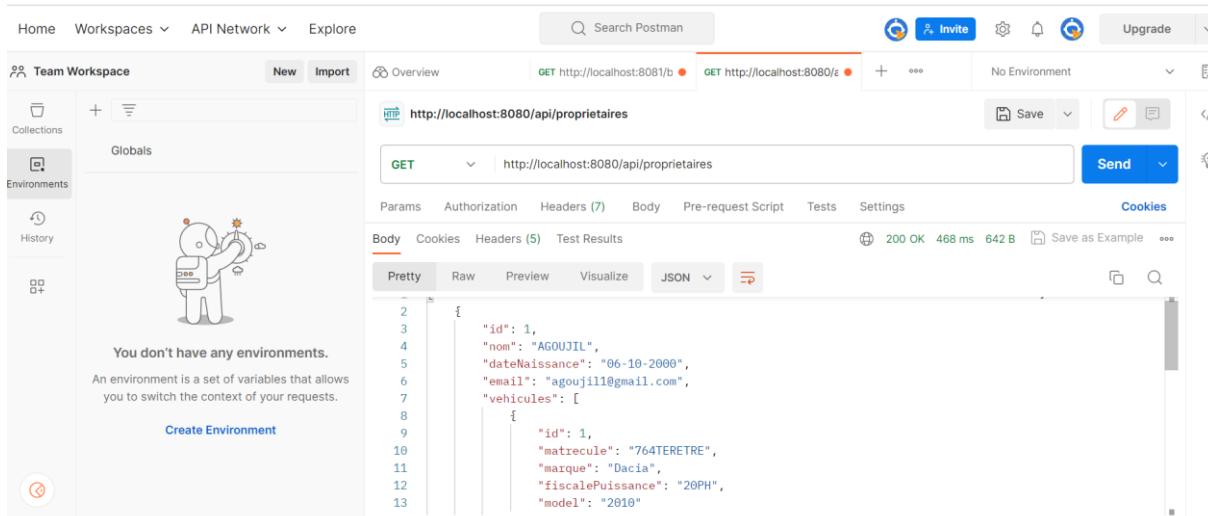


```
[{"id":1,"matrecule":"764TERETRE","marque":"Dacia","fiscalePuissance":"20PH","model":"2010"}, {"id":2,"matrecule":"764TEREsjdjsTRE","marque":"ford","fiscalePuissance":"300FH","model":"2020"}, {"id":3,"matrecule":"764TERETRE","marque":"toyota","fiscalePuissance":"20PH","model":"2010"}]
```

○ Tester les méthodes avec postman

○ Table Proprietaire :

➤ Méthode GET



You don't have any environments.

An environment is a set of variables that allows you to switch the context of your requests.

Create Environment

HTTP <http://localhost:8080/api/proprietaires>

GET http://localhost:8080/b ● GET http://localhost:8080/e ●

Params Authorization Headers (5) Body Pre-request Script Tests Settings Cookies

Pretty Raw Preview Visualize JSON

```
{ "id": 1, "nom": "AGOUJIL", "dateNaissance": "06-10-2000", "email": "agoujill@gmail.com", "vehicules": [ { "id": 1, "matrecule": "764TERETRE", "marque": "Dacia", "fiscalePuissance": "20PH", "model": "2010" } ] }
```

HTTP Overview GET http://localhost:8081/b GET http://localhost:8080/a + ... No Environment

http://localhost:8080/api/proprietaires/2

GET http://localhost:8080/api/proprietaires/2

Save Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Body Cookies Headers (5) Test Results 200 OK 41 ms 354 B Save as Example

Pretty Raw Preview Visualize JSON

```

1 "id": 2,
2 "nom": "Ablkrim",
3 "dateNaissance": "06-10-2000",
4 "email": "abdlkrim2@gmail.com",
5 "vehicules": [
6     {
7         "id": 3,
8         "matrecule": "764TERETRE",
9         "marque": "toyota",
10        "fiscalePuissance": "20PH",
11        "model": "2010"
12    }
13 ]

```

➤ Méthode POST

HTTP Overview GET http://localhost:8081/b POST http://localhost:8080/ + ... No Environment

http://localhost:8080/api/proprietaires

POST http://localhost:8080/api/proprietaires

Save Send

Params Authorization Headers (10) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON

Pretty Raw Preview Visualize JSON

```

1 ...
2 ...
3 ...
4 ...
5 ...

