

## Systèmes Distribué

### COMPTE RENDU DE L'EXAMEN



Pr. Mohammed YOUSSEFI.

Réalisé par :

- FAJRI WALID

Master Intelligence Artificiel et Analyse des données.

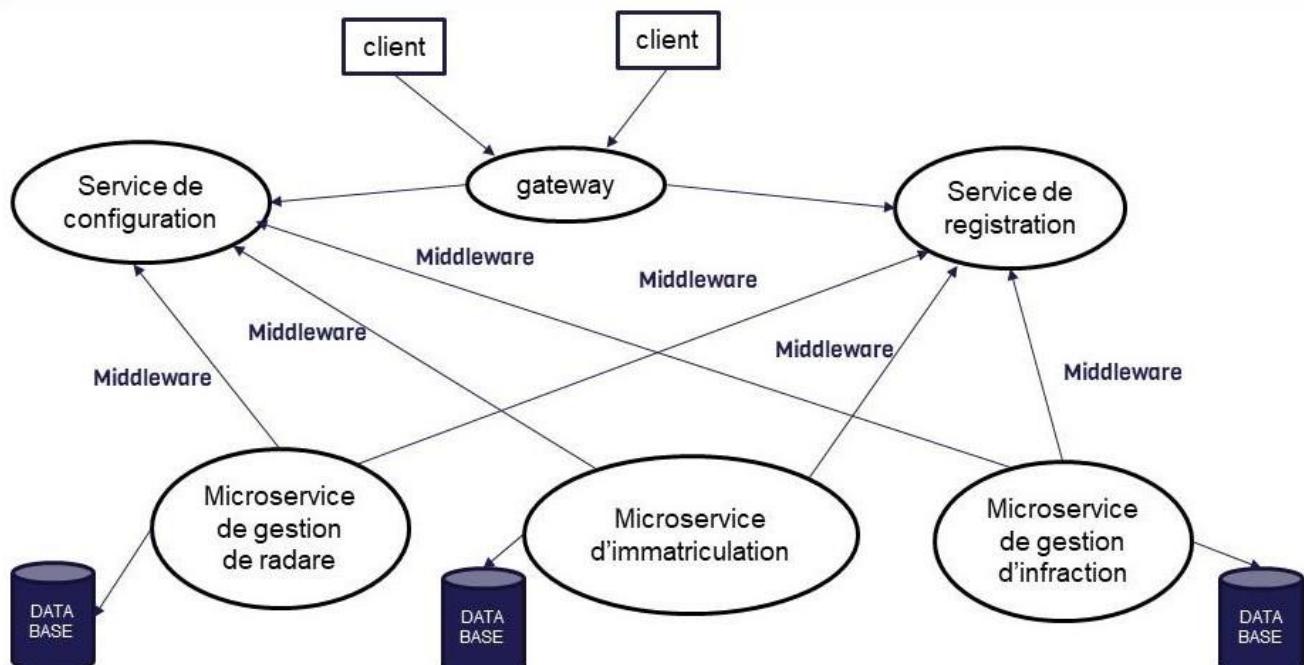
## Objectif :

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 suite à des dépassements de vitesses détectés par des radars automatiques. Le système se compose de trois micro-services :

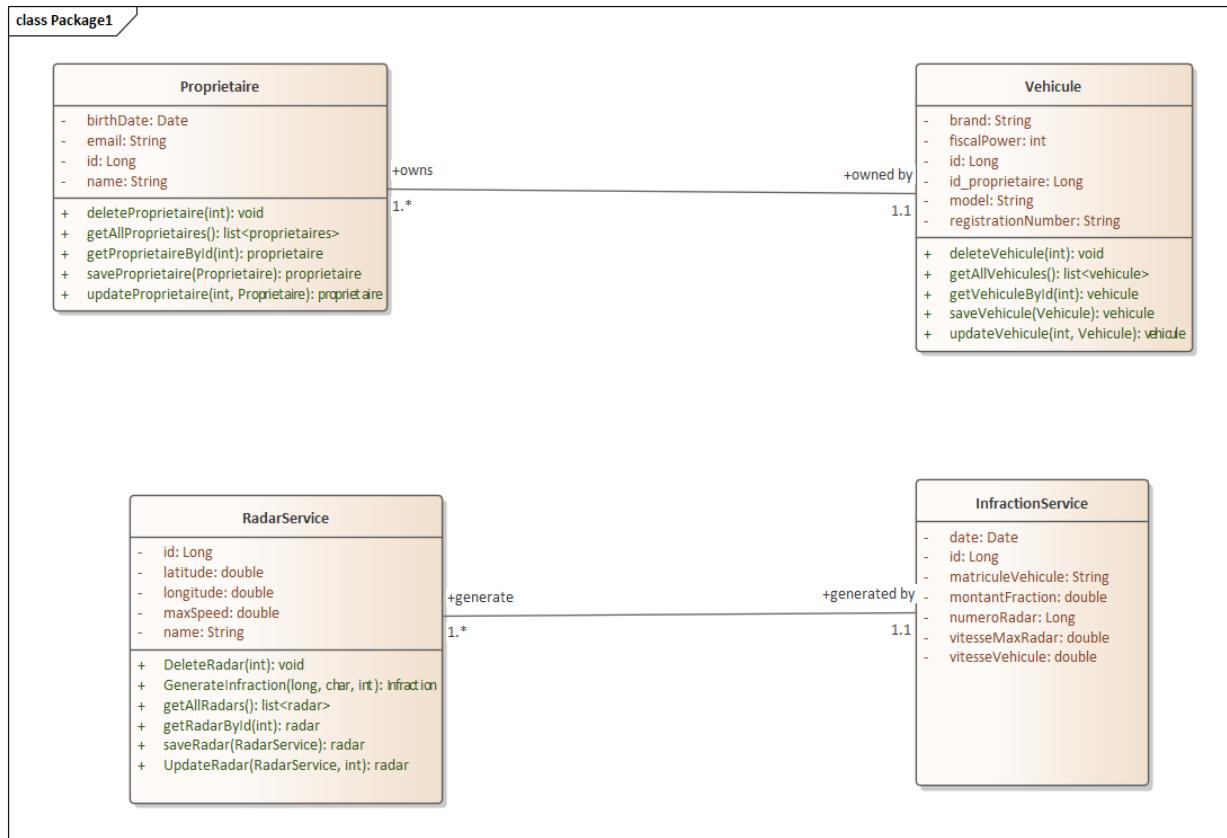
- Le micro-service qui permet de gérer les radars. Chaque radar est défini par son id, sa vitesse maximale, des coordonnées : Longitude et Latitude.
- Le micro-service d'immatriculation qui permet de gérer des véhicules appartenant des propriétaires. Chaque véhicule appartient à un seul propriétaire. Un propriétaire est défini par son id, son nom, sa date de naissance, son email et son email. Un véhicule est défini par son id, son numéro de matricule, sa marque, sa puissance fiscale et son modèle.
- Le micro-service qui permet de gérer les infractions. Chaque infraction est définie par son id, sa date, le numéro du radar qui a détecté le dépassement, le matricule du véhicule, la vitesse du véhicule, la vitesse maximale du radar et le montant de l'infraction.

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

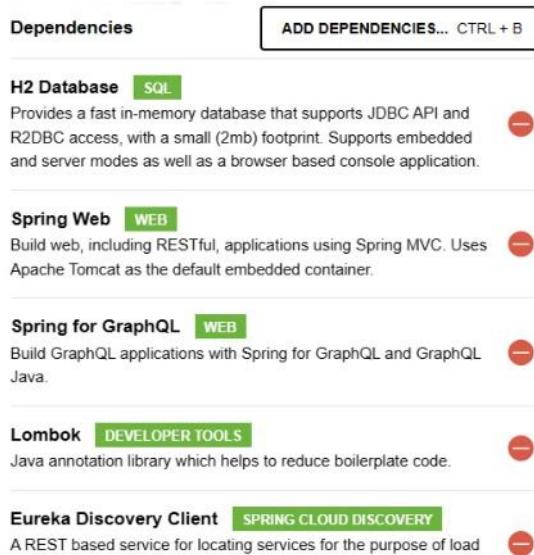
### 1. Établir une architecture technique du projet :

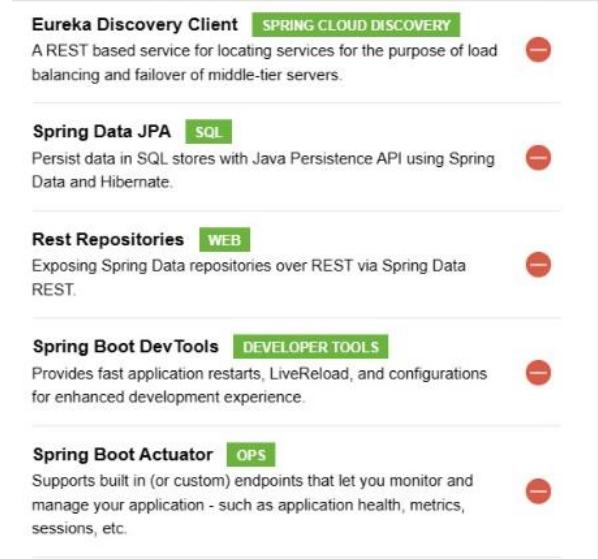


## 2. Établir un diagramme de classe global du projet :



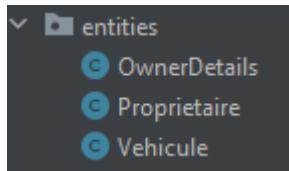
On crée un projet spring à partir de spring starter : en ajoutant les dépendances lombok, spring bootdevtools,spring web , spring data jpa, h2 database et spring for graphql .





## a. Entités JPA et Interface JpaRepository basées sur Spring data :

On commence par la création des entités Propriétaire et Vehicule dans un package Entités :



L'entité Véhicule :

```
package com.example.immatriculationservice.entities;

import ...

@Entity
@Data @NoArgsConstructor @AllArgsConstructor @Builder
public class Vehicule {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String registrationNumber;
    private String brand;
    private String model;
    private int fiscalPower;
}
```

L'entité Propriétaire :

```
package com.example.immatriculationservice.entities;

import ...  
  
@Entity  
@Data @NoArgsConstructor @AllArgsConstructor @Builder
public class Proprietaire {  
    1 usage  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Long id;  
    1 usage  
    private String name;  
    1 usage  
    private LocalDate birthDate;  
    1 usage  
    private String email;  
    1 usage  
    @OneToOne(mappedBy = "proprietaire", fetch = FetchType.EAGER)  
    @JsonIgnore  
    private Collection<Vehicule> vehicules;  
    //
```

L'entité Owner Details :

```
package com.example.immatriculationservice.entities;

import ...  
  
1 usage
@Data
@NoArgsConstructor
@AllArgsConstructor
public class OwnerDetails {  
    no usages
    private String name;
    no usages
    private String email;
```

Maintenant dans un package Repositories, on va créer les deux interfaces VehiculeRepository et ProprietaireRepository. :



VehiculeRepository :

```
package com.example.immatriculationservice.repositories;

import ...

9 usages
@RepositoryRestResource
public interface VehiculeRepository extends JpaRepository<Vehicule, Long> {
    1 usage
    public Collection<Vehicule> getVehiculeByProprietaireId(long id);
    //Vehicule findByMatricule(String matricule);

    //used in gRPC
    1 usage
    @Query("SELECT v FROM Vehicule v WHERE v.id LIKE ?1")
    Vehicule selectById(Long id);

    1 usage
    @Query("SELECT v.proprietaire.id FROM Vehicule v WHERE v.id LIKE ?1")
    long selectProprietaireByIdVehicule(long id);

    1 usage
```

ProprietaireRepository :

```
package com.example.immatriculationservice.repositories;

import com.example.immatriculationservice.entities.Proprietaire;
import com.example.immatriculationservice.entities.Vehicule;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.rest.core.annotation.RepositoryRestResource;

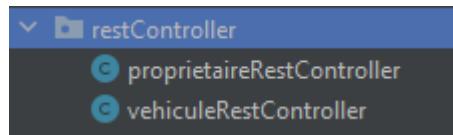
12 usages
@RepositoryRestResource
public interface ProprietaireRepository extends JpaRepository<Proprietaire, Long> {

}
```

### 3. Les 4 web services REST, GraphQL, SOAP et GRPC:

#### b. Pour le web service REST :

Dans un Package appeler Web, nous allons créer les classes suivantes :



