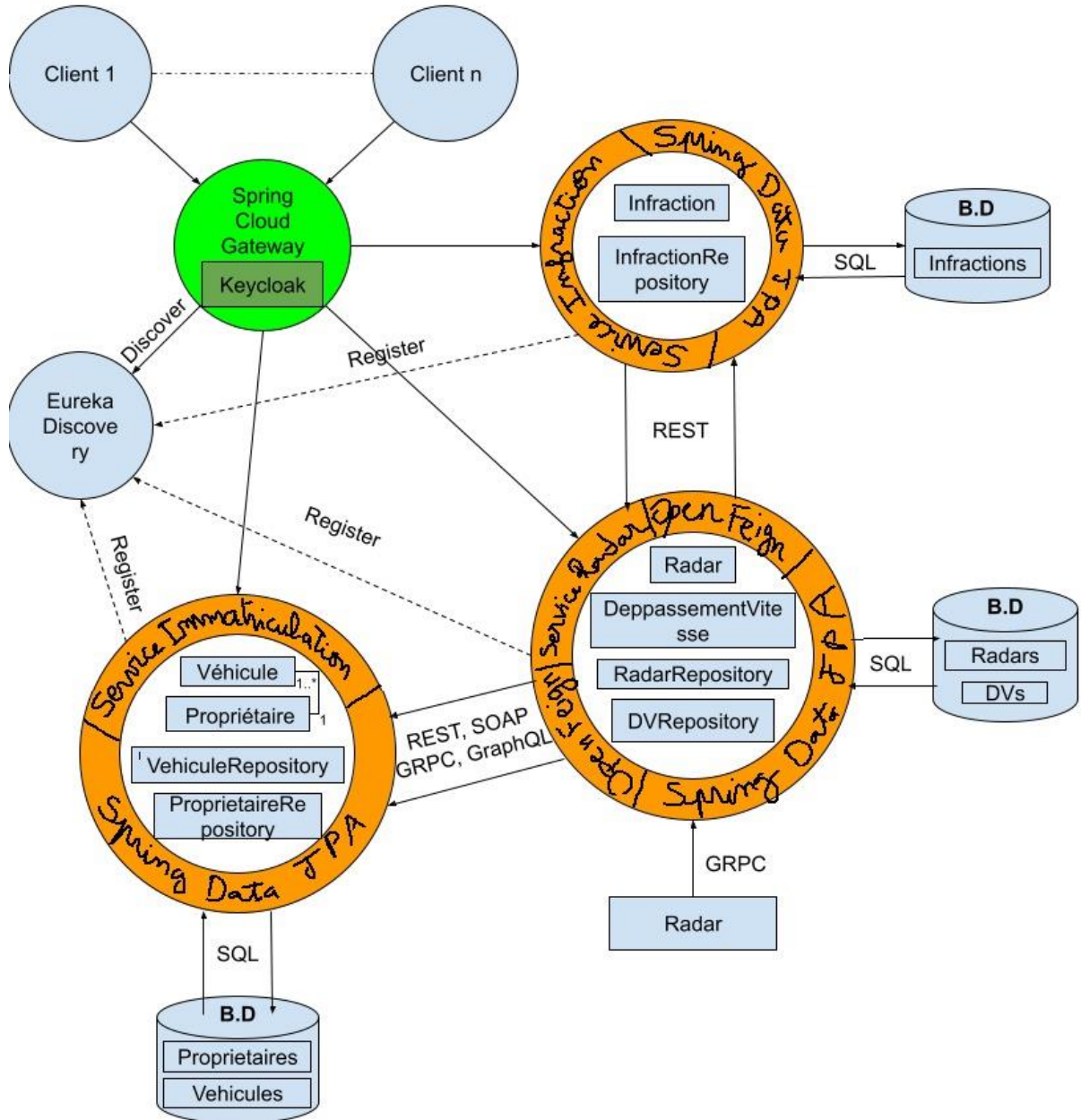


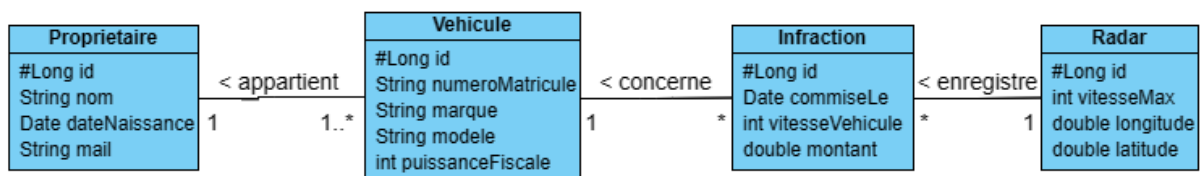
# Rapport Contrôle Système Distribué

## 1. Établir une architecture technique du projet :

Voici mon point de vue concernant l'architecture technique de ce système :