```

Body Cookies Headers (5) Test Results 200 OK 35 ms 268 B Save as Example

```

1 "id": 3,
2 "nom": "abdelaziz",
3 "dateNaissance": "07-05-2005",
4 "email": "a.abdelaziz@gmail.com",
5 "vehicules": null

```

Auto commit Max rows: 1000 Run Run Selected Auto complete Clear SQL statement:

jdbc:h2:mem:Immatriculation-db PROPRIETAIRE VEHICULE INFORMATION_SCHEMA Users H2 2.1.214 (2022-06-13)

SELECT * FROM PROPRIETAIRE;

ID	DATE_NAISANCE	EMAIL	NOM
1	06-10-2000	agoujil1@gmail.com	AGOUJIL
2	06-10-2000	abdlkrim2@gmail.com	Ablkrim
3	07-05-2005	a.abdelaziz@gmail.com	abdelaziz

(3 rows, 2 ms)

➤ Méthode PUT

The screenshot shows the Postman interface. At the top, there are tabs for Overview, GET http://localhost:8081/b, PUT http://localhost:8080/ə, and a new tab. The PUT tab is selected. Below it, the URL is set to http://localhost:8080/api/proprietaires/3. The Body tab is active, showing a JSON payload:

```
1 {  
2   "dateNaissance": "01-02-2003"  
3 }
```

Below the body, the response status is 200 OK with 61 ms latency and 266 B size. The response body is displayed in Pretty format:

```
1 {  
2   "id": 3,  
3   "nom": "abdelaziz",  
4   "dateNaissance": "01-02-2003",  
5   "email": "a.abdelaziz@gmail.com",  
6   "vehicules": []
```

The screenshot shows the MySQL Workbench interface. At the top, there are buttons for Run, Run Selected, Auto complete, Clear, and SQL statement. The SQL statement is:

```
SELECT * FROM PROPRIETAIRE
```

The results pane displays the query:

```
SELECT * FROM PROPRIETAIRE;
```

ID	DATE_NAISSANCE	EMAIL	NOM
1	06-10-2000	agoujil1@gmail.com	AGOUIL
2	06-10-2000	abdkrim2@gmail.com	Abdkrim
3	01-02-2003	a.abdelaziz@gmail.com	abdelaziz

(3 rows, 1 ms)

Edit

➤ Méthode DELETE

Postman screenshot showing a DELETE request to `http://localhost:8080/api/proprietaires/3`. The response is successful (200 OK) with a duration of 21 ms and a size of 123 B.

H2 Database Console screenshot showing the `PROPRIETAIRE` table. The query `SELECT * FROM PROPRIETAIRE` is executed, returning the following results:

ID	DATE_NAISSANCE	EMAIL	NOM
1	06-10-2000	agoujil1@gmail.com	AGOUJIL
2	06-10-2000	abdlkrim2@gmail.com	Abdkrim

- Table Véhicule :

- Méthode GET

The screenshot shows the Postman interface with two GET requests made to `http://localhost:8080/api/vehicles`.

Request 1 (List of vehicles):

- Method: GET
- URL: `http://localhost:8080/api/vehicles`
- Response Body (Pretty Print):

```

3   "id": 1,
4     "matricule": "764TERETRE",
5     "marque": "Dacia",
6     "fiscalePuissance": "20PH",
7     "model": "2010"
8   },
9   {
10    "id": 2,
11    "matricule": "764TEREsjdjsTRE",
12    "marque": "ford",
13    "fiscalePuissance": "300FH",
14    "model": "2020"

```

Request 2 (Single vehicle):

- Method: GET
- URL: `http://localhost:8080/api/vehicles/2`
- Response Body (Pretty Print):

```

1   "id": 2,
2     "matricule": "764TEREsjdjsTRE",
3     "marque": "ford",
4     "fiscalePuissance": "300FH",
5     "model": "2020"

```

➤ Méthode POST

POST http://localhost:8080/api/vehicles

Body (JSON)

```

2   "matrecule": "6544GH",
3   "marque": "Mercedes-Benz",
4   "fiscalePuissance": "42PH",
5   "model": "2022",
6   "nom": "Ablkrim"

```

Body Cookies Headers (5) Test Results

200 OK 113 ms 259 B Save as Example

Max rows: 1000 | Auto complete: Off | Auto select: On | ?

Run Run Selected Auto complete Clear SQL statement:

```
SELECT * FROM VEHICULE
```

SELECT * FROM VEHICULE;

ID	FISCALE_PUISSANCE	MARQUE	MATRECULE	MODEL	PROPRIETAIRE_ID
1	20PH	Dacia	764TERETRE	2010	1
2	300FH	ford	764TEREsjdsTRE	2020	1
3	20PH	toyota	764TERETRE	2010	2
4	42PH	Mercedes-Benz	6544GH	2022	2

(4 rows, 9 ms)

Edit

➤ Méthode POST
Changer le model

Overview GET http://localhost:8081/b PUT http://localhost:8080/a + No Environment

HTTP http://localhost:8080/api/vehicles/4

PUT http://localhost:8080/api/vehicles/4

Save Send

Params Authorization Headers (10) Body **JSON** Pre-request Script Tests Settings Cookies Beautify

```

3   .... "marque": "Mercedes-Benz",
4   .... "fiscalePuissance": "42PH",
5   .... "model": "2023",
6   .... "nom": "Ablkrim"
7