#### VehiculeRestController :

```
package com.example.immatriculationservice.web.restController;

import ...

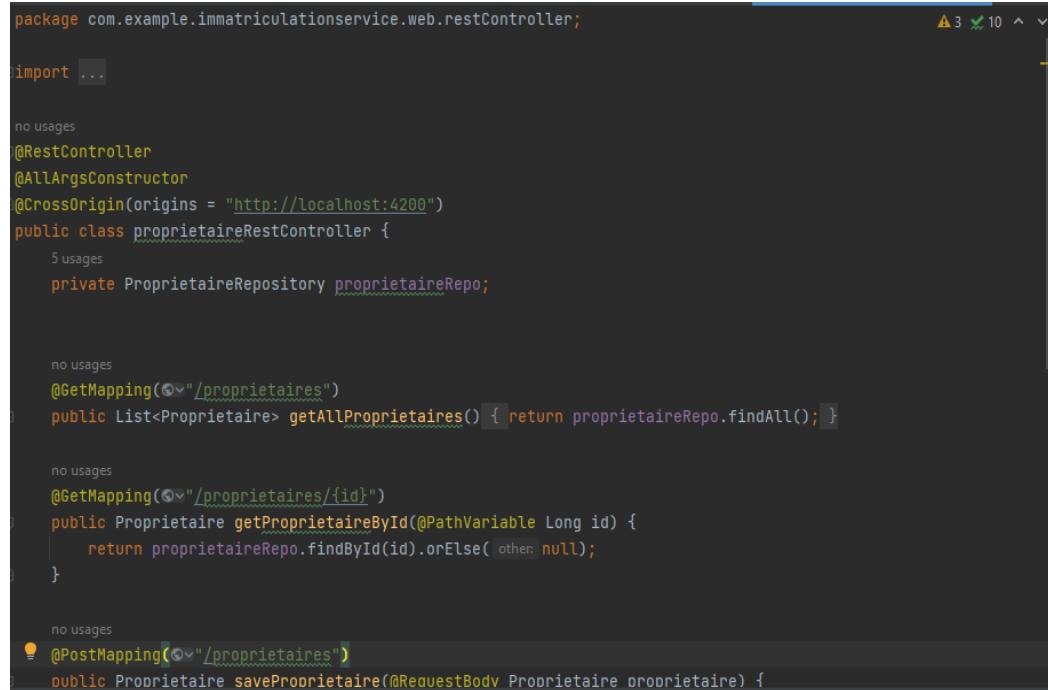
no usages
@RestController
@AllArgsConstructor
@CrossOrigin(origins = "*", allowedHeaders = "*", methods = {RequestMethod.GET, RequestMethod.POST, RequestMethod
public class vehiculeRestController {
    13 usages
    private VehiculeRepository vehiculeRepo;
    1 usage
    private ProprietaireRepository proprietaireRepository;

    no usages
    @GetMapping("/vehicules")
    public List<Vehicule> getAllVehicules() { return vehiculeRepo.findAll(); }

    no usages
    @GetMapping("/{id}")
    public Vehicule getVehiculeById(@PathVariable Long id) { return vehiculeRepo.findById(id).orElse(null); }

    no usages
    @GetMapping("/vehicules/matricules/{matricule}") // hadii
    public ResponseEntity<Vehicule> findByMatricule(@PathVariable String matricule){
```

ProprietaireRestController :



```
package com.example.immatriculationservice.web.restController;

import ...

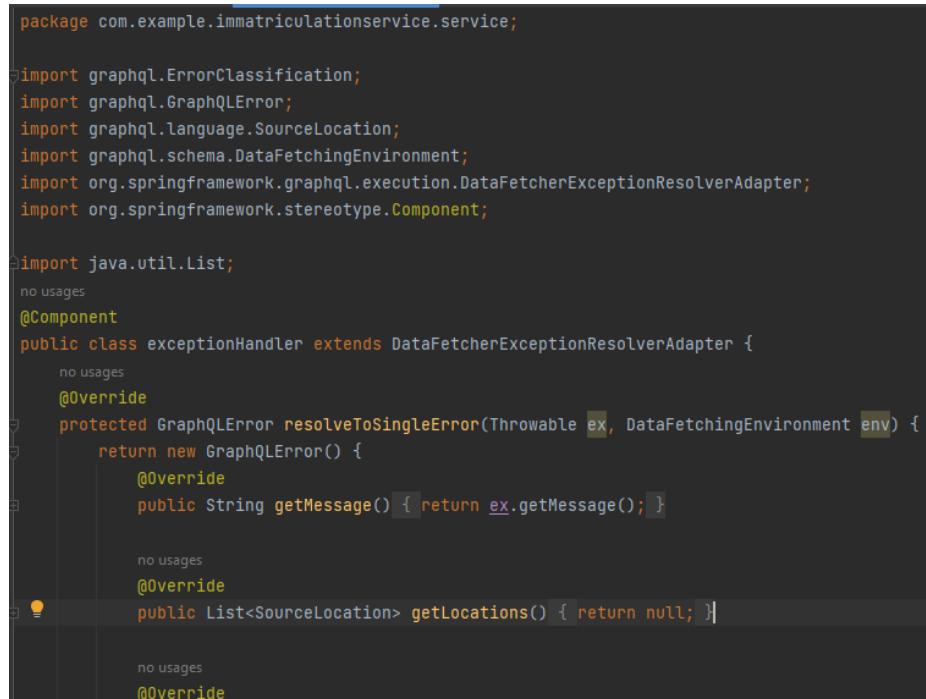
no usages
@RestController
@AllArgsConstructor
@CrossOrigin(origins = "http://localhost:4200")
public class proprietaireRestController {
    5 usages
    private ProprietaireRepository proprietaireRepo;

    no usages
    @GetMapping("proprietaires")
    public List<Proprietaire> getAllProprietaires() { return proprietaireRepo.findAll(); }

    no usages
    @GetMapping("proprietaires/{id}")
    public Proprietaire getProprietaireById(@PathVariable Long id) {
        return proprietaireRepo.findById(id).orElse( other: null);
    }

    no usages
    @PostMapping("proprietaires")
    public Proprietaire saveProprietaire(@RequestBody Proprietaire proprietaire) {
```

Dans un Package Appelé Service, on va créer la classe exceptionHandler:



```
package com.example.immatriculationservice.service;

import graphql.ErrorClassification;
import graphql.GraphQLError;
import graphql.language.SourceLocation;
import graphql.schema.DataFetchingEnvironment;
import org.springframework.graphql.execution.DataFetcherExceptionResolverAdapter;
import org.springframework.stereotype.Component;

import java.util.List;
no usages
@Component
public class exceptionHandler extends DataFetcherExceptionResolverAdapter {
    no usages
    @Override
    protected GraphQLError resolveToSingleError(Throwable ex, DataFetchingEnvironment env) {
        return new GraphQLError() {
            @Override
            public String getMessage() { return ex.getMessage(); }

            no usages
            @Override
            public List<SourceLocation> getLocations() { return null; }

            no usages
            @Override
```

Dans un Package Appelé DTO, on va créer la classe suivante :

```
package com.example.immatriculationservice.dto;

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

6 usages
@Data
@NoArgsConstructor@AllArgsConstructor
public class ProprietaireRequest{
    no usages
    private String name;
    no usages
    private String email;
}
```

Pour faciliter le transfert des objets entre les différents couches du projet, nous allons créer les mapper pour chacune de nos entités dans un package appelé mappers:

ProprietaireMapper :

```
package com.example.immatriculationservice.mapper;

import com.example.immatriculationservice.dto.ProprietaireRequest;
import com.example.immatriculationservice.entities.Proprietaire;
import org.modelmapper.ModelMapper;
import org.springframework.stereotype.Component;

4 usages
@Component
public class ProprietaireMapper {
    no usages
    private ModelMapper modelMapper;
    2 usages
    public Proprietaire from(ProprietaireRequest request){
        Proprietaire proprietaire=new Proprietaire();
        proprietaire.setName(request.getName());
        proprietaire.setEmail(request.getEmail());
        return proprietaire;
    }
}
```

## c. Pour le web service GraphQL :

VehiculeGraphQlController :

```
package com.example.immatriculationservice.web.graphqlController;

import ...

//@Controller
no usages
@AllArgsConstructor
public class VehiculeGraphQLController {
    6 usages
    private VehiculeRepository vehicleRepo;

    no usages
    @QueryMapping
    public List<Vehicule> getAllVehicles() { return vehicleRepo.findAll(); }

    no usages
    @QueryMapping
    public Vehicule getVehiculeById(@Argument Long id) { return vehicleRepo.findById(id).orElse( other: null); }

    no usages
    @MutationMapping
    public Vehicule saveVehicule(@Argument Vehicule vehicle) { return vehicleRepo.save(vehicle); }

    no usages
```

ProprietaireGraphQlController :

```
package com.example.immatriculationservice.web.graphqlController;

import ...

//@Controller
no usages
@AllArgsConstructor
public class ProprietaireGraphQLController {
    1 usage
    private ProprietaireMapper proprietaireMapper;
    6 usages
    private ProprietaireRepository proprietaireRepo;

    //retourner une liste des proprietaires
    no usages
    @QueryMapping
    public List<Proprietaire> getAllProprietaires() { return proprietaireRepo.findAll(); }

    no usages
    @QueryMapping
    public Proprietaire getProprietaireById(@Argument Long id) {
        Proprietaire proprietaire=proprietaireRepo.findById(id).orElse( other: null);
        if(proprietaire==null) throw new RuntimeException(String.format("Owner %s not found",id));
        return proprietaire;
    }
```

Shéma de QraphQL :

```
type Query {
    getAllVehicules: [Vehicule],
    getVehiculeById(id : Int): Vehicule,
    getAllProprietaires: [Proprietaire],
    getProprietaireById(id : Int): Proprietaire
}

type Mutation {
    saveVehicule(vehicule : VehiculeType):Vehicule
    updateVehicule(vehicule : VehiculeType, id : Int):Vehicule
    deleteVehicule(id : Int):String

    saveProprietaire(proprietaire : ProprietaireType):Proprietaire
    updateProprietaire(proprietaire : ProprietaireType, id : Int):Proprietaire
    deleteProprietaire(id : Int):String
}

type Vehicule {
    id : Int,
    numMatricule : String,
    marque : String,
    puissanceFiscale : String,
```

## d. Pour le web service GRPC :

Dans un package appelé GRPC, on va créer deux packages appelés Stubs et SERVICE :

Pour le package Service : on crée la classe suivante ;

La classe ProprietaireGrpc :

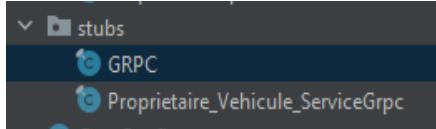
```
package com.example.immatriculationservice.grpc.services;

import ...

//@GrpcService
no usages
@AllArgsConstructor
public class ProprietaireGrpc extends Proprietaire_Vehicule_ServiceGrpc.Proprietaire_Vehicule_ServiceImplBase
    3 usages
    private final ProprietaireRepository proprietaireRepository;
    4 usages
    private final VehiculeRepository vehiculeRepository;

    1 usage
    @Override
    public void getProprietaireById(GRPC.Proprietaireid request, StreamObserver<GRPC.Proprietaire> responseObserver) {
        Long id=request.getId();
        Proprietaire proprietaire=proprietaireRepository.findById(id).orElseThrow( exceptionSupplier: null);
        if (proprietaire!=null) {
            GRPC.Proprietaire proprietaireResponse = GRPC.Proprietaire.newBuilder()
                .setId(Math.toIntExact(proprietaire.getId()))
                .setNom(proprietaire.getName())
                .setEmail(proprietaire.getEmail())
                .build();
            responseObserver.onNext(proprietaireResponse);
        }
    }
}
```

Dans le package Stub :



Dans GRPC :

```
.../.../
```

```
package com.example.immatriculationservice.grpc.stubs;

669 usages
public final class GRPC {
    no usages
    private GRPC() {}
    1 usage
    public static void registerAllExtensions(
        com.google.protobuf.ExtensionRegistryLite registry) {
    }

    no usages
    public static void registerAllExtensions(
        com.google.protobuf.ExtensionRegistry registry) {
        registerAllExtensions(
            com.google.protobuf.ExtensionRegistryLite) registry);
    }

    11 usages 2 implementations
    public interface VehiculeOrBuilder extends
        // @protoc_insertion_point(interface_extends:Vehicule)
        com.google.protobuf.MessageOrBuilder {

    }

    /**
     * ...
     */
}
```