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



### 3. Développer le micro-service Immatriculation :

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

Voici les entités réalisées suivies de leurs repositories :

```
1 usage
@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
@ToString
public class Proprietaire {
    no usages
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id;
    no usages
    String nom;
    no usages
    Date dateNaissance;
    no usages
    String mail;
    no usages
    @OneToMany(mappedBy = "proprietaire")
    List<Vehicule> vehiculesPossedees;
}
```

```

package fs.iaad.immatriculation.repositories;

import fs.iaad.immatriculation.entities.Vehicule;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;

5 usages
@Repository
public interface VehiculeRepository extends JpaRepository<Vehicule, Long> {
    1 usage
    List<Vehicule> findByProprietaireId(Long id);
}

```

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

Voici l'implémentation des quatre web services :

- Rest :

```

package fs.iaad.immatriculation.web;

import ...

no usages
@RestController
@RequestMapping("/api")
public class RestImmatriculationService {
    6 usages
    private final ProprietaireService proprietaireService;
    9 usages
    private final VehiculeService vehiculeService;

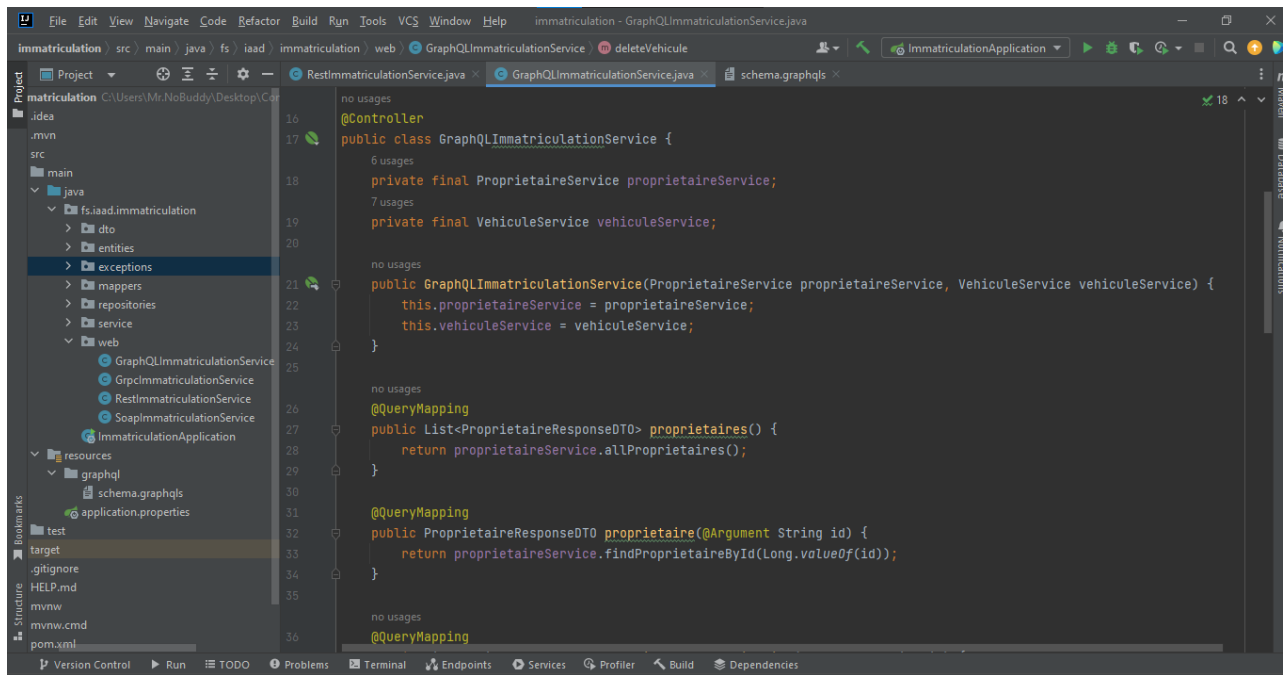
    no usages
    public RestImmatriculationService(ProprietaireService proprietaireService, VehiculeService vehiculeService) {
        this.proprietaireService = proprietaireService;
        this.vehiculeService = vehiculeService;
    }

    @GetMapping("/proprietaires")
    public List<ProprietaireResponseDTO> proprietaires() {
        return proprietaireService.allProprietaires();
    }

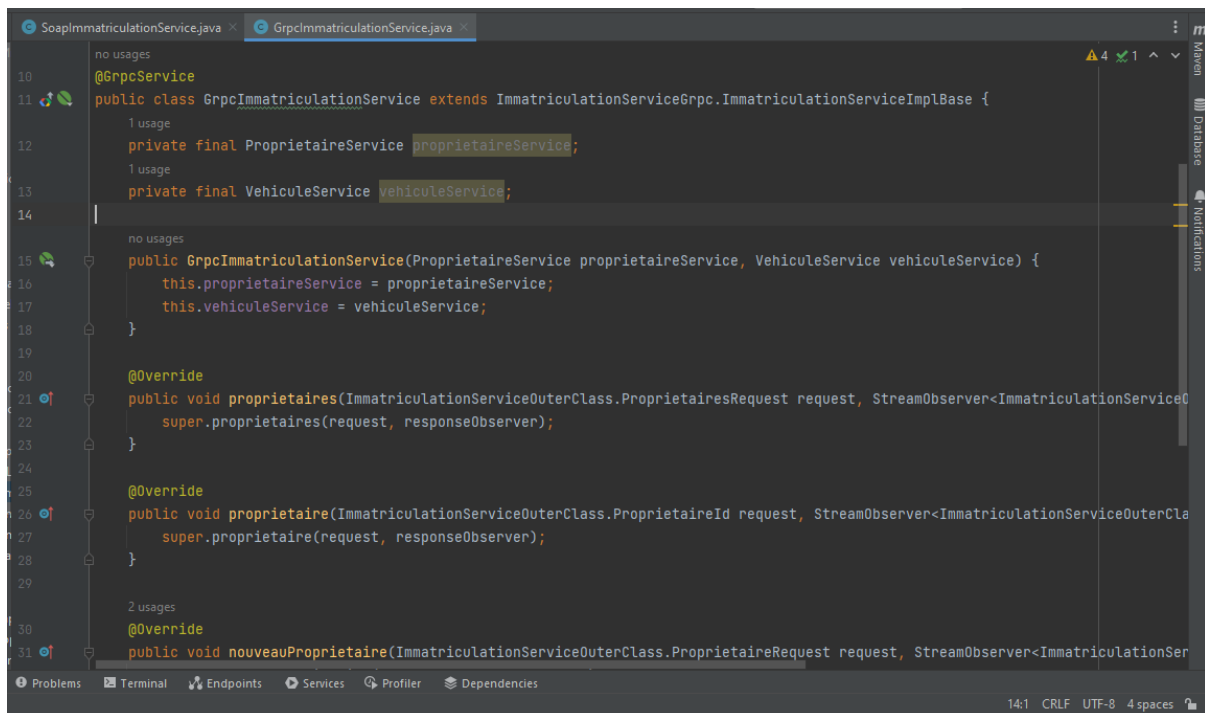
    no usages
    @GetMapping("/vehicules")
    public List<VehiculeResponseDTO> vehicules() {