```

Body Cookies Headers (5) Test Results 200 OK 66 ms 259 B Save as Example

Pretty Raw Preview Visualize JSON

```

1 {
2   "id": 4,
3   "matrecule": "6544GH",
4   "marque": "Mercedes-Benz",

```

Max rows: 1000 Auto complete Off Auto select On

Run Run Selected Auto complete Clear SQL statement:

SELECT * FROM VEHICLE

SELECT * FROM VEHICLE;

ID	FISCALE_PUISSANCE	MARQUE	MATRECULE	MODEL	PROPRIETAIRE_ID
1	20PH	Dacia	764TERETRE	2010	1
2	300FH	ford	764TEREsjdjsTRE	2020	1
3	20PH	toyota	764TERETRE	2010	2
4	42PH	Mercedes-Benz	6544GH	2023	2

(4 rows, 1 ms)

Edit

➤ Méthode DELETE
Supprimer le dernier véhicule ajouté :

HTTP <http://localhost:8080/api/vehicles/4>

DELETE | <http://localhost:8080/api/vehicles/4>

Send

Params Authorization Headers (10) Body • Pre-request Script Tests Settings Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body Cookies Headers (4) Test Results

200 OK 41 ms 123 B Save as Example ...

Pretty Raw Preview Visualize Text

1

Max rows: 1000 | Auto complete: Off | Auto select: On | ?

Run Run Selected Auto complete Clear SQL statement:

```
SELECT * FROM VEHICLE;
```

SELECT * FROM VEHICLE;

ID	FISCALE_PUSSANCE	MARQUE	MATRECULE	MODEL	PROPRIETAIRE_ID
1	20PH	Dacia	764TERETRE	2010	1
2	300FH	ford	764TEREsjdjsTRE	2020	1
3	20PH	toyota	764TERETRE	2010	2

(3 rows, 4 ms)

Edit

- **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

← → C i localhost:8080/services/ImmatriculationService?wsdl

[Gmail](#) [YouTube](#) [Maps](#)

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

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

○ En utilisant l'application SoapUI

The screenshot shows the SoapUI interface. On the left, the Navigator panel displays a project named 'soap1' which contains a 'ImmatriculationWSSoapBinding' service. This service is further expanded to show operations like 'proprietaireById', 'proprietaires', 'vehiculeByid', and 'vehicules'. The main central area is titled 'SoapUI Start Page' and features the SoapUI logo and a 'Try ReadyAPI' button. To the right, there are several promotional boxes: one for 'APACHE KAFKA SUPPORT' (mentioning ReadyAPI supports Kafka testing), another for 'Get Started with SoapUI Open Source' (listing 'Add/Import Kafka APIs', 'Create Kafka Tests', and 'Assertion Groups'), and two for 'REST API Sample Project' and 'SOAP API Sample Project'. At the bottom right, there's a link to an 'Amazon Web Service Sample Project'.

○ Afficher les propriétaires :

SoapUI Start Page

Request 1

http://localhost:8080/services/ImmatriculationService

XML Response

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:proprietairesResponse xmlns:ns2="http://soap.web.immatriculation_service.enset">
      <return>
        <dateNaissance>06-10-2000</dateNaissance>
        <email>agoujili@gmail.com</email>
        <id>1</id>
        <nom>AGOUJIL</nom>
      </return>
      <return>
        <dateNaissance>06-10-2000</dateNaissance>
        <email>abdlkrim2@gmail.com</email>
        <id>2</id>
        <nom>Abdkrim</nom>
      </return>
    </ns2:proprietairesResponse>
  </soap:Body>
</soap:Envelope>

```

Headers (6) Attachments (0) SSL Info WSS (0) JMS (0)

response time: 1137ms (465 bytes)

Properties

Property	Value
Name	Request 1

- Afficher un propriétaire par son id

SoapUI Start Page

Request 1

http://localhost:8080/services/ImmatriculationService

XML Request

```

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope">
  <soapenv:Header/>
  <soapenv:Body>
    <soap:proprietaireById>
      <!--Optional:-->
      <id>1</id>
    </soap:proprietaireById>
  </soapenv:Body>
</soapenv:Envelope>

```

XML Response

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:proprietaireByIdResponse xmlns:ns2="http://soap.web.immatriculation_service.enset">
      <return>
        <dateNaissance>06-10-2000</dateNaissance>
        <email>agoujili@gmail.com</email>
        <id>1</id>
        <nom>AGOUJIL</nom>
      </return>
    </ns2:proprietaireByIdResponse>
  </soap:Body>
</soap:Envelope>

```

Headers (6) Attachments (0) SSL Info WSS (0) JMS (0)

response time: 166ms (351 bytes)

Properties

Property	Value
Name	Request 1

- Afficher les véhicules :

SoapUI Start Page

Request 1

http://localhost:8080/services/ImmatriculationService

XML Request

```

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope">
  <soapenv:Header/>
  <soapenv:Body>
    <soap:vehicules/>
  </soapenv:Body>
</soapenv:Envelope>