Dans GRPC :

```
package com.example.immatriculationservice.grpc.stubs;

import static io.grpc.MethodDescriptor.generateFullMethodName;
import static io.grpc.stub.ClientCalls.asyncBidiStreamingCall;
import static io.grpc.stub.ClientCalls.asyncClientStreamingCall;
import static io.grpc.stub.ClientCalls.asyncServerStreamingCall;
import static io.grpc.stub.ClientCalls.asyncUnaryCall;
import static io.grpc.stub.ClientCalls.blockingServerStreamingCall;
import static io.grpc.stub.ClientCalls.blockingUnaryCall;
import static io.grpc.stub.ClientCalls.futureUnaryCall;
import static io.grpc.stub.ServerCalls.asyncBidiStreamingCall;
import static io.grpc.stub.ServerCalls.asyncClientStreamingCall;
import static io.grpc.stub.ServerCalls.asyncServerStreamingCall;
import static io.grpc.stub.ServerCalls.asyncUnaryCall;
import static io.grpc.stub.ServerCalls.asyncUnimplementedStreamingCall;
import static io.grpc.stub.ServerCalls.asyncUnimplementedUnaryCall;

23 usages
@javax.annotation.Generated(
    value = "by gRPC proto compiler (version 1.15.0)",
    comments = "Source: gRPC.proto")
public final class Proprietaire_Vehicule_ServiceGrpc {
```

```

private Proprietaire_Vehicule_ServiceGrpc() {}

▲ 8 ▲

6 usages
public static final String SERVICE_NAME = "Proprietaire_Vehicule_Service";

// Static method descriptors that strictly reflect the proto.
3 usages
private static volatile io.grpc.MethodDescriptor<com.example.immatriculationservice.grpc.stubs.GRPC.P
    com.example.immatriculationservice.grpc.stubs.GRPC.Proprietaire> getGetProprietaireByIdMethod;

6 usages
@io.grpc.stub.annotations.RpcMethod(
    fullMethodName = SERVICE_NAME + '/' + "getProprietaireById",
    requestType = com.example.immatriculationservice.grpc.stubs.GRPC.Proprietaireid.class,
    responseType = com.example.immatriculationservice.grpc.stubs.GRPC.Proprietaire.class,
    methodType = io.grpc.MethodDescriptor.MethodType.UNARY)
public static io.grpc.MethodDescriptor<com.example.immatriculationservice.grpc.stubs.GRPC.Proprietair
    com.example.immatriculationservice.grpc.stubs.GRPC.Proprietaire> getGetProprietaireByIdMethod() {
    io.grpc.MethodDescriptor<com.example.immatriculationservice.grpc.stubs.GRPC.Proprietaireid, com.ex
    if ((getGetProprietaireByIdMethod = Proprietaire_Vehicule_ServiceGrpc.getGetProprietaireByIdMethod)
        synchronized (Proprietaire_Vehicule_ServiceGrpc.class) {
            if ((getGetProprietaireByIdMethod = Proprietaire_Vehicule_ServiceGrpc.getGetProprietaireByIdMet
                Proprietaire_Vehicule_ServiceGrpc.getGetProprietaireByIdMethod = getGetProprietaireByIdMethod
                io.grpc.MethodDescriptor.<~>newBuilder()
                    .setType(io.grpc.MethodDescriptor.MethodType.UNARY)
                    .setFullMethodName(generateFullMethodName(

```

### GrpcConfig :

```

package com.example.immatriculationservice.grpc;

import ...

no usages
@Configuration
@ImportAutoConfiguration({
    net.devh.boot.grpc.client.autoconfigure.GrpcClientAutoConfiguration.class,
    net.devh.boot.grpc.client.autoconfigure.GrpcClientMetricAutoConfiguration.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.

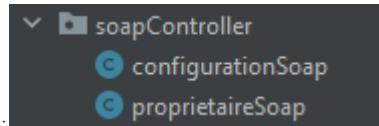
```

Grpc.proto :

```
syntax = "proto3";
option java_package="com.example.immatriculationservice.grpc.stubs"; // ou put file copiler
message Vehicule{
    int32 id=1;
    string nom=2;
    string matricule=3;
    string marque =4;
    int32 puissFiscal=5;
    string modele=6;
    Proprietaire Proprietaire=7;
}
message VehiculeId{
    int32 id=1;
}
message Proprietaire{
    int32 id=1;
    string nom=2;
    string birthDate=3;
    string email=4;
}
message Proprietaireid{
    int64 id=1;
}
message GetVehiculeListRequest{}
```

## e. Pour le web service SOAP :

Dans le package Web on a créé autre package qui s'appelle soap Controller qui contient deux Classes :



ConfigurationSoap :

```
package com.example.immatriculationservice.web.soapController;

import ...

//@Configuration
no usages
@AllArgsConstructor
public class configurationSoap {
    1 usage
    private Bus bus;
    1 usage
    private proprietaireSoap proprietaireSoap;
    no usages
    @Bean
    public EndpointImpl endpoint(){
        EndpointImpl endpoint=new EndpointImpl(bus,proprietaireSoap);
        endpoint.publish( addr: "/ProprietaireService");
        return endpoint;
    }
}
```

ProprietaireSoap :

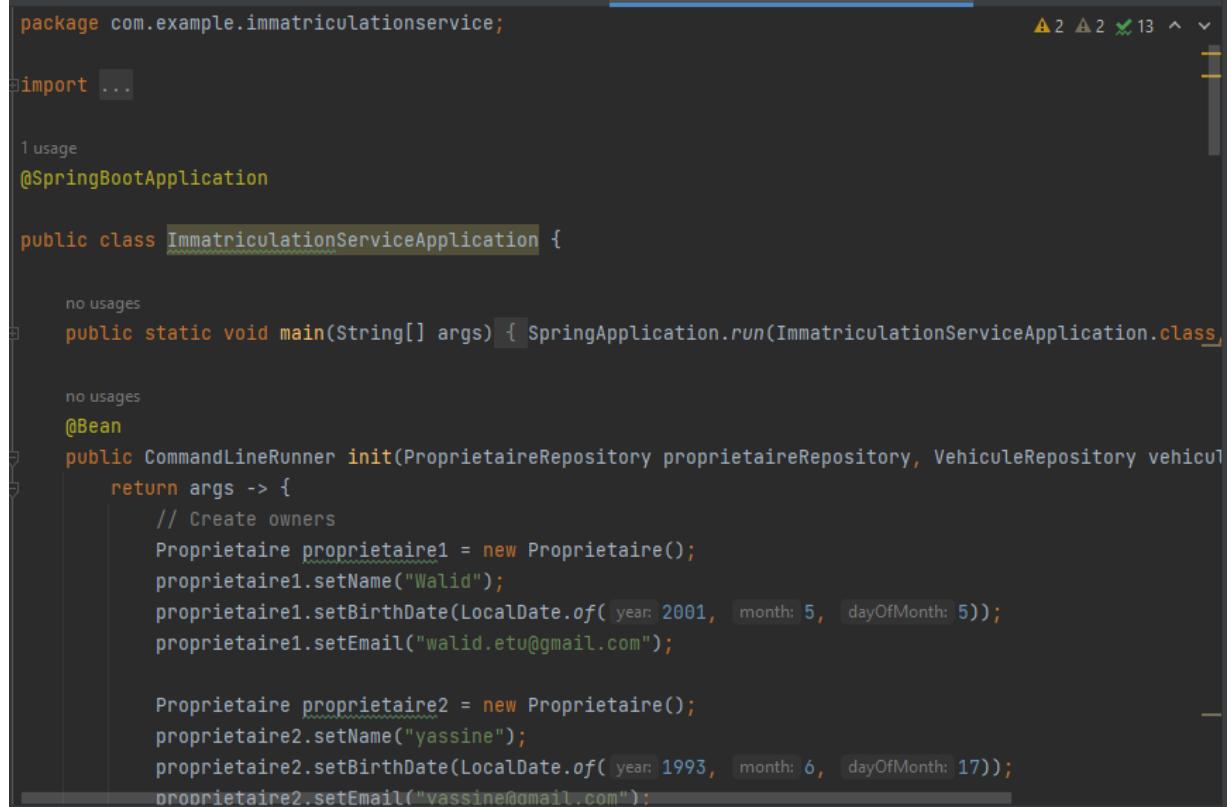
```
package com.example.immatriculationservice.web.soapController;

import ...

//@Component
1 usage
@AllArgsConstructor
@CrossOrigin(origins = "http://localhost:4200/")
public class proprietaireSoap {
    3 usages
    private ProprietaireRepository proprietaireRepository;
    1 usage
    private ProprietaireMapper mapper;
    no usages
    @WebMethod
    public List<Proprietaire> proprietaireList() { return proprietaireRepository.findAll(); }
    //return owner
    no usages
    @WebMethod
    public Proprietaire proprietaireById(@WebParam(name = "id") Long id){
        return proprietaireRepository.findById(id).orElse( other: null);
    }
    //savemethod
}
```

## f. Tester les 4 web services:

On commence par créer les objets :



```
package com.example.immatriculationservice;

import ...

1 usage
@SpringBootApplication

public class ImmatriculationServiceApplication {

    no usages
    public static void main(String[] args) { SpringApplication.run(ImmatriculationServiceApplication.class, args); }

    no usages
    @Bean
    public CommandLineRunner init(ProprietaireRepository proprietaireRepository, VehiculeRepository vehiculeRepository) {
        return args -> {
            // Create owners
            Proprietaire proprietaire1 = new Proprietaire();
            proprietaire1.setName("Walid");
            proprietaire1.setBirthDate(LocalDate.of(2001, 5, 5));
            proprietaire1.setEmail("walid.etu@gmail.com");

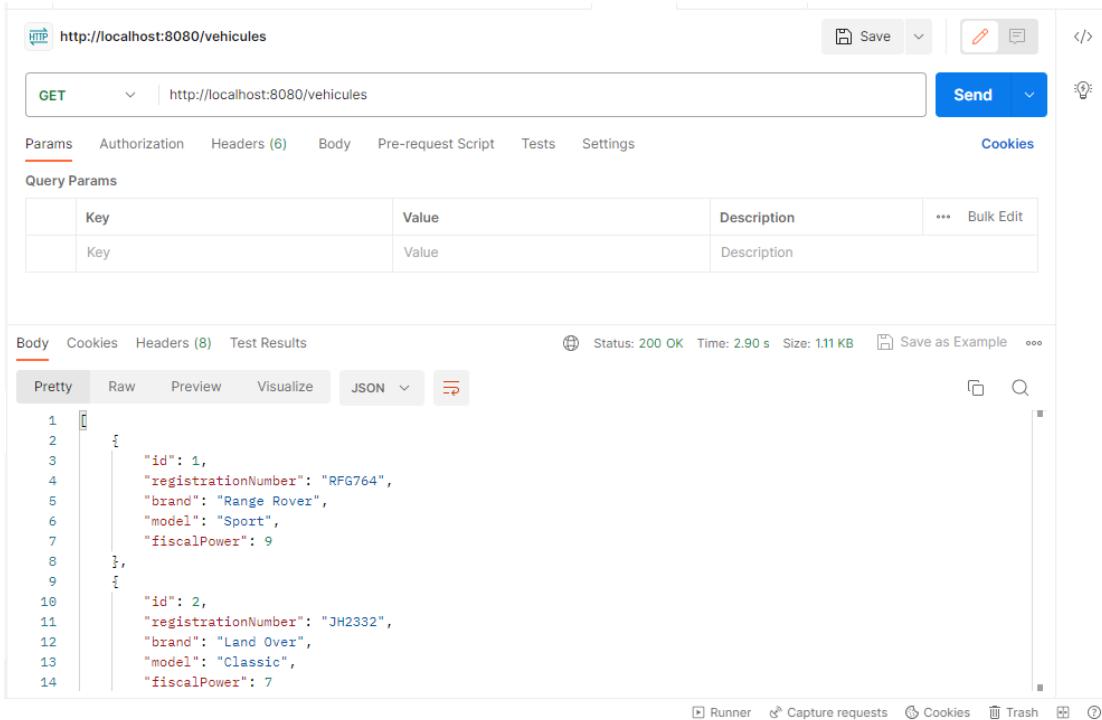
            Proprietaire proprietaire2 = new Proprietaire();
            proprietaire2.setName("yassine");
            proprietaire2.setBirthDate(LocalDate.of(1993, 6, 17));
            proprietaire2.setEmail("vassine@gmail.com");
        };
    }
}
```

Et on configure les propriétés de l'application :

```
spring.port=8080
spring.application.name=immatriculation-service
spring.datasource.url=jdbc:h2:mem:immatriculation-db
spring.h2.console.enabled=true
#spring.graphql.graphiql.enabled=true
#grpc.server.port=8888
#grpc.server.address=*
spring.cloud.discovery.enabled=true
```

Maintenant on lance le micro service, et on teste les méthodes qu'on a développé à travers l'outil Postman : **pour le Rest :**