```

- GraphQL :



- GRPC :



- SOAP :

```
15 SoapImmatriculationService.java
16 2 usages
17 @Component // ni controleur no REST Controleur mais un simple composant
18 @WebService(name = "ImmatriculationSoapService")
19 public class SoapImmatriculationService {
20     6 usages
21     private final ProprietaireService proprietaireService;
22     8 usages
23     private final VehiculeService vehiculeService;
24
25     no usages
26     public SoapImmatriculationService(ProprietaireService proprietaireService, VehiculeService vehiculeService) {
27         this.proprietaireService = proprietaireService;
28         this.vehiculeService = vehiculeService;
29     }
30
31     @WebMethod
32     public List<ProprietaireResponseDTO> proprietaires() { return proprietaireService.allProprietaires(); }
33
34     @WebMethod
35     public ProprietaireResponseDTO proprietaire(@WebParam(name = "id") String id) {
36         return proprietaireService.findProprietaireById(Long.valueOf(id));
37     }
38
39     no usages
40     @WebMethod
41     public ProprietaireResponseDTO nouveauProprietaire(@WebParam(name = "proprietaire") ProprietaireRequestDTO proprietaire) {
42         return proprietaireService.addProprietaire(proprietaire);
43     }
44 }
```

### c. Tester les 4 web services

Voici les tests concernant ces web services :

- Rest :
  - ✖ Afficher tous les propriétaires :

# OpenAPI definition v0 OAS3

[/v3/api-docs](#)

Servers

[http://localhost:1230 - Generated server url](#)

## rest-immatriculation-service

[GET](#) /proprietaires/{id}[PUT](#) /proprietaires/{id}[DELETE](#) /proprietaires/{id}[POST](#) /proprietaires/addProprietaire[GET](#) /proprietaires

Parameters

[Cancel](#)