```

XML Response

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:vehiculesResponse xmlns:ns2="http://soap.web.immatriculation_service.enset">
      <return>
        <fiscalePuissance>20PH</fiscalePuissance>
        <id>1</id>
        <marque>Dacia</marque>
        <matrécule>764TERETRE</matrécule>
        <model>2010K</model>
        <proprietaire>
          <dateNaissance>06-10-2000</dateNaissance>
          <email>agoujili@gmail.com</email>
          <id>1</id>
          <nom>AGOUJIL</nom>
        </proprietaire>
      </return>
      <return>
        <fiscalePuissance>300FW</fiscalePuissance>
        <id>2</id>
        <marque>Ford</marque>
      </return>
    </ns2:vehiculesResponse>
  </soap:Body>
</soap:Envelope>

```

Headers (6) Attachments (0) SSL Info WSS (0) JMS (0)

response time: 105ms (1044 bytes)

Properties

Property	Value
Name	Request 1

- Afficher les véhicules par leur id

The screenshot shows the SoapUI interface with a project named 'soap1' containing a service 'ImmatriculationWSSoapBinding'. A request named 'Request 1' is selected, showing its XML structure. The request XML is:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope">
<soapenv:Header/>
<soapenv:Body>
<soap:vehicleById>
<!--Optional:-->
<id>1</id>
</soap:vehicleById>
</soapenv:Body>
</soapenv:Envelope>
```

The response XML is:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope">
<soap:Body>
<ns2:vehicleByIdResponse xmlns:ns2="http://soap.web.immatriculation">
<return>
<fiscalePuissance>20PH</fiscalePuissance>
<id>1</id>
<marque>dacia</marque>
<matrecule>7647ERETRE</matrecule>
<model>2010</model>
<proprietaire>
<dateNaissance>06-10-2000</dateNaissance>
<email>agoujill1@gmail.com</email>
<id>1</id>
<nom>AGOUJIL</nom>
</proprietaire>
</return>
</ns2:vehicleByIdResponse>
</soap:Body>
</soap:Envelope>
```

Request Properties table:

Property	Value
Name	Request 1

SoapUI log, http log, jetty log, error log, wsrm log, memory log

● Tester GRAPHQL :

← → ⌂ ⓘ localhost:8080/graphiql?path=/graphql

Gmail YouTube Maps

The screenshot shows the GraphiQL interface with a code editor containing a sample GraphQL query:

```
1 # Welcome to GraphiQL
2 #
3 # GraphQL is an in-browser tool for writing, validating,
4 # testing GraphQL queries.
5 #
6 # Type queries into this side of the screen, and you will
7 # typeheads aware of the current GraphQL type schema and
8 # validation errors highlighted within the text.
9 #
10 # GraphQL queries typically start with a "{" character. Lines
11 # with a # are ignored.
12 #
13 # An example GraphQL query might look like:
14 #
15 #   {
16 #     field(arg: "value") {
17 #       subField
18 #     }
19 #   }
20 #
21 # Keyboard shortcuts:
22 #
23 #   Prettify query: Shift-Ctrl-P (or press the prettify button)
24 #
```

Variables and Headers buttons are visible at the bottom.

○ Afficher les propriétaires :

```

1
2 *     query{
3 *         proprietaires{
4 *             nom,
5 *             dateNaissance,
6 *             email
7 *         }
8 *
9 }

```

The right panel shows the response from the GraphQL server:

```

{
  "data": {
    "proprietaires": [
      {
        "nom": "AGOUJIL",
        "dateNaissance": "06-10-2000",
        "email": "agoujil1@gmail.com"
      },
      {
        "nom": "Ablkrim",
        "dateNaissance": "06-10-2000",
        "email": "abdlkrim2@gmail.com"
      }
    ]
  }
}

```

○ Afficher les propriétaires par leur id :

```

1
2 *     query{
3 *         proprietaireRequestById(id:1){
4 *             nom
5 *         }
6 *     }

```

The right panel shows the response from the GraphQL server:

```

{
  "data": {
    "proprietaireRequestById": {
      "nom": "AGOUJIL"
    }
  }
}

```

○ Afficher les véhicules :

```

1
2 *     query{
3 *         vehicules{
4 *             marque,
5 *             model,
6 *             fiscalePuissance,
7 *             proprietaire{
8 *                 nom
9 *             }
10 *            }
11 *       }
12

```

The right panel shows the response from the GraphQL server:

```

{
  "data": {
    "vehicules": [
      {
        "marque": "Dacia",
        "model": "2010",
        "fiscalePuissance": "20PH",
        "proprietaire": {
          "nom": "AGOUJIL"
        }
      },
      {
        "marque": "ford",
        "model": "2020",
        "fiscalePuissance": "300FH",
        "proprietaire": {
          "nom": "AGOUJIL"
        }
      },
      {
        "marque": "toyota",
        "model": "2010",
        "fiscalePuissance": "20PH",
        "proprietaire": {
          "nom": "Ablkrim"
        }
      }
    ]
  }
}

```

○ Afficher les véhicules par leur id :