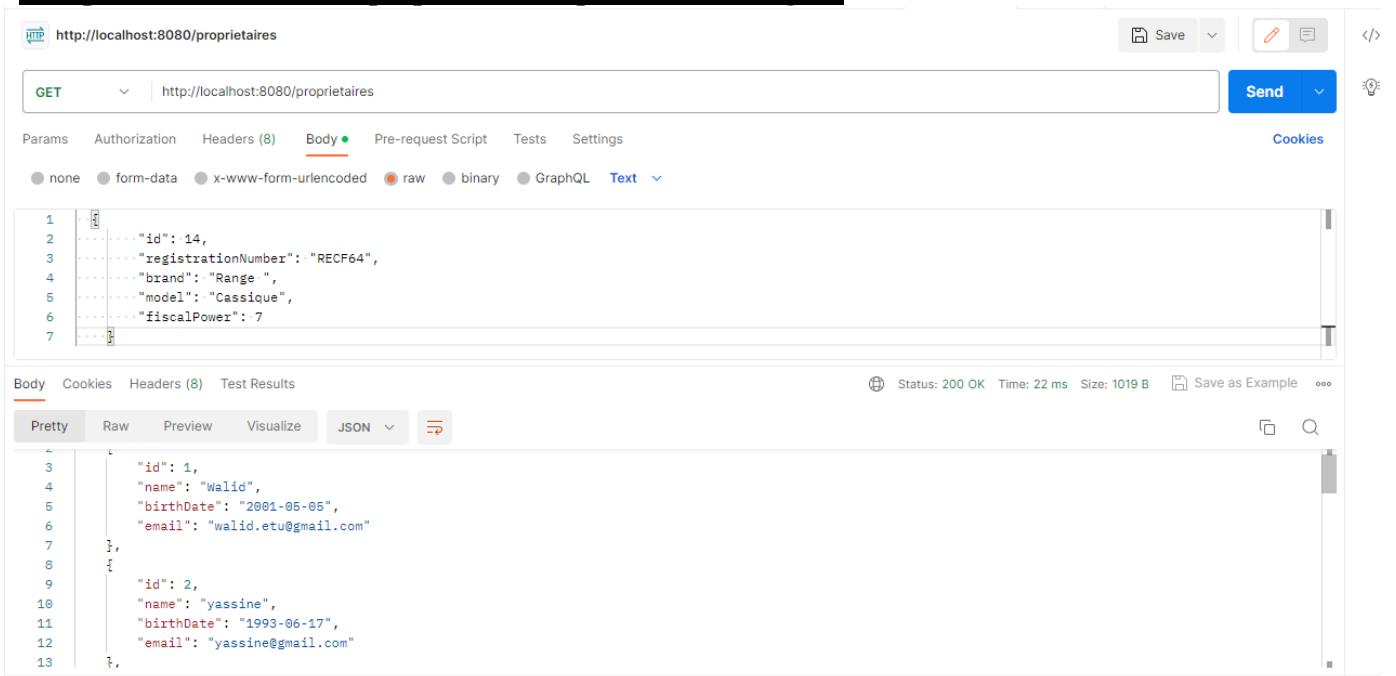
### **Récupération de toutes les véhicules : par la méthode get :**



The screenshot shows the Postman interface with a GET request to `http://localhost:8080/vehicules`. The response status is 200 OK, and the response body is a JSON array containing two vehicle objects:

```
1  [
2   {
3     "id": 1,
4     "registrationNumber": "RFG764",
5     "brand": "Range Rover",
6     "model": "Sport",
7     "fiscalPower": 9
8   },
9   {
10    "id": 2,
11    "registrationNumber": "JH2332",
12    "brand": "Land Over",
13    "model": "Classic",
14    "fiscalPower": 7
15  }
16]
```

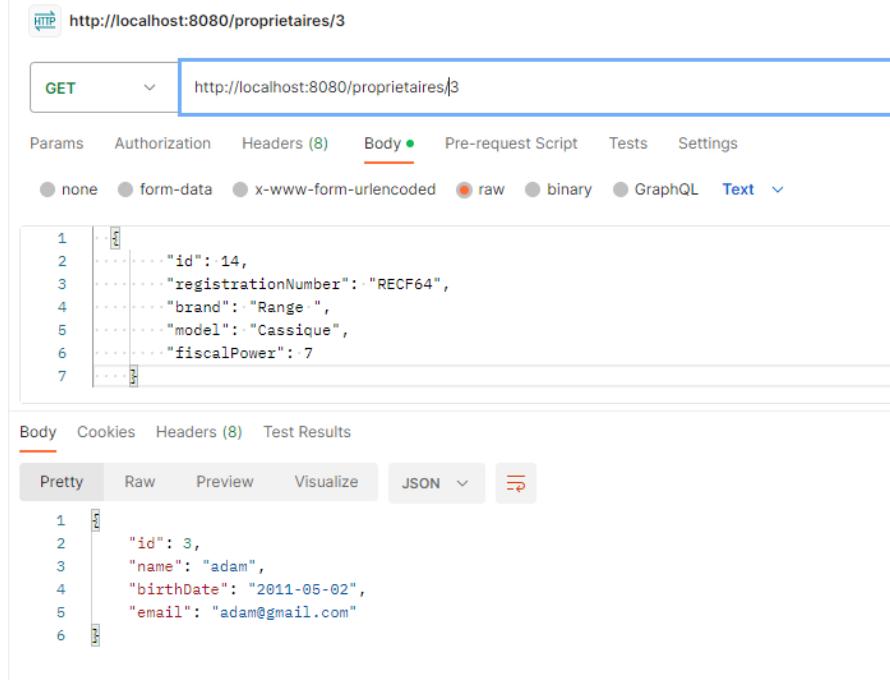
### **Récupération de toutes les propriétaires : par la méthode get :**



The screenshot shows the Postman interface with a GET request to `http://localhost:8080/proprietaires`. The response status is 200 OK, and the response body is a JSON array containing two owner objects:

```
1  [
2   {
3     "id": 1,
4     "name": "Walid",
5     "birthDate": "2001-05-05",
6     "email": "walid.etu@gmail.com"
7   },
8   {
9     "id": 2,
10    "name": "yassine",
11    "birthDate": "1993-06-17",
12    "email": "yassine@gmail.com"
13  }
14]
```

## Récupération propriétaires par Id :



HTTP <http://localhost:8080/proprietaires/3>

GET <http://localhost:8080/proprietaires/3>

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Body (Text)

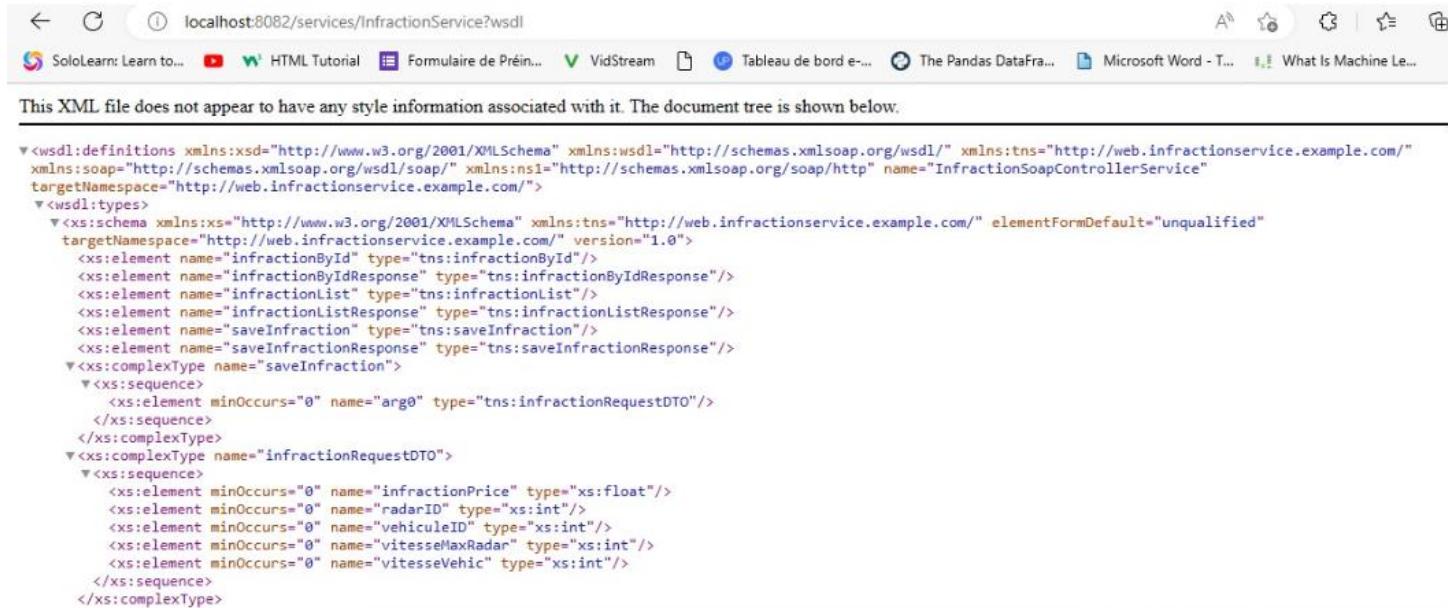
```
1 ... "id": 14,
2 ... "registrationNumber": "RECF64",
3 ... "brand": "Range",
4 ... "model": "Cassique",
5 ... "fiscalPower": 7
6 ...
7 ...
```

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```
1 ...
2 ...
3 ...
4 ...
5 ...
6 ...
```

Maintenant on lance le micro service, et on teste les méthodes qu'on a développé à travers l'outil SOAPUI : **pour le SOAP :**



localhost:8082/services/InfractionService?wsdl

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

```
<wsdl:definitions xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="http://web.infractionservice.example.com/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:ns1="http://schemas.xmlsoap.org/soap/http" name="InfractionSoapControllerService" targetNamespace="http://web.infractionservice.example.com/">
  <wsdl:types>
    <xss:schema xmlns:xss="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://web.infractionservice.example.com/" elementFormDefault="unqualified" targetNamespace="http://web.infractionservice.example.com/" version="1.0">
      <xss:element name="infractionById" type="tns:infractionById"/>
      <xss:element name="infractionByIdResponse" type="tns:infractionByIdResponse"/>
      <xss:element name="infractionList" type="tns:infractionList"/>
      <xss:element name="infractionListResponse" type="tns:infractionListResponse"/>
      <xss:element name="saveInfraction" type="tns:saveInfraction"/>
      <xss:element name="saveInfractionResponse" type="tns:saveInfractionResponse"/>
    </xss:schema>
    <xss:complexType name="saveInfraction">
      <xss:sequence>
        <xss:element minOccurs="0" name="arg0" type="tns:infractionRequestDTO"/>
      </xss:sequence>
    </xss:complexType>
  </wsdl:types>
  <xss:complexType name="infractionRequestDTO">
    <xss:sequence>
      <xss:element minOccurs="0" name="infractionPrice" type="xs:float"/>
      <xss:element minOccurs="0" name="radarID" type="xs:int"/>
      <xss:element minOccurs="0" name="vehiculeID" type="xs:int"/>
      <xss:element minOccurs="0" name="vitesseMaxRadar" type="xs:int"/>
      <xss:element minOccurs="0" name="vitesseVehic" type="xs:int"/>
    </xss:sequence>
  </xss:complexType>

```

SO API Request 1

http://localhost:8082/services/InfractionService

Raw XML

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope">
  <soapenv:Header>
  <soapenv:Body>
    <web:infractionById>
      <!--Optional:-->
      <id>4</id>
    </web:infractionById>
  </soapenv:Body>
</soapenv:Envelope>
```

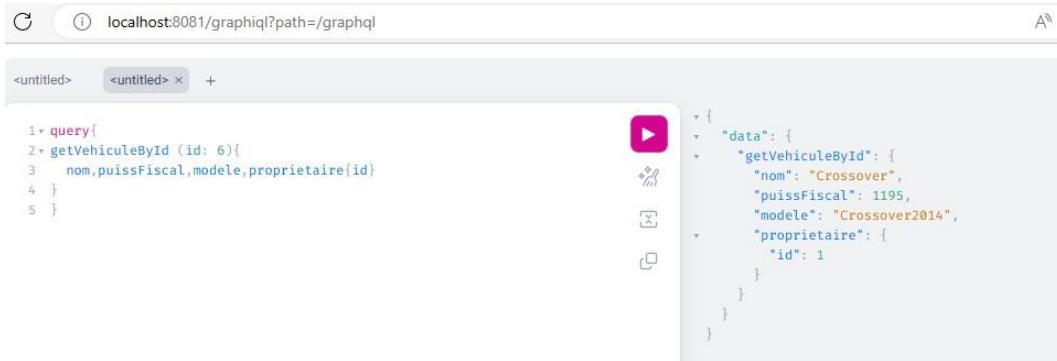
Raw XML

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope">
  <soap:Body>
    <ns2:infractionByIdResponse xmlns:ns2="http://web.infra">
      <return>
        <dateInfra>2023-06-02T12:32:44.909+01:00</dateInfra>
        <id>4</id>
        <infractionPrice>238.21776</infractionPrice>
        <radarID>57702190</radarID>
        <vehiculeID>1171963211</vehiculeID>
        <vitesseMaxRadar>102</vitesseMaxRadar>
        <vitesseVehic>131</vitesseVehic>
      </return>
    </ns2:infractionByIdResponse>
  </soap:Body>
</soap:Envelope>
```

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

**Maintenant, on passe vers le teste de web service GraphOL :**

**La récupération d'un véhicule par identifiant :**



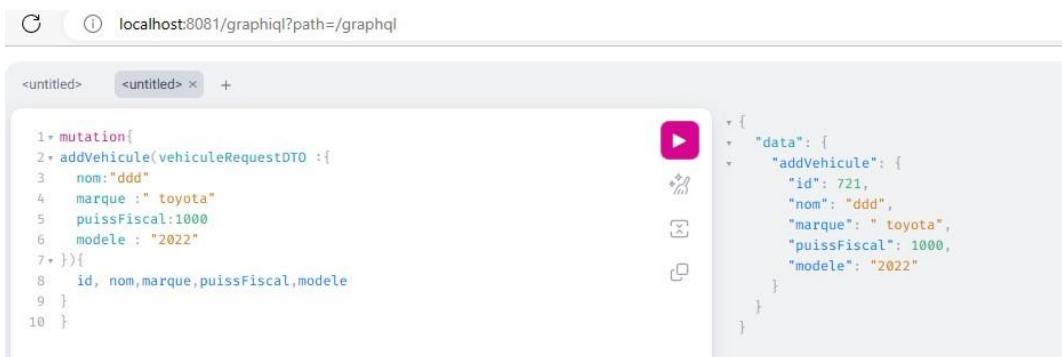
The screenshot shows the GraphQL playground interface at `localhost:8081/graphiql?path=/graphql`. A query is being run:

```
query {
  getVehiculeById(id: 6) {
    nom, puissFiscal, modele, proprietaire{id}
  }
}
```

The response pane shows the result of the query:

```
{
  "data": {
    "getVehiculeById": {
      "nom": "Crossover",
      "puissFiscal": 1195,
      "modele": "Crossover2014",
      "proprietaire": {
        "id": 1
      }
    }
  }
}
```

**L'ajout d'un véhicule :**



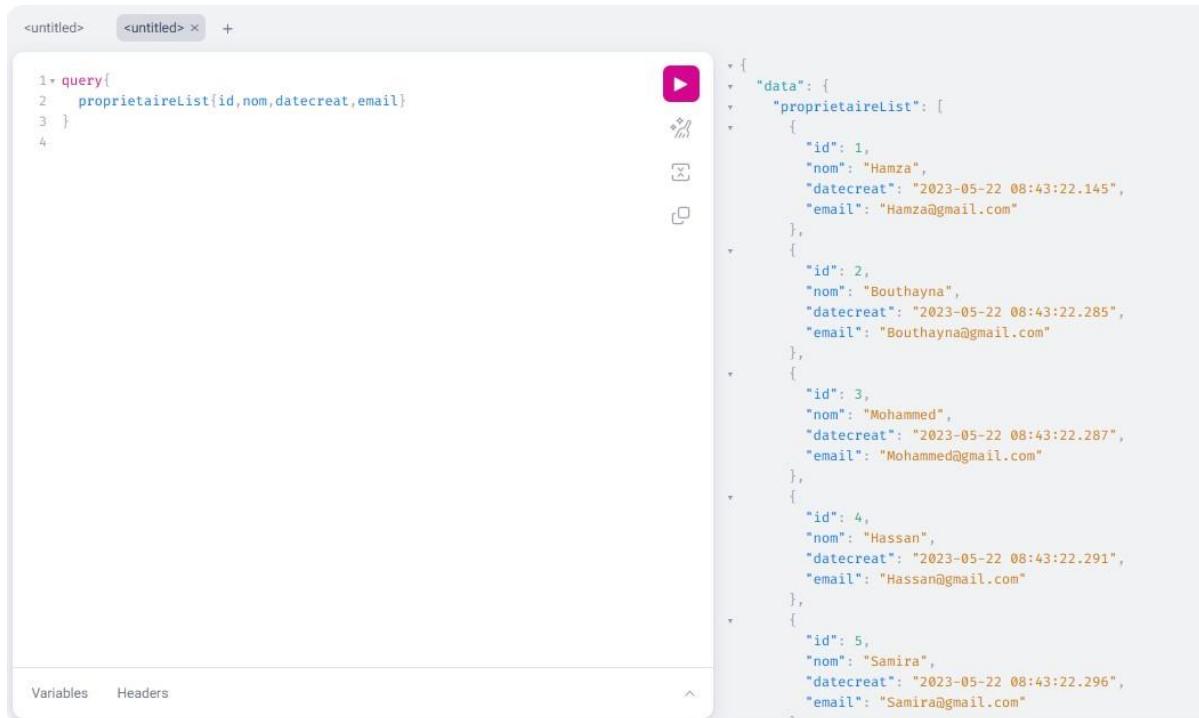
The screenshot shows the GraphQL playground interface at `localhost:8081/graphiql?path=/graphql`. A mutation is being run:

```
mutation{
  addVehicule(vehiculeRequestDTO :{
    nom:"ddd"
    marque :" toyota"
    puissFiscal:1000
    modele : "2022"
  }){
    id, nom,marque,puissFiscal,modele
  }
}
```

The response pane shows the result of the mutation:

```
{
  "data": {
    "addVehicule": {
      "id": 721,
      "nom": "ddd",
      "marque": " toyota",
      "puissFiscal": 1000,
      "modele": "2022"
    }
  }
}
```

Pour l'entité propriétaire, on a **proprietaireList** :



The screenshot shows a GraphQL playground interface with a query editor and a results panel.

**Query:**

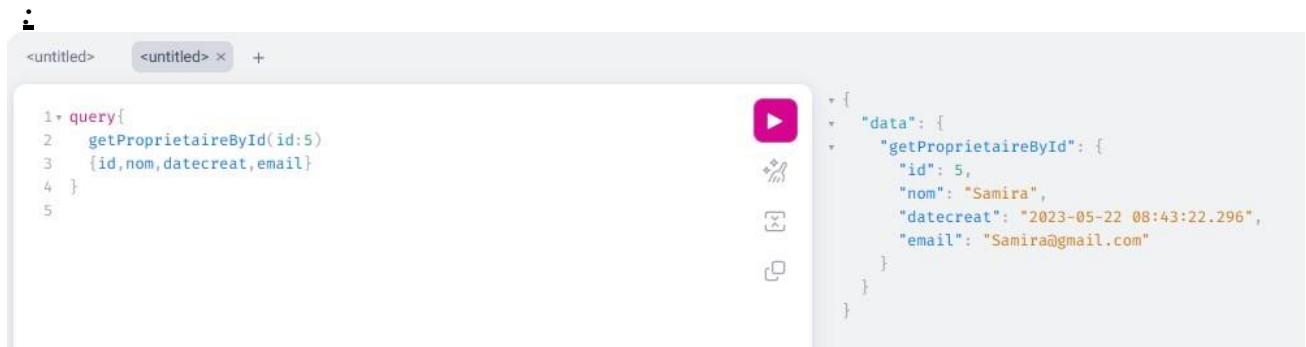
```
1+ query{  
2   proprietaireList{id,nom,datecreat,email}  
3 }  
4
```

**Results:**

```
+ {  
  "data": {  
    "proprietaireList": [  
      {  
        "id": 1,  
        "nom": "Hamza",  
        "datecreat": "2023-05-22 08:43:22.145",  
        "email": "Hamza@gmail.com"  
      },  
      {  
        "id": 2,  
        "nom": "Bouthayna",  
        "datecreat": "2023-05-22 08:43:22.285",  
        "email": "Bouthayna@gmail.com"  
      },  
      {  
        "id": 3,  
        "nom": "Mohammed",  
        "datecreat": "2023-05-22 08:43:22.287",  
        "email": "Mohammed@gmail.com"  
      },  
      {  
        "id": 4,  
        "nom": "Hassan",  
        "datecreat": "2023-05-22 08:43:22.291",  
        "email": "Hassan@gmail.com"  
      },  
      {  
        "id": 5,  
        "nom": "Samira",  
        "datecreat": "2023-05-22 08:43:22.296",  
        "email": "Samira@gmail.com"  
      }  
    ]  
  }  
}
```

Variables Headers

### **La récupération d'un propriétaire par identifiant**



The screenshot shows a GraphQL playground interface with a query editor and a results panel.

**Query:**

```
1+ query{  
2   getProprietaireById(id:5)  
3   {id,nom,datecreat,email}  
4 }  
5
```

**Results:**

```
+ {  
  "data": {  
    "getProprietaireById": {  
      "id": 5,  
      "nom": "Samira",  
      "datecreat": "2023-05-22 08:43:22.296",  
      "email": "Samira@gmail.com"  
    }  
  }  
}
```

## 4. Développer le micro-service Infractions:

Dans le package entities on a créé infraction :

```
@Data @NoArgsConstructor @AllArgsConstructor @ToString @Builder
public class Infraction {
    no usages
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    no usages
    private Date date;
    no usages
    private Boolean paid = Boolean.FALSE;
    no usages
    private Long numeroRadar;
    no usages
    private Double vitesseMaxRadar;
    no usages
    private String matriculeVehicule;
    no usages
    private double vitesseVehicule;
    no usages
    private double montantFraction;
    no usages
    private String ownerName;
}
```

Dans le package repositories on a créé l'interface infractionrepository :

```
package com.example.infractionservices.repositories;

import com.example.infractionservices.entities.Infraction;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.rest.core.annotation.RepositoryRestResource;

4 usages
@RepositoryRestResource
public interface InfractionRepository extends JpaRepository<Infraction, Long> {
```

Dans le package web on a créé la classe InfractionRestRepository :

```
package com.example.infractionservices.web;

import ...;

no usages
@RestController
@AllArgsConstructor
@RestResource
@CrossOrigin(origins = "http://localhost:4200")
public class InfractionRestController {

    6 usages
    private InfractionRepository infractionRepository;

    //private VehicleRestClient vehicleRestClient;
    //private RadarRestClient radarRestClient;

    no usages
    @GetMapping("/infractions")
    public List<Infraction> getAllInfractions() { return infractionRepository.findAll(); }

    no usages
    @GetMapping("/infractions/{id}")
    public Infraction getOneInfraction(@PathVariable long id) { return infractionRepository.findById(id).get(); }

    //Update method :
}
```

La création des infractions :

```
package com.example.infractionservices;

import ...;

1 usage
@SpringBootApplication
public class InfractionServicesApplication {
    no usages
    public static void main(String[] args) {
        SpringApplication.run(InfractionServicesApplication.class, args);
    }

    no usages
    @Bean
    public CommandLineRunner demo(InfractionRepository infractionRepository) {
        return args -> {
            // Create 6 instances of Infraction
            Infraction infraction1 = Infraction.builder()
                .date(new Date())
                .numeroRadar(2L)
                .vitesseMaxRadar(100.0)
                .matriculeVehicule("12AABB")
                .vitesseVehicule(120.0)
                .montantFraction(50.0)
                .ownerName("Messi")
                .build();
        };
    }
}
```

Application.Properties :

```
spring.application.name=infraction-service
server.port=8081
spring.datasource.url=jdbc:h2:mem:infraction-db
spring.h2.console.enabled=true
spring.cloud.discovery.enabled=true
```

## 5. Développer le micro-service Radar:

Dans le package entities on a créé Radar :

```
package com.example.radarservice.entities;

import ...

20 usages
@javax.persistence.Entity
@Data @NoArgsConstructor @AllArgsConstructor @Builder
public class Radar {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private Double maxSpeed = 120.0;
    private boolean status; // recently added
    //private String numeroRadar;
    private double longitude;
    private double latitude;
}
```

Dans le package feignClients on a créé ImmatriculationClient :

```
package com.example.radarservice.feignClients;

import ...

2 usages
@FeignClient(name = "IMMATRICULATION-SERVICE", url = "http://localhost:8080")
public interface ImmatriculationClient {
    no usages
    @GetMapping("/{proprietaires}/{id}")
    Proprietaire getProprietaire(@PathVariable Long id);

    1 usage
    @GetMapping("/{vehicules}/matricules/{matricule}")
    Vehicule getVehiculeByMatricule(@PathVariable String matricule);
}
```

Dans le package feignClients on a créé InfractionClient :

```
package com.example.radarservice.feignClients;

import ...

2 usages
@FeignClient(name = "INFRACTION-SERVICE", url = "http://localhost:8081")
public interface InfractionClient {
    1 usage
    @PostMapping ("/infractions")
    Infraction createInfraction(@RequestBody Infraction infraction);

}
```

Dans le package models on a créé la classe Infraction :

```
package com.example.radarservice.models;

import ...;

7 usages
@Entity
@Data @NoArgsConstructor @AllArgsConstructor @ToString
@Builder
public class Infraction {
    no usages
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    no usages
    private Date date;
    no usages
    private Long numeroRadar;
    no usages
    private Double vitesseMaxRadar;
    no usages
    private String matriculeVehicule;
    no usages
    private double vitesseVehicule;
    no usages
    private double montantFraction;
```

Dans le package models on a créé la classe Proprietaire :

```
package com.example.radarservice.models;

import ...;

@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class Proprietaire {
    1 usage
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    1 usage
    private String name;
    1 usage
    private LocalDate birthDate;
    1 usage
    private String email;
    1 usage
    @OneToMany(mappedBy = "proprietaire", fetch = FetchType.EAGER)
    @JsonIgnore
    private Collection<Vehicule> vehicules;
    //
    // @Override
```

Dans le package models on a créé la classe Véhicule :

```
package com.example.radarService.models;

import ...

@Entity
@Data @NoArgsConstructor @AllArgsConstructor @Builder
public class Véhicule {
    1 usage
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    1 usage
    private String registrationNumber;
    1 usage
    private String brand;
    1 usage
    private String model;
    1 usage
    private int fiscalPower;
    no usages
    @ManyToOne
    @JsonIgnore
    Proprietaire propriétaire;
```

Dans le package repositories on a créé l'interface RadarRepository :

```
package com.example.radarService.repositories;

import ...