No parameters

Execute

Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:1230/proprietaires' \
  -H 'accept: */*'
```

Request URL

<http://localhost:1230/proprietaires>

Server response

Code	Details
200	<div><div>Response body</div><pre>{   "id": 3,   "nom": "Kautar",   "dateNaissance": "1991-11-13T14:22:30.424+00:00",   "mail": "kautar@gmail.com",   "vehiculesPossedees": [     {       "id": 1,       "numeroMatriculation": "76308368",       "marque": "Jeep",       "puissanceFiscale": 206,       "modele": "Avenger"     }   ] }, {   "id": 2,   "nom": "Sanae",   "dateNaissance": "1956-10-14T14:22:30.425+00:00",   "mail": "sanae@gmail.com",   "vehiculesPossedees": [     {       "id": 2,       "numeroMatriculation": "53904287",       "marque": "Jaguar",       "puissanceFiscale": 189,       "modele": "XJ"     }   ] } }</pre></div>

Response headers

```
connection: keep-alive
content-type: application/json
date: Sun, 11 Jun 2023 13:25:47 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	OK	No links

Media type

\*/\*

Controls Accept header.

[Example Value](#) | [Schema](#)

```
{
  "id": 0,
  "nom": "string",
  "dateNaissance": "2023-06-11T13:25:47.437Z",
  "mail": "string",
  "vehiculesPossedees": [
    {
      "id": 0,
      "numeroMatriculation": "string",
      "marque": "string",
      "puissanceFiscale": 0,
      "modele": "string",
      "proprietaire": "string"
    }
  ]
}
```

Schemas

[ProprietaireRequestDTO](#) >[ProprietaireResponseDTO](#) >[Proprietaire](#) >[Vehicule](#) >

✕ Chercher un propriétaire par son identifiant :

rest-immatriculation-service

GET /proprietaires/{id}

Parameters

Name	Description
id * required	
integer(\$int64)	
(path)	

Execute

Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:1230/proprietaires/1' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:1230/proprietaires/1
```

Server response

Code	Details
200	<div><div>Response body</div><div><pre>{   "id": 1,   "nom": "Kwatar",   "dateNaissance": "1981-11-13T14:22:30.424+00:00",   "mail": "Kwatar@mail.com",   "vehiculesPossedees": [     {       "id": 1,       "numeroMatriculation": "76308368",       "marque": "Jeep",       "puissanceFiscale": 200,       "modele": "Avenger"     }   ] }</pre></div><div>Download</div></div> <div><div>Response headers</div><div><pre>connection: keep-alive content-type: application/json date: Sun, 11 Jun 2023 13:24:21 GMT keep-alive: timeout=60 transfer-encoding: chunked</pre></div></div>

Responses

Code	Description	Links
200	OK	No links

Media type

\*/\*

Controls Accept header

Example Value | Schema

```
{
  "id": 0,
  "nom": "string",
  "dateNaissance": "2023-06-11T13:24:21.828Z",
  "mail": "string",
  "vehiculesPossedees": [
    {
      "id": 0,
      "numeroMatriculation": "string",
      "marque": "string",
      "puissanceFiscale": 0,
      "modele": "string",
      "proprietaire": "string"
    }
  ]
}
```

✕ Ajouter un nouveau propriétaire :



rest-immatriculation-service

GET /proprietaires/{id}

Parameters

Name Description  
id <sup>required</sup>  
Integer (int32) 7  
(path)

Execute Clear

Responses

Curl

curl -X 'GET' \
 'http://localhost:1236/proprietaires/7' \
 -H 'accept: \*/\*'

Request URL

http://localhost:1236/proprietaires/7

Server response

Code Details

200

Response body

{
 "id": 7,
 "nom": "Meline",
 "dateNaissance": "1973-06-11T13:31:04.389Z",
 "mail": "meline@gmail.com",
 "matriculeProcedure": {}
}

Download

Response headers

connection: keep-alive  
content-type: application/json  
date: Sun, 11 Jun 2017 11:31:03 GMT  
keep-alive: timeout=60  
transfer-encoding: chunked

Responses

Code Description Links

200 OK 200 OK

Media type

\*/\*

Content accept header

Example Value : Schema

{
 "id": 7,
 "nom": "Meline",
 "dateNaissance": "1973-06-11T13:31:04.389Z",
 "mail": "meline@gmail.com",
 "matriculeProcedure": {
 "id": 1,
 "nom": "Meline",
 "dateNaissance": "1973-06-11T13:31:04.389Z",
 "mail": "meline@gmail.com",
 "matriculeProcedure": {}
 }
}

PUT /proprietaires/{id}

DELETE /proprietaires/{id}

POST /proprietaires/addProprietaire

Parameters

No parameters

Request body <sup>required</sup>

application/json

{
 "nom": "Meline",
 "dateNaissance": "1973-06-11T13:31:04.389Z",
 "mail": "meline@gmail.com"
}

Execute Clear

Responses

Curl

curl -X 'POST' \
 'http://localhost:1236/proprietaires/addProprietaire' \
 -H 'accept: \*/\*' \
 -H 'Content-Type: application/json' \
 -d '{
 "nom": "Meline",
 "dateNaissance": "1973-06-11T13:31:04.389Z",
 "mail": "meline@gmail.com"
 }'

Request URL

http://localhost:1236/proprietaires/addProprietaire

Server response

Code Details

200

Response body

{
 "id": 7,
 "nom": "Meline",
 "dateNaissance": "1973-06-11T13:31:04.389Z",
 "mail": "meline@gmail.com"
}

Download

Response headers

connection: keep-alive  
content-type: application/json  
date: Sun, 11 Jun 2017 11:31:04 GMT  
keep-alive: timeout=60  
transfer-encoding: chunked

Responses

Code Description Links

200 OK 200 OK

Media type

\*/\*

Content accept header

Example Value : Schema

{
 "id": 7,
 "nom": "Meline",
 "dateNaissance": "1973-06-11T13:31:04.389Z",
 "mail": "meline@gmail.com"
}

✗ Mettre à jour un propriétaire :

PUT /proprietaires/{id}

Parameters

Cancel

Reset

NameDescription

idrequiredInteger (int32)7

(optional)

Request bodyrequiredapplication/json