```

1
2 *     query{
3 *         vehiculeRequestById(id:1){
4 *             marque,
5 *             model
6 *         }
7 *     }

```

The right panel shows the response from the GraphQL server:

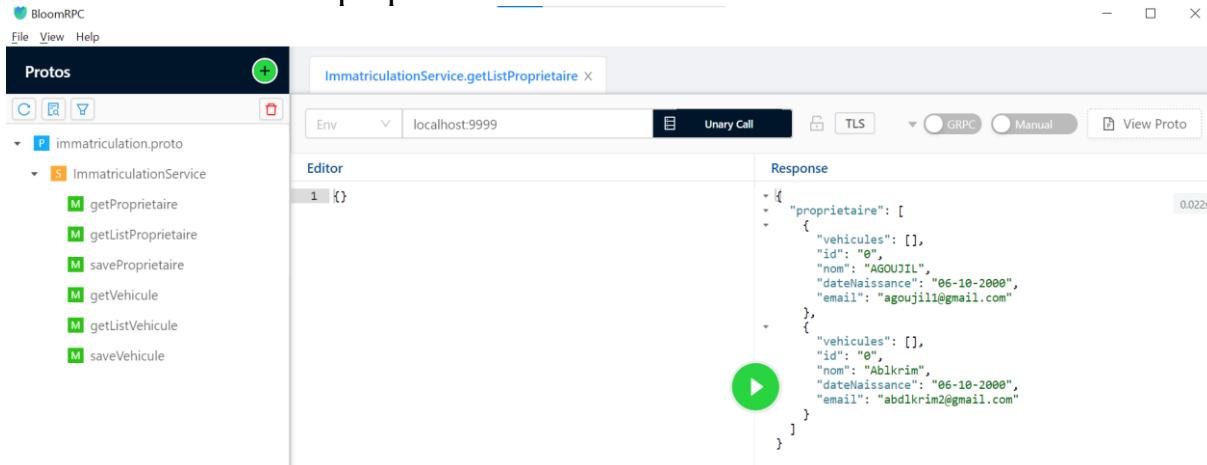
```

{
  "data": {
    "vehiculeRequestById": {
      "marque": "Dacia",
      "model": "2010"
    }
  }
}

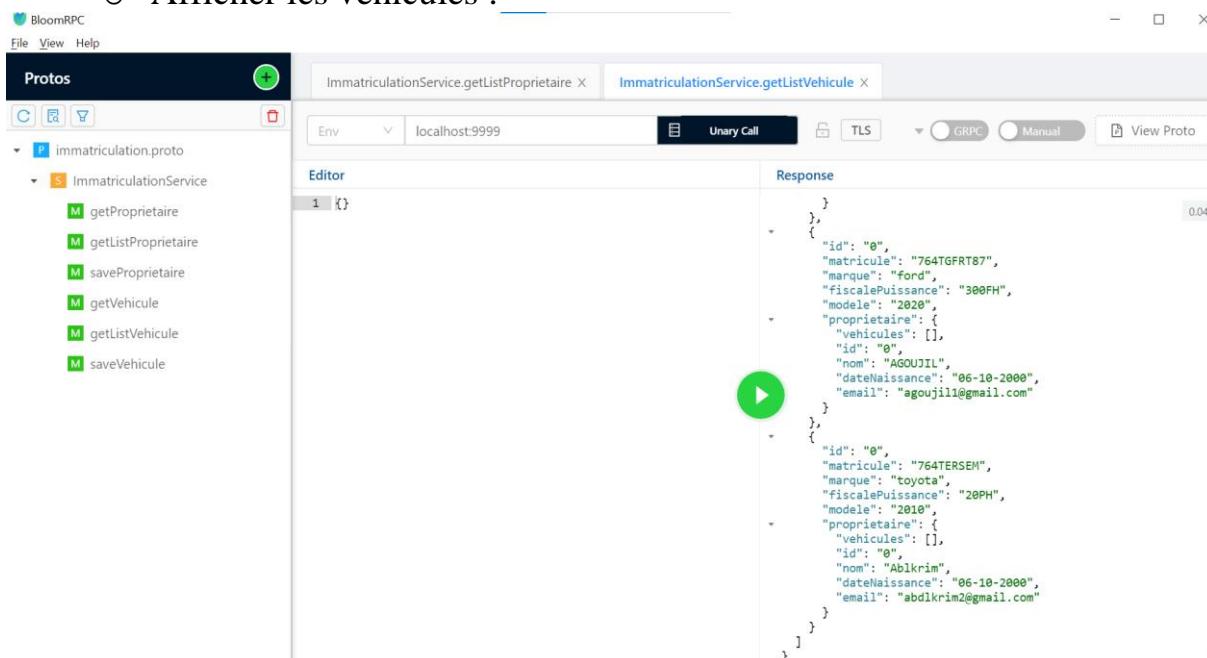
```

- 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.ensemt.infractionsservice.repositories;

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

import java.util.List;

public interface InfractionRepository extends
JpaRepository<Infraction, Long> {
}