4 usages
public interface RadarRepository extends JpaRepository<Radar, Long> {
    1 usage
    Page<Radar> findByNameContains(String keyword, Pageable pageable);
    //Optional<Radar> findByNumero(String numero);
}
```

Dans le package web on a créé l'interface RadarRestController :

```
no usages
@RestController
@AllArgsConstructor
@CrossOrigin(origins = "http://localhost:4200")
public class RadarRestController {
    10 usages
    private RadarRepository radarRepository;
    1 usage
    private ImmatriculationClient immatriculationClient;
    1 usage
    private InfractionClient infractionClient;

    no usages
    @GetMapping("/radars")
    public List<Radar> getAllRadars() { return radarRepository.findAll(); }

    no usages
    @GetMapping("/{id}")
    public Radar getRadarById(@PathVariable Long id) {
        if (id == null) {
            throw new IllegalArgumentException("Radar ID cannot be null");
        }

        return radarRepository.findById(id).orElse(null);
    }
}
```

La création des Radars :

```
package com.example.radarservice;

import ...

1 usage
@SpringBootApplication
@EnableFeignClients
public class RadarServiceApplication {

    no usages
    public static void main(String[] args) { SpringApplication.run(RadarServiceApplication.class, args); }

    no usages
    @Bean
    public CommandLineRunner init(RadarRepository radarRepository) {
        return args -> {
            Radar radar = Radar.builder()
                .name("Valid Radar")
                .maxSpeed(100.0)
                .longitude(123.456)
                .latitude(78.910)
                // .numeroRadar("P00")
                .build();

            radarRepository.save(radar);
            Radar radar1 = Radar.builder()
                .name("Adam Radar")
        };
    }
}
```

Application.properties :

```
spring.application.name=radar-service
server.port=8082
spring.datasource.url=jdbc:h2:mem:radar-db
spring.h2.console.enabled=true
spring.cloud.discovery.enabled=true
```

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

Gateway est responsable de la réception des requêtes client et de la communication avec le service eureka :

Le service eureka:

The screenshot shows the Eureka UI interface. At the top, there's a header with tabs like 'localhost:8761', 'HTML Tutorial', 'Formulaire de Prén...', 'VidStream', 'Tableau de bord e...', 'The Pandas DataFra...', 'Microsoft Word - T...', and 'What Is Machine Le...'. Below the header, there are two main sections: 'Environment' and 'DS Replicas'.

**Environment** section details:

Environment	test	Current time	2023-06-25T09:50:11 +0100
Data center	default	Uptime	00:07
		Lease expiration enabled	true
		Renews threshold	8
		Renews (last min)	12

**DS Replicas** section details:

localhost
-----------

**Instances currently registered with Eureka**

Application	AMIs	Availability Zones	Status
GATEWAY-SERVICE	n/a (1)	(1)	UP (1) - localhost:gateway-service:8888
IMMATRICULATION-SERVICE	n/a (1)	(1)	UP (1) - DESKTOP-496CS9N.lan:immatriculation-service
INFRACTION-SERVICE	n/a (1)	(1)	UP (1) - DESKTOP-496CS9N.lan:infraction-service:8081
RADAR-SERVICE	n/a (1)	(1)	UP (1) - localhost:radar-service:8082

**General Info**

Le service GATEWAY :

```
package com.example.gatewayprojet;

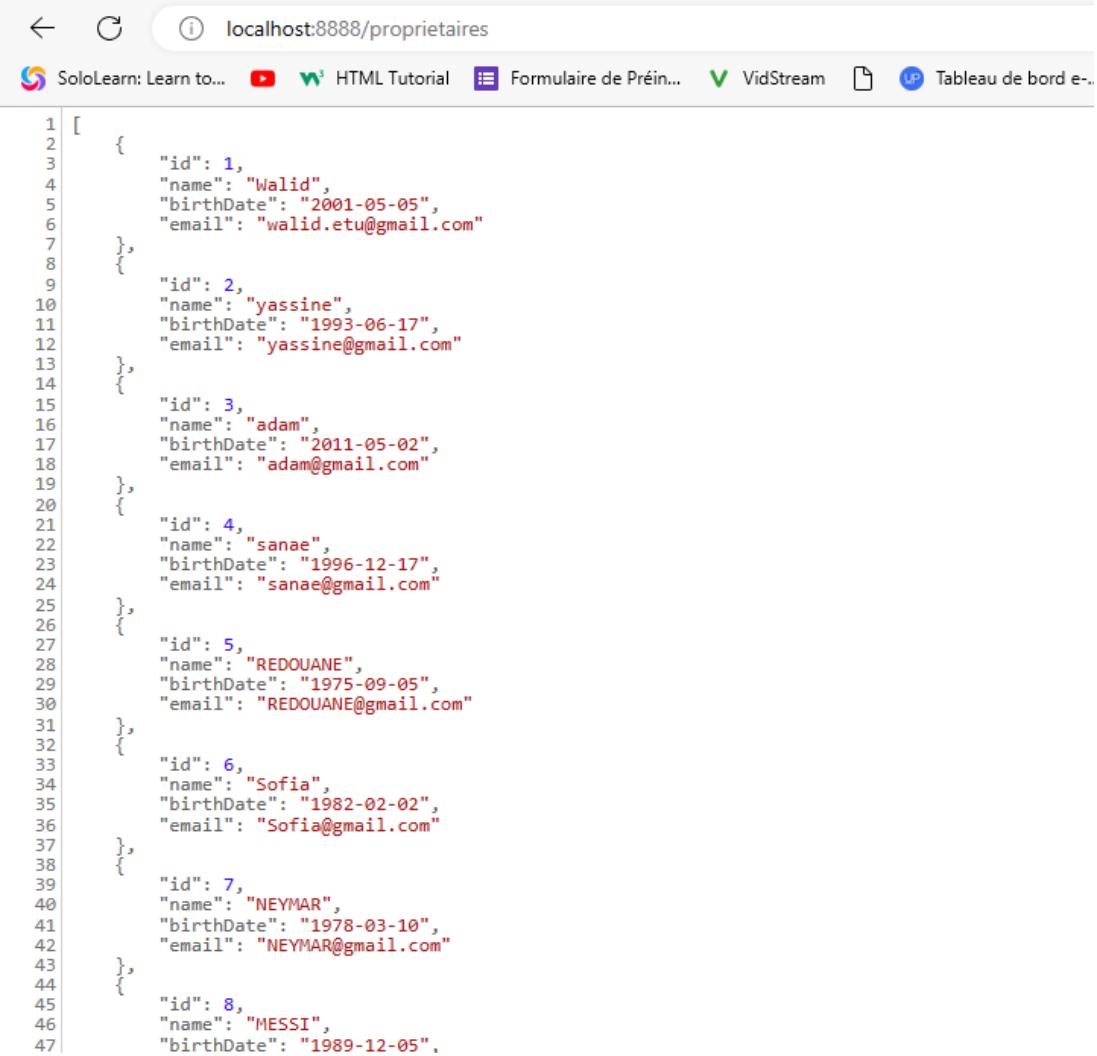
import ...;

@Usage
@SpringBootApplication
public class GatewayProjetApplication {

    no usages
    public static void main(String[] args) { SpringApplication.run(GatewayProjetApplication.class, args); }
    no usages
    @Bean
    RouteLocator routeLocator(RouteLocatorBuilder builder){
        return builder.routes()
            .route((r)->r.path("/**").uri("http://localhost:8080/"))
            .route((r)->r.path("/**").uri("http://localhost:8080/"))
            .route((r)->r.path("/**").uri("http://localhost:8081/"))
            .route((r)->r.path("/**").uri("http://localhost:8082/"))
            .route((r)->r.path("/proprietaires/**").uri("lb://IMMATRICULATION-SERVICE"))
            .route((r)->r.path("/vehicules/**").uri("lb://INFRACTION-SERVICE"))
            .route((r)->r.path("/infractions/**").uri("lb://IMMATRICULATION-SERVICE"))
            .build();
    }
    @Bean
    // DiscoveryClientRouteDefinitionLocator definitionLocator(
    //     ReactiveDiscoveryClient rdc,
    // )
}
```

On essaie d'accéder aux services depuis le port de la gateway :

Le microservice Proprietaire :



```
[{"id": 1, "name": "Walid", "birthDate": "2001-05-05", "email": "walid.etu@gmail.com"}, {"id": 2, "name": "yassine", "birthDate": "1993-06-17", "email": "yassine@gmail.com"}, {"id": 3, "name": "adam", "birthDate": "2011-05-02", "email": "adam@gmail.com"}, {"id": 4, "name": "sanae", "birthDate": "1996-12-17", "email": "sanae@gmail.com"}, {"id": 5, "name": "REDOUANE", "birthDate": "1975-09-05", "email": "REDOUANE@gmail.com"}, {"id": 6, "name": "Sofia", "birthDate": "1982-02-02", "email": "Sofia@gmail.com"}, {"id": 7, "name": "NEYMAR", "birthDate": "1978-03-10", "email": "NEYMAR@gmail.com"}, {"id": 8, "name": "MESSI", "birthDate": "1989-12-05", "email": "MESSI@gmail.com"}]
```

## Le microservice infraction :



A screenshot of a browser window displaying a JSON array of four traffic violation records. The array is indexed from 1 to 4. Each record contains fields such as id, date, paid, numeroRadar, vitesseMaxRadar, matriculeVehicule, vitesseVehicule, montantFraction, and ownerName.

```
[{"id": 1, "date": "2023-06-25T08:48:59.243+00:00", "paid": null, "numeroRadar": 2, "vitesseMaxRadar": 120, "matriculeVehicule": "12AABB", "vitesseVehicule": 120, "montantFraction": 50, "ownerName": "NEYMAR"}, {"id": 2, "date": "2023-06-25T08:48:59.243+00:00", "paid": null, "numeroRadar": 2, "vitesseMaxRadar": 80, "matriculeVehicule": "z20DD", "vitesseVehicule": 90, "montantFraction": 30, "ownerName": "HAKIMI"}, {"id": 3, "date": "2023-06-25T08:48:59.243+00:00", "paid": null, "numeroRadar": 3, "vitesseMaxRadar": 80, "matriculeVehicule": "XYZ789", "vitesseVehicule": 80, "montantFraction": 90, "ownerName": "MESSI"}, {"id": 4, "date": "2023-06-25T08:48:59.243+00:00", "paid": null, "numeroRadar": 2, "vitesseMaxRadar": 120, "matriculeVehicule": "UIO890", "vitesseVehicule": 130, "montantFraction": 20, "ownerName": "ADAM"}]
```

## Le microservice radar :



A screenshot of a browser window displaying a JSON array of three radar station records. The array is indexed from 1 to 3. Each record contains fields such as id, name, maxSpeed, status, longitude, and latitude.