```
{
  "nom": "Hélène",
  "dateNaissance": "2000-07-11T14:04:47.781Z",
  "mail": "hine@gmail.com"
}
```

Execute

Clear

Responses

200

Content

```
curl -X 'PUT' \
  http://localhost:1230/proprietaires/? \
  -H 'accept: */*' \
  -H 'content-type: application/json' \
  -d '{
    "nom": "Hélène",
    "dateNaissance": "2000-07-11T14:04:47.781Z",
    "mail": "hine@gmail.com"
  }'
```

Request URL

```
http://localhost:1230/proprietaires/
```

Server response

CodeDetails

200

Response body

```
{
  "id": 7,
  "nom": "Hélène",
  "dateNaissance": "2000-07-11T14:04:47.781000000",
  "mail": "hine@gmail.com"
}
```

Response headers

```
connection: keep-alive
content-type: application/json
date: Sun, 11 Jun 2023 14:05:10 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

Responses

CodeDescriptionLinks

200OK

Media type

\*/\*

Content Accept Header

Example ValueSchema

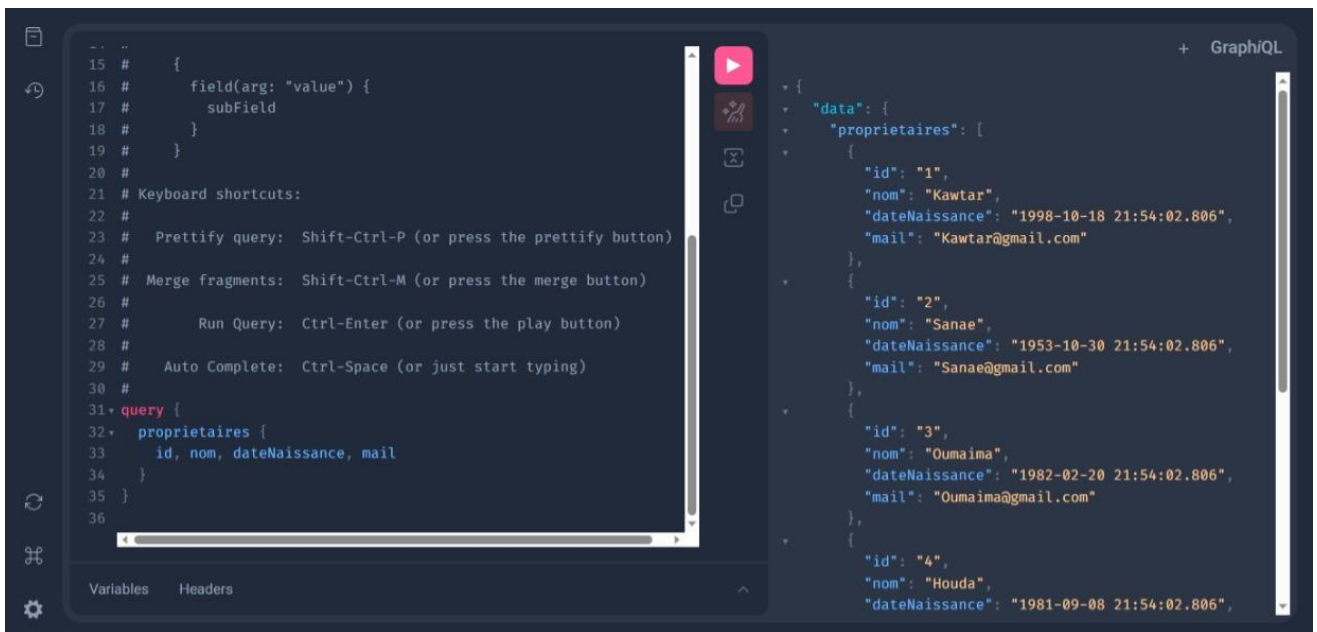
```
{
  "id": 7,
  "nom": "Hélène",
  "dateNaissance": "2023-06-11T14:05:10.898Z",
  "mail": "hine@gmail.com"
}
```