```

- **InfractionRestController**

```

• package ma.ensemt.infractionsservice.controller;

import lombok.AllArgsConstructor;
import ma.ensemt.infractionsservice.entities.Infraction;
import ma.ensemt.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

HTTP New Collection / infraction / infractions

GET localhost:8090/INFRACTION-SERVICE/infractions

200 OK 1852 ms 373 B Save as Example

```

1 [
2 {
3 "id": 1,
4 "matricule": null,
5 "vitesseVehicle": 130.0,
6 "dateInfraction": "2023-05-27T09:55:21.416+00:00",
7 "radarId": 1,
8 "vehiculeId": 1,
9 "maxVitesseAccepte": 120.0,
10 "montant": 600.0
11 }
12]

```

  - Méthode POST

POST localhost:8090/INFRACTION-SERVICE/infractions

Body (JSON)

```

1 "id": null,
2 "matricule": "454YTR655R",
3 "vitesseVehicle": 110.0,
4 "dateInfraction": null,
5 "radarId": 2,
6 "vehiculeId": 3,
7 "maxVitesseAccepte": 120.0,
8 "montant": 700.0

```

200 OK 54 ms 352 B

Vérifier dans la base H2 console :

| ID | DATE_INFRACTION         | MATRICULE  | MAX_VITESSE_ACCEPTE | MONTANT | RADAR_ID | VEHICULE_ID | VITESSE_VEHICLE |
|----|-------------------------|------------|---------------------|---------|----------|-------------|-----------------|
| 1  | 2023-05-27 10:55:21.416 | null       | 120.0               | 600.0   | 1        | 1           | 130.0           |
| 2  | null                    | 454YTR655R | 120.0               | 700.0   | 2        | 3           | 110.0           |

- Méthode PUT

Changer le montant de la deuxième infraction :

PUT localhost:8090/INFRACTION-SERVICE/infractions/2

Body (JSON)

```

1 {
2 "montant": 800.0
3 }

```

200 OK 27 ms 352 B

## ○ Méthode DELETE

DELETE localhost:8090/INFRACTION-SERVICE/infractions/2

Body (Text)

```
1
```

200 OK 28 ms 164 B

Vérifier dans la base de données H2 console :

jdbc:h2:mem:infraction-db

INFRACITION

SELECT \* FROM INFRACTION;

| ID | DATE_INFRACTION         | MATRICULE | MAX_VITESSE_ACCEPTE | MONTANT | RADAR_ID | VEHICULE_ID | VITESSE_VEHICLE |
|----|-------------------------|-----------|---------------------|---------|----------|-------------|-----------------|
| 1  | 2023-05-27 10:55:21.416 | null      | 120.0               | 600.0   | 1        | 1           | 130.0           |

## 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.ensemt.radarservice.models;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
import ma.ensemt.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.ensemt.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 @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;

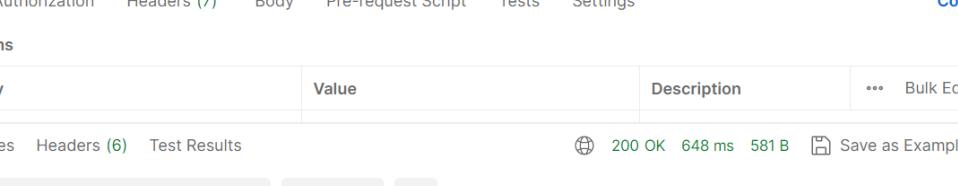
@SpringBootApplication
@EnableFeignClients
public class RadarServiceApplication {

 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



GET localhost:8090/RADAR-SERVICE/radar Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

|  | Key | Value | Description | ... | Bulk Edit |
|--|-----|-------|-------------|-----|-----------|
|  |     |       |             |     |           |

Body Cookies Headers (6) Test Results 200 OK 648 ms 581 B Save as Example ...

Pretty Raw Preview Visualize JSON undo redo

```
2 {
3 "id": 1,
4 "maxVitesseAccepte": 120.0,
5 "longitude": 77.98860335659403,
6 "latitude": 12.4244269313876
7 },
8 {
9 "id": 2,
10 "maxVitesseAccepte": 120.0,
11 "longitude": 68.3568390310971,
12 "latitude": 34.60851429762087
```

#### ○ Méthode POST

The screenshot shows the Postman application interface. At the top, it displays a POST request to the endpoint `localhost:8090/RADAR-SERVICE/radaras`. The "Body" tab is selected, showing a JSON payload:

```
1 {
2 "maxVitesseAccepte": 90.0,
3 "longitude": 72.98860335659403,
4 "latitude": 17.4244269313876
5 }
```

Below the body, the "Body" tab is also selected, showing the raw JSON response received from the server:

```
1 {
2 "id": 5,
3 "maxVitesseAccepte": 90.0,
4 "longitude": 72.98860335659403,
5 "latitude": 17.4244269313876
6 }
```

Vérifier dans la base H2 console :

SELECT \* FROM RADAR;

| ID | LATITUDE          | LONGITUDE          | MAX_VITESSE_ACCEPTE |
|----|-------------------|--------------------|---------------------|
| 1  | 12.4244269313876  | 77.98860335659403  | 120.0               |
| 2  | 34.60851429762087 | 68.3568390310971   | 120.0               |
| 3  | 83.87423501998529 | 16.427771631850618 | 120.0               |
| 4  | 53.48895778830657 | 95.41533949065187  | 120.0               |
| 5  | 17.4244269313876  | 72.98860335659403  | 90.0                |

(5 rows, 3 ms)

[Edit](#)

## ○ Méthode PUT

Changer la vitesse du dernier par 100 :

PUT | localhost:8090/RADAR-SERVICE/radar/5 | Send

Params Authorization Headers (10) Body **JSON** Pre-request Script Tests Settings Cookies Beautify

```

1 {
2 "maxVitesseAccepte": 100.0
3 }

```

Body Cookies Headers (6) Test Results 200 OK 84 ms 297 B Save as Example

Pretty Raw Preview Visualize JSON

```

1 {
2 "id": 5,
3 "maxVitesseAccepte": 100.0,
4 "longitude": 72.98860335659403,
5 "latitude": 17.4244269313876
6 }

```

## ○ Méthode DELETE

DELETE | localhost:8090/RADAR-SERVICE/radar/5 | Send

Params Authorization Headers (10) Body **JSON** Pre-request Script Tests Settings Cookies </> Beautify

```

1 {
2 "maxVitesseAccepte": 100.0
3 }

```

Body Cookies Headers (5) Test Results 200 OK 34 ms 164 B Save as Example

Pretty Raw Preview Visualize Text