```
[{"id": 1, "name": "Test Radar", "maxSpeed": 100, "status": false, "longitude": 123.456, "latitude": 78.91}, {"id": 2, "name": "adam Radar", "maxSpeed": 100.3, "status": false, "longitude": 130.456, "latitude": 30.91}, {"id": 3, "name": "walid Radar", "maxSpeed": 130.9, "status": false, "longitude": 123.456, "latitude": 78.91}]
```

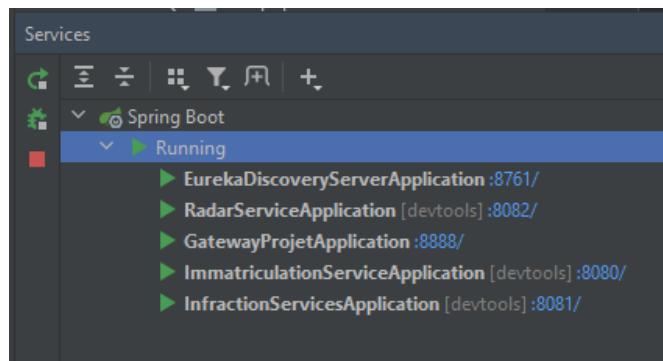
## Le microservice Vehicles :



The screenshot shows a browser window with the URL `localhost:8888/vehicules`. The page title is "SoloLearn: Learn to...". Below the title, there are tabs for "HTML Tutorial" and "Formulaire de Préin...". The main content area displays a JSON array of vehicle data, numbered from 1 to 6. Each vehicle object contains fields: id, registrationNumber, brand, model, and fiscalPower.

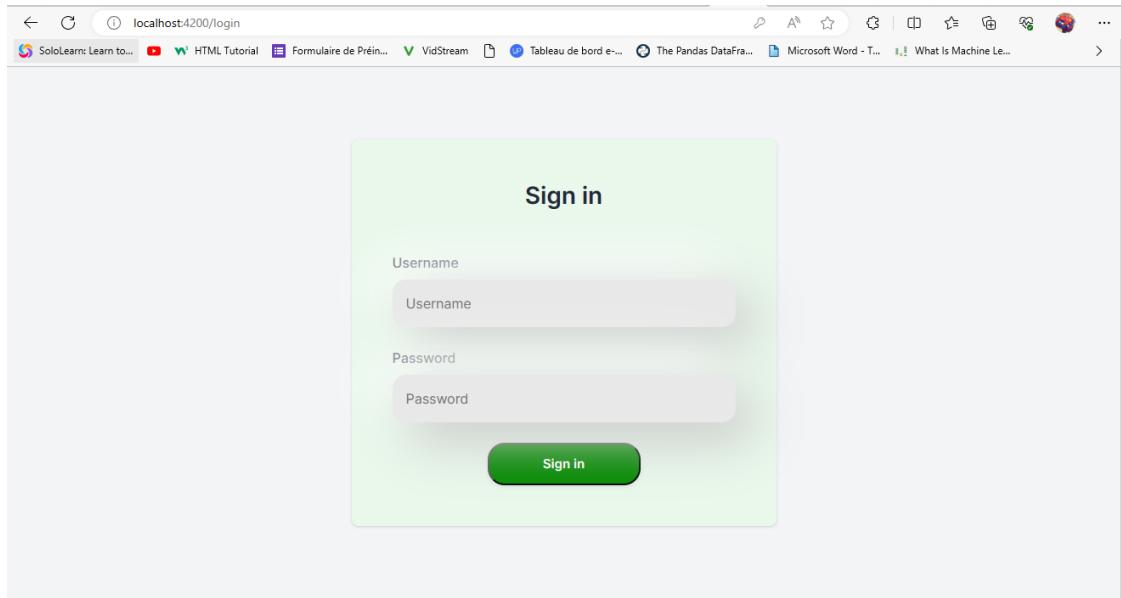
```
1 [  
2 {  
3     "id": 1,  
4     "registrationNumber": "RFG764",  
5     "brand": "Range Rover",  
6     "model": "Sport",  
7     "fiscalPower": 9  
8 },  
9 {  
10    "id": 2,  
11    "registrationNumber": "JH2332",  
12    "brand": "Land Rover",  
13    "model": "Classic",  
14    "fiscalPower": 7  
15 },  
16 {  
17    "id": 3,  
18    "registrationNumber": "HJG412",  
19    "brand": "BMW",  
20    "model": "2021",  
21    "fiscalPower": 10  
22 },  
23 {  
24    "id": 4,  
25    "registrationNumber": "87TGH",  
26    "brand": "Dacia",  
27    "model": "SANDRO",  
28    "fiscalPower": 6  
29 },  
30 {  
31    "id": 5,  
32    "registrationNumber": "JKG2367",  
33    "brand": "Fiat",  
34    "model": "2009",  
35    "fiscalPower": 4  
36 },  
37 {  
38    "id": 6,  
39    "registrationNumber": "I5612",  
40    "brand": "Mercedes",  
41    "model": "not",  
42    "fiscalPower": 9  
43 },  
44 {
```

## Démarrer tous les Micro Services :

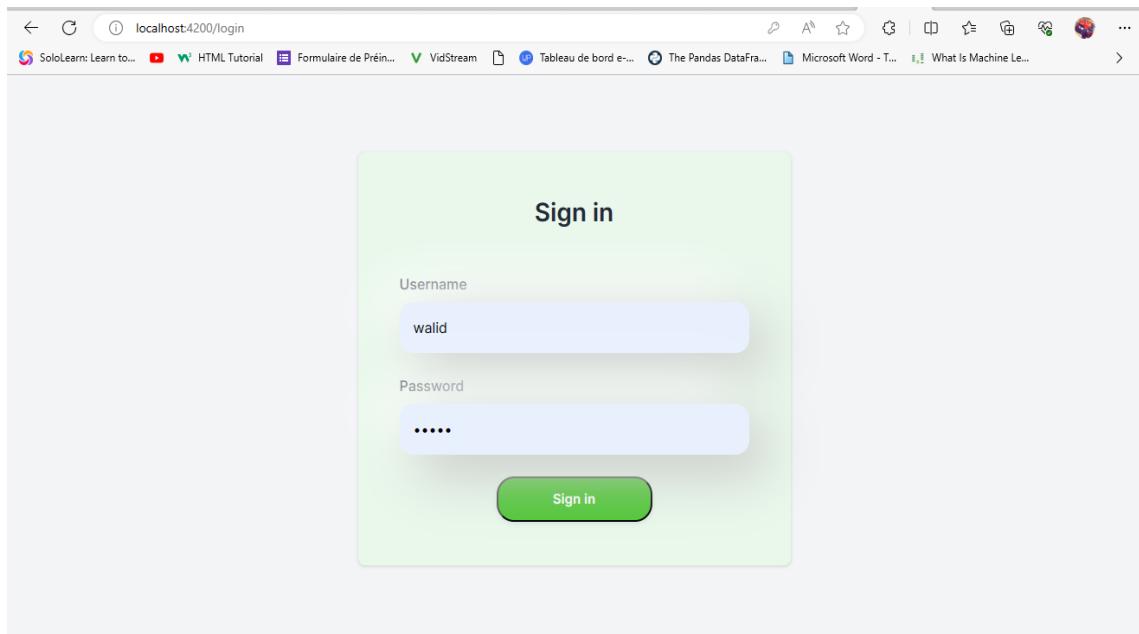


## La Partie Front End avec l'utilisation de Angular :

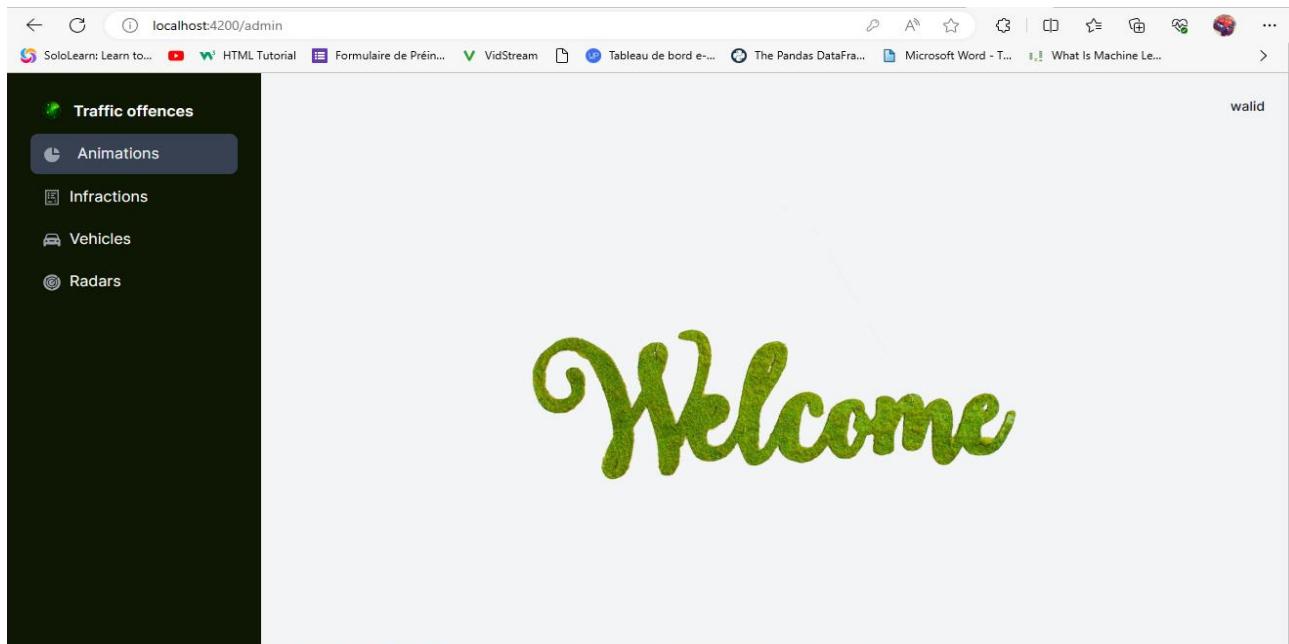
La page de Login :



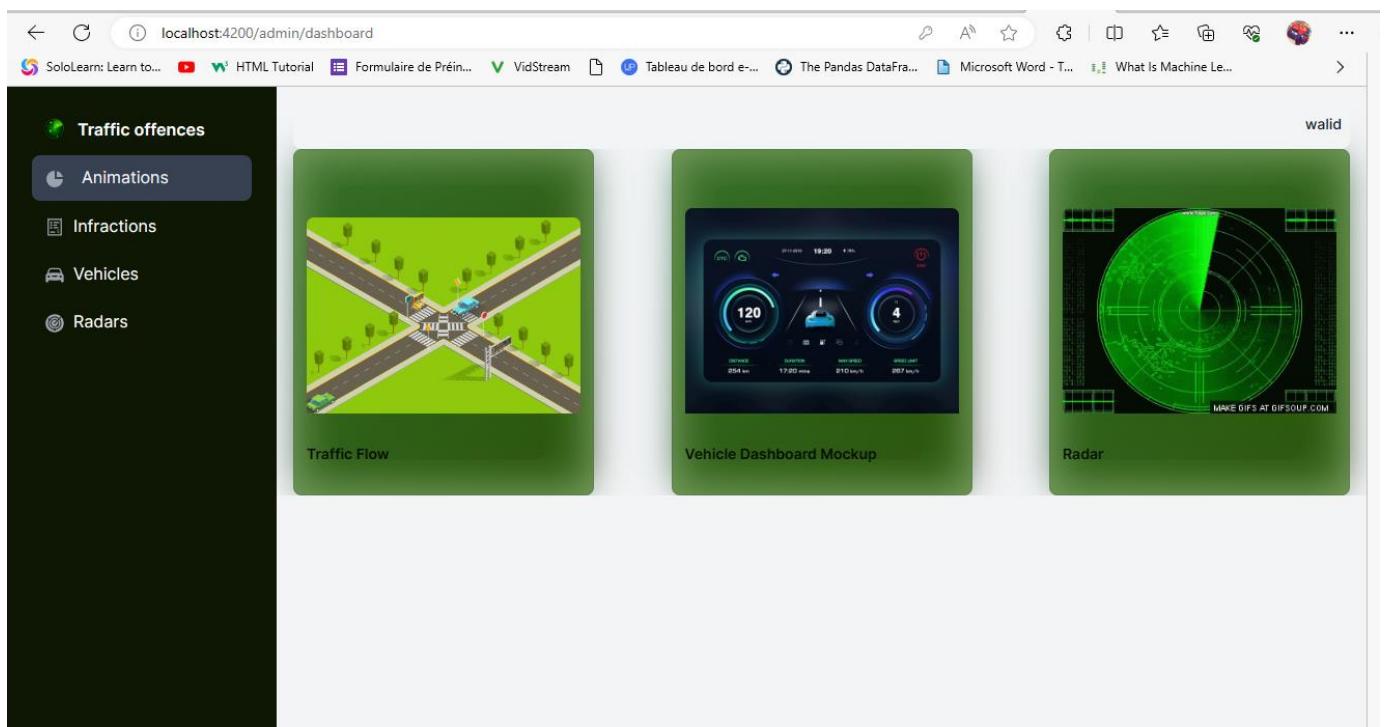
Saisir le nom d'utilisateur et le mot de passe :



La page d'accueil de Dashboard :



La page d'animations de Dashboard :



La page d'infraction de Dashboard :

Traffic offences

Animations

Infractions

Vehicles

Radars

### Speeding fines

	DATE	MATRICULE	OWNER'S NAME	AMOUNT
1	23-06-25 [18:56:00]	12AABB	adam	50
2	23-06-25 [18:56:00]	z20DD	redouane	30
3	23-06-25 [18:56:00]	XYZ789	halima	90
4	23-06-25 [18:56:00]	UIO890	Ovia	20

New infraction

radar id

matricule

vitesse

Submit

Ajouter une infraction :

### New infraction

3

ABC123

100

Submit

Voilà l'infraction a ajoutée :

The screenshot shows a web application interface for managing traffic offenses. On the left, a sidebar menu includes 'Traffic offences' (highlighted), 'Animations', 'Infractions', 'Vehicles', and 'Radars'. The main content area is titled 'Speeding fines' and displays a table of five entries:

	DATE	MATRICULE	OWNER'S NAME	AMOUNT
1	23-06-25 [18:56:00]	12AABB	adam	50
2	23-06-25 [18:56:00]	z20DD	redouane	30
3	23-06-25 [18:56:00]	XYZ789	halima	90
4	23-06-25 [18:56:00]	UIO890	Ovia	20
5	23-06-25 [18:59:49]	ABC123	Walid	400

To the right, a 'New infraction' form contains fields for 'radar id', 'matricule', and 'vitesse', with a 'Submit' button.

La page Vehicule de Dashboard :

The screenshot shows a 'Vehicles' page from the dashboard. The sidebar menu is identical to the previous page. The main content area is titled 'Vehicles' and displays a table of six vehicles:

MATRICULE	BRAND	MODEL	HORSE	OWNER
ABC123	Dacia	Logan	6	
12AABB	Jeep	sport	7	
z20DD	bougati	classic	12	
XYZ789	Reunaut	2001	8	
MNB456	Mercedes	190	9	
PQR987	Mercedes-Benz	Classe A	10	

At the bottom, there is a navigation bar with 'Page 1 of 2' and '»'.

En cliquant sur le propriétaire de chaque ligne, le nom et l'adresse électronique du propriétaire s'affichent :

The screenshot shows a web application interface for managing vehicles. On the left, there is a sidebar with navigation links: 'Traffic offences', 'Animations' (which is currently selected), 'Infractions', 'Vehicles', and 'Radars'. The main content area is titled 'Vehicles' and displays a table with the following data:

MATRICULE	BRAND	MODEL	HORSE	OWNER
ABC123	Dacia	Logan	6	
12AABB	Jeep	sport	7	
z20DD	bougati	classic	12	
XYZ789	Reunaut	2001	8	
MNB456	Mercedes	190	9	
PQR987	Mercedes-Benz	Classe A	10	

Below the table, there is a navigation bar with 'Page 1 of 2' and '»' buttons. To the right of the table, there is a large user profile card for 'Walid' with a placeholder image, the name 'Walid', and the email 'Walid@gmail.com'.

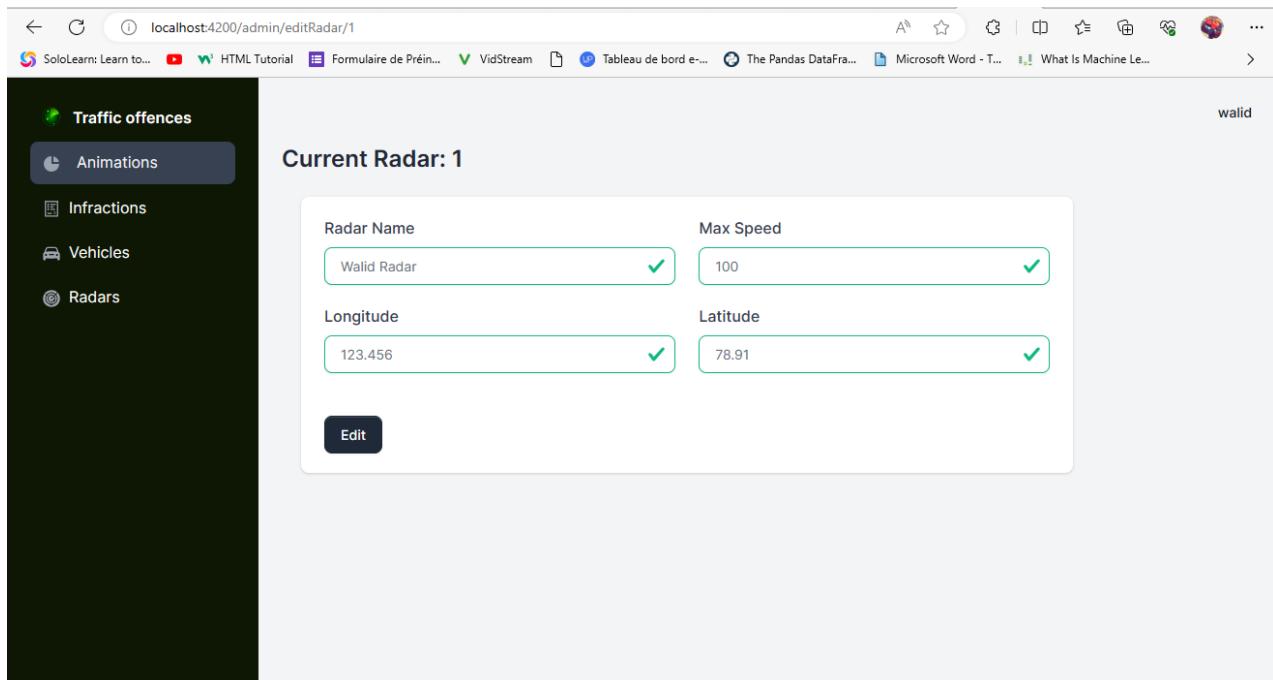
La page Radar de Dashboard :

The screenshot shows a web application interface for managing radars. On the left, there is a sidebar with navigation links: 'Traffic offences', 'Animations' (which is currently selected), 'Infractions', 'Vehicles', and 'Radars'. The main content area is titled 'Radars' and displays a table with the following data:

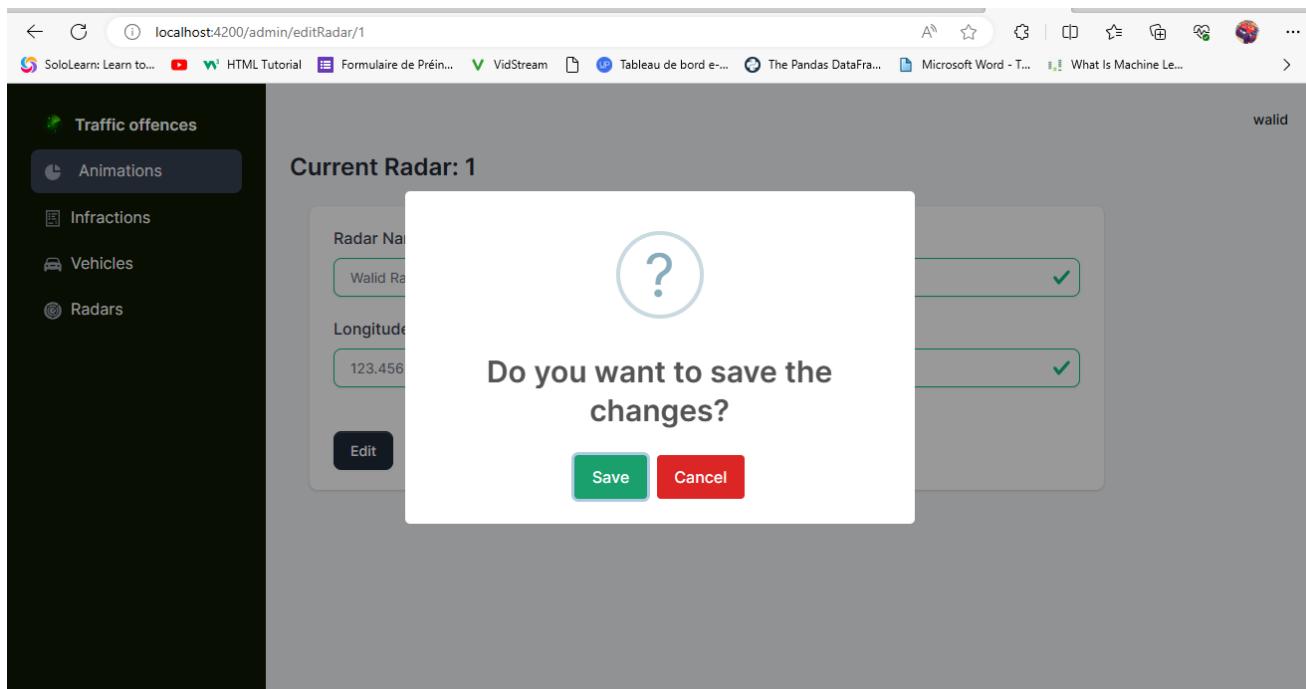
NAME	LONGITUDE	LATITUDE	MAX SPEED	ACTIONS
1 Wалид Radar	123.46	78.91	100 km/h	
2 Adam Radar	130.46	30.91	100.3 km/h	
3 Redouane Radar	123.46	78.91	130.9 km/h	

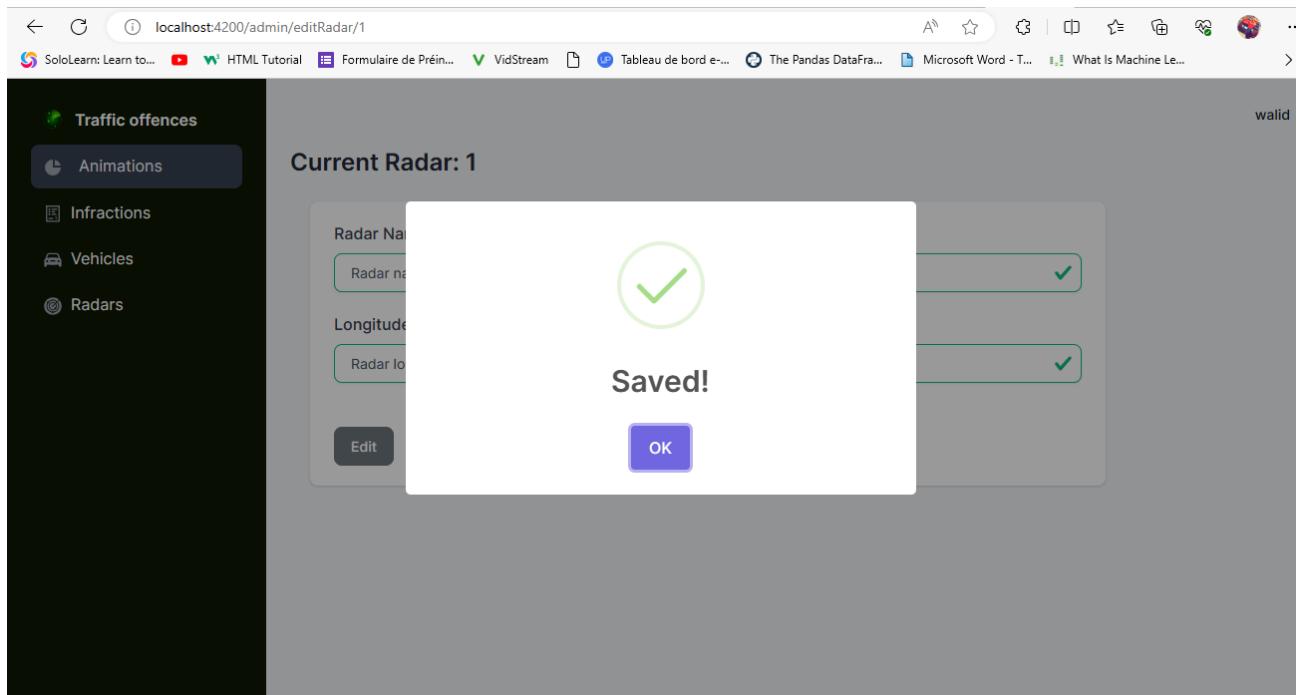
Below the table, there is a navigation bar with 'Page 1 of 1' and '»' buttons. Above the table, there is a button labeled 'New Radar'.

Modification d'un radar :

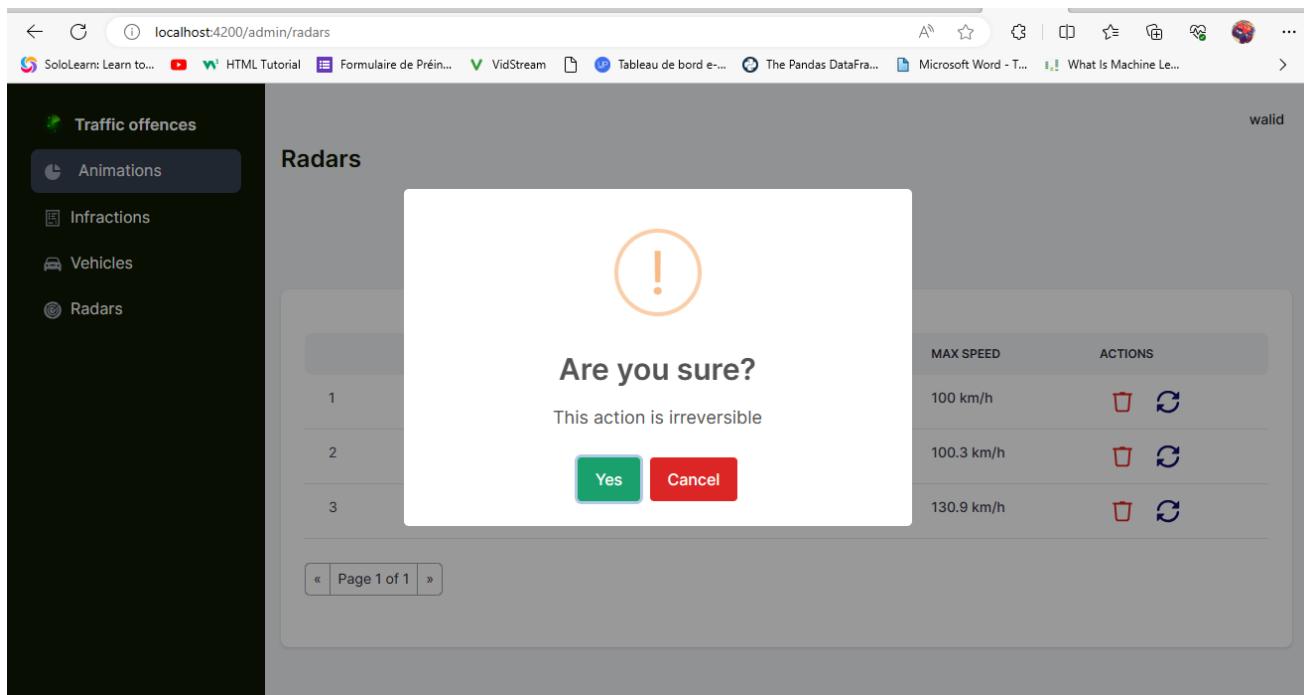


La confirmation :





La suppression d'un rada:



Ajouter un nouveau radar :

New Radar

General information

Radar Name Max Speed

Radar name ✓ Max Speed ✓

Longitude Latitude

Radar longitude ✓ Radar latitude ✓

Save all

Logout :