- ✕ Supprimer un propriétaire :

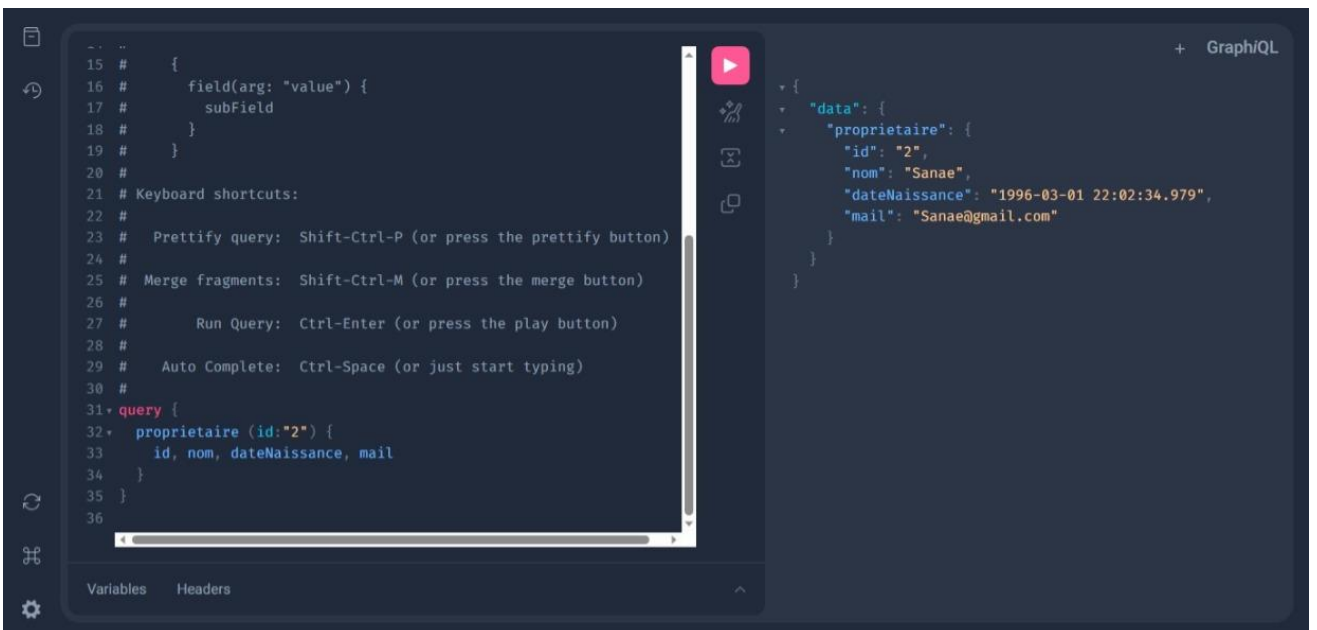
- GraphQL :

- GraphQL :

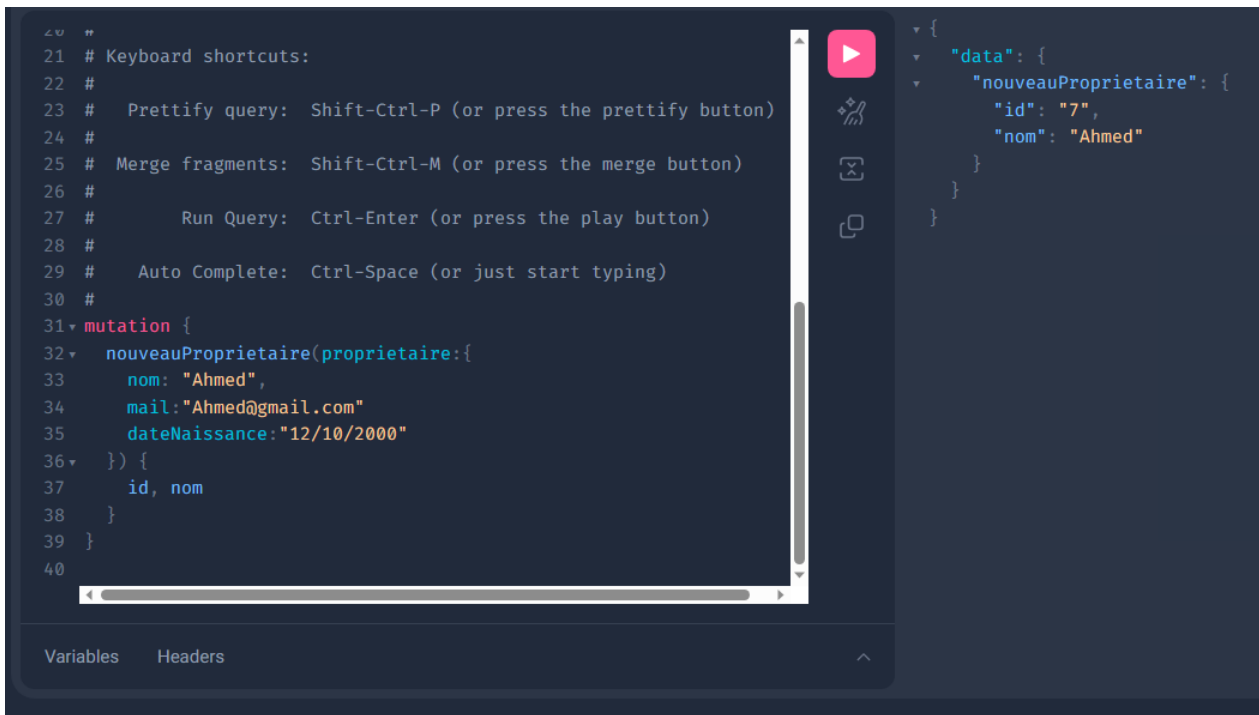
✖ Afficher tous les propriétaires :