```

1

```

Vérifier dans la base de données H2 console :

The screenshot shows a MySQL Workbench interface. On the left, there's a tree view with nodes: 'jdbc:h2:mem:radar-db', 'RADAR', 'INFORMATION\_SCHEMA', 'Users', and 'H2 2.1.214 (2022-06-13)'. The main area has tabs: 'Run', 'Run Selected', 'Auto complete', 'Clear', and 'SQL statement'. Below these tabs, the SQL statement 'SELECT \* FROM RADAR' is entered. The results show a table with four rows:

| ID | LATITUDE           | LONGITUDE         | MAX_VITESSE_ACCEPTE |
|----|--------------------|-------------------|---------------------|
| 1  | 5.867083795124239  | 79.7295358092217  | 120.0               |
| 2  | 49.54572267600695  | 78.5792886258488  | 120.0               |
| 3  | 3.953585595885778  | 59.19086399868297 | 120.0               |
| 4  | 22.603055598149126 | 59.41611245483518 | 120.0               |

(4 rows, 3 ms)

**Edit**

## 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.VehicleClientRepository;

import ma.enset.radarsimulation.grpc.stub.RadarGrpcServiceGrpc;
import ma.enset.radarsimulation.grpc.stub.RadarOuterClass;
import ma.enset.radarsimulation.models.Vehicle;
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 {

 VehicleClientRepository vehicleClientRepository;
 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.DiscoveryClientRouteDefinitionLocator;
import
org.springframework.cloud.gateway.discovery.DiscoveryLocatorProperties;
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 EurikaDiscoveryApplication {

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

### 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

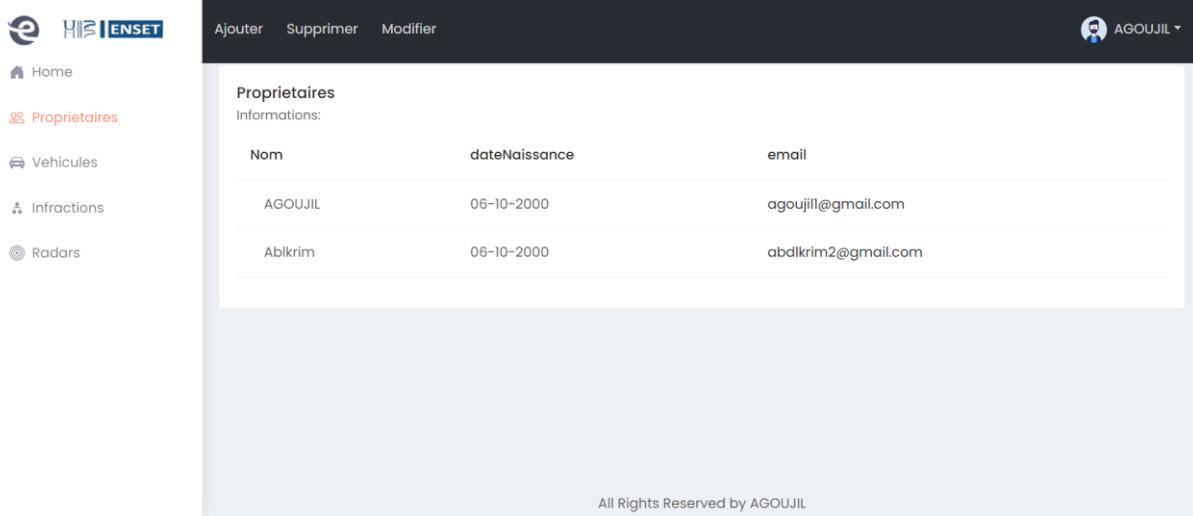
| Application             | AMIs    | Availability Zones | Status                                                        |
|-------------------------|---------|--------------------|---------------------------------------------------------------|
| GATEWAY-SERVICE         | n/a (1) | (1)                | UP (1) - <a href="#">AGOUJIL:gateway-service:8090</a>         |
| IMMATRICULATION-SERVICE | n/a (1) | (1)                | UP (1) - <a href="#">AGOUJIL:immatriculation-service:8080</a> |
| INFRACTION-SERVICE      | n/a (1) | (1)                | UP (1) - <a href="#">AGOUJIL:infraction-service:8081</a>      |
| RADAR-SERVICE           | n/a (1) | (1)                | UP (1) - <a href="#">AGOUJIL:RADAR-SERVICE:8082</a>           |

### General Info

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

## 7. Développer votre application Frontend avec Angular :

- Interface Propriétaires :



The screenshot shows the 'Propriétaires' (Owners) page of an Angular application. The page has a header with a logo and navigation links for Home, Propriétaires, Vehicles, Infractions, and Radars. Below the header, there's a table with columns for Nom (Name), dateNaissance (Birth Date), and email. Two rows are present in the table:

| Nom     | dateNaissance | email               |
|---------|---------------|---------------------|
| AGOUJIL | 06-10-2000    | agoujil@gmail.com   |
| Abikrim | 06-10-2000    | abdikrim2@gmail.com |

At the bottom of the page, it says "All Rights Reserved by AGOUJIL".

- **Interface Vehicles :**

The screenshot shows a web application interface for managing vehicles. The top navigation bar includes links for Ajouter, Supprimer, and Modifier. On the left, a sidebar menu lists Home, Propriétaires, Vehicles (which is selected and highlighted in orange), Infractions, and Radars. The main content area is titled "Vehicles" and contains a table with the following data:

| Marque | Matricule  | Model | Puissance Fiscale |
|--------|------------|-------|-------------------|
| Dacia  | 764TERETRE | 2010  | 20PH              |
| ford   | 764TGFRT87 | 2020  | 300FH             |
| toyota | 764TERSEM  | 2010  | 20PH              |

All Rights Reserved by AGOUJIL.

- **Interface Infractions :**

The screenshot shows a web application interface for managing infractions. The top navigation bar includes links for Ajouter, Supprimer, and Modifier. On the left, a sidebar menu lists Home, Propriétaires, Vehicles (selected and highlighted in orange), Infractions (selected and highlighted in orange), and Radars. The main content area is titled "Infractions" and contains a table with the following data:

| matricule | vitesseVehicle | radarId | vehicleId | maxVitesseAccepte | montant                       | dateInfraction |
|-----------|----------------|---------|-----------|-------------------|-------------------------------|----------------|
| 130       | 1              | 1       | 120       | 600               | 2023-05-29T01:32:19.649+00:00 |                |
| 90        | 1              | 2       | 120       | 0                 | 2023-05-29T01:32:19.886+00:00 |                |
| 110       | 2              | 1       | 120       | 0                 | 2023-05-29T01:32:19.889+00:00 |                |

All Rights Reserved by AGOUJIL.

- Interface Radars :

The screenshot shows a web application interface for managing radar data. The top navigation bar includes a logo for 'HIS ENSET', a user icon, and the name 'AGOUJIL'. The main menu on the left lists 'Home', 'Propriétaires', 'Véhicules', 'Infractions', and 'Radars' (which is highlighted in red). The central content area has a header 'Ajouter Supprimer Modifier' and a sub-header 'Informations: Radars'. A table displays four rows of radar data with columns for 'Vitesse Maximale', 'longitude', and 'latitude'. The data is as follows:

| Vitesse Maximale | longitude         | latitude          |
|------------------|-------------------|-------------------|
| 90               | 91.89461716024016 | 61.99404179718119 |
| 110              | 96.11517928284023 | 35.87837213939042 |
| 130              | 73.30933806832363 | 51.70941525231123 |
| 150              | 27.01399075355464 | 71.27475081252052 |

All Rights Reserved by AGOUJIL.

## 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.