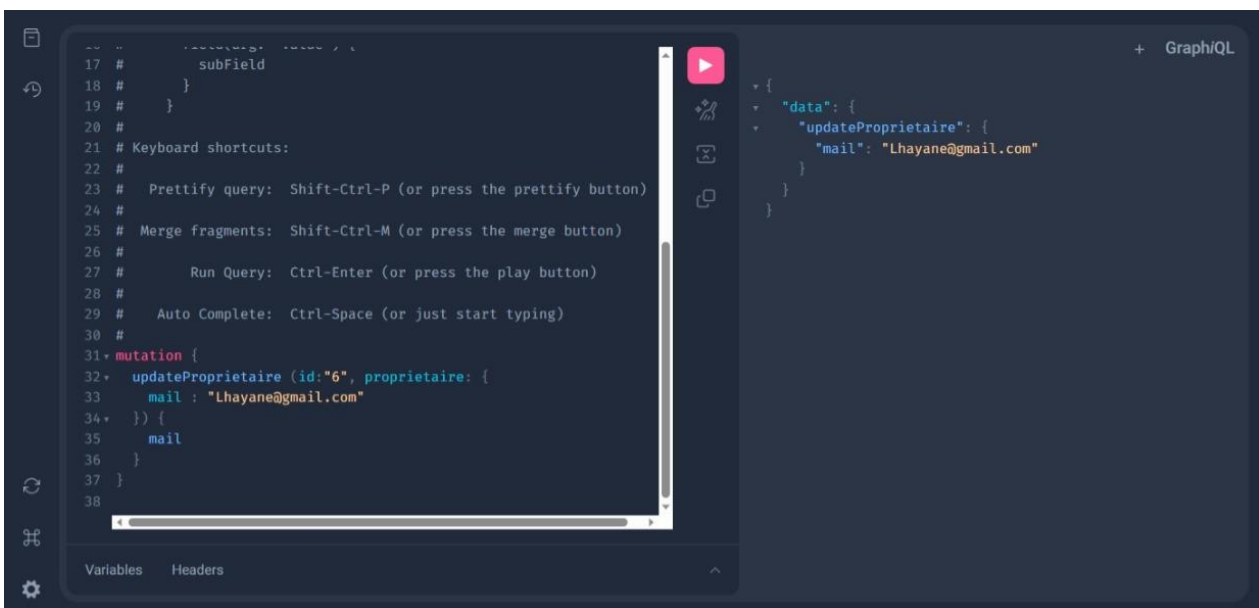
✖ Chercher un propriétaire par son identifiant :



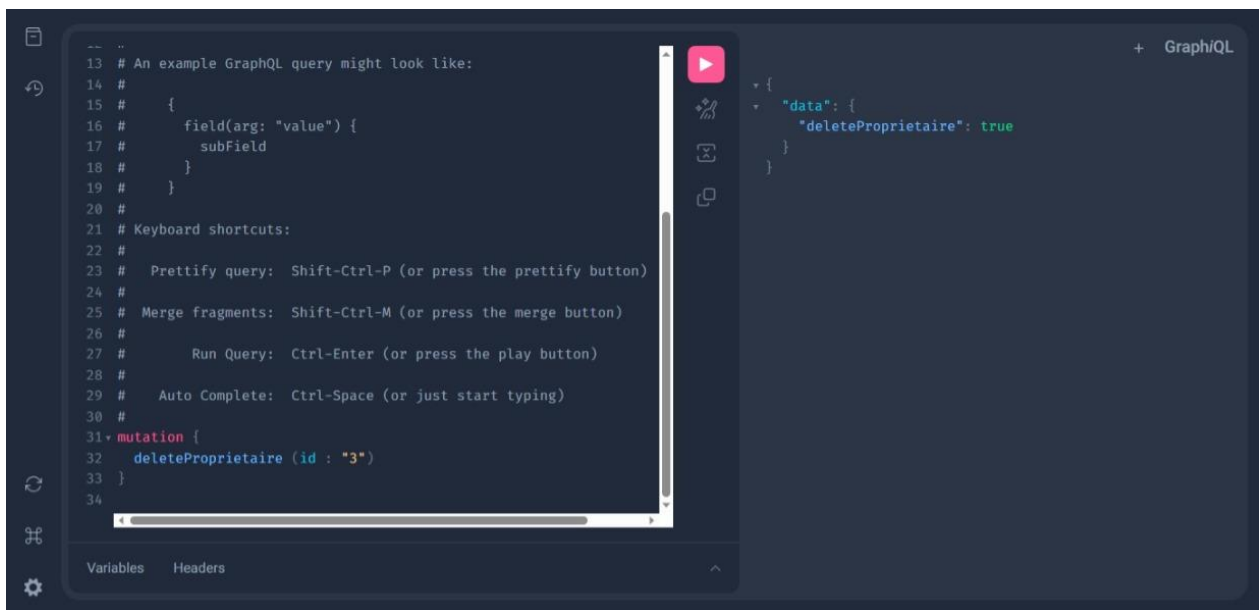
✖ Ajouter un nouveau propriétaire :



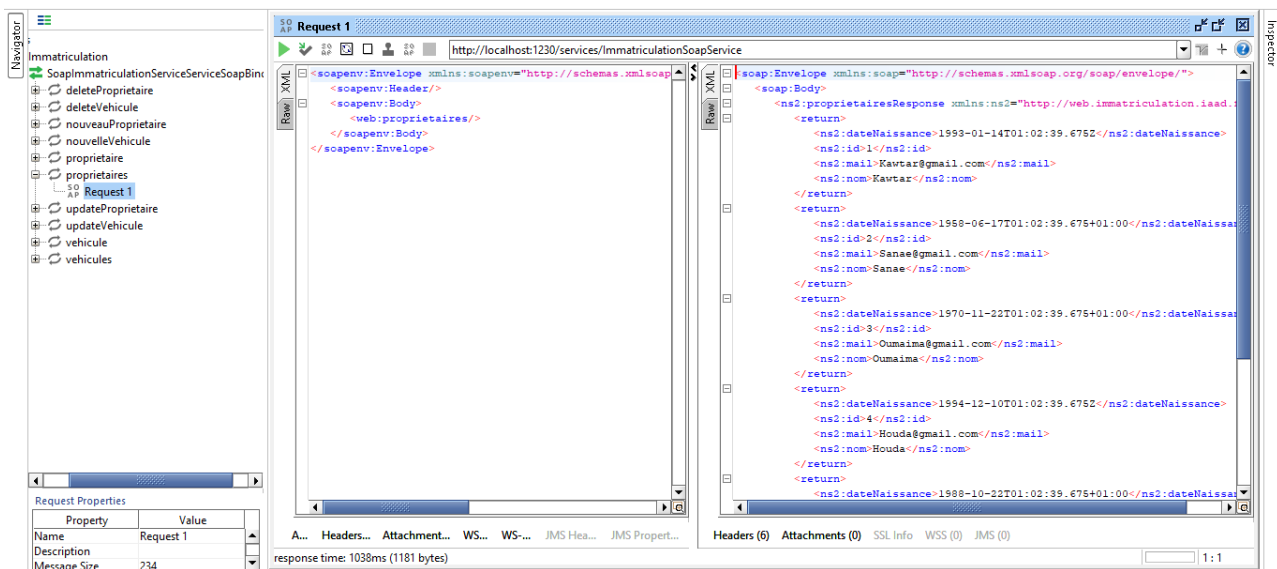
✖ Mettre à jour un propriétaire :



✖ Supprimer un propriétaire :



- **GRPC :**
  - ✗ Afficher tous les propriétaires :
  - ✗ Chercher un propriétaire par son identifiant :
  - ✗ Ajouter un nouveau propriétaire :
  - ✗ Mettre à jour un propriétaire :
  - ✗ Supprimer un propriétaire :
- **SOAP :**
  - ✗ Afficher tous les propriétaires :



- ✗ Chercher un propriétaire par son identifiant :

Immatriculation

- SoapImmatriculationServiceServiceSoapBin
- deleteProprietaire
- deleteVehicule
- nouveauProprietaire
- nouvelleVehicule
- proprietaire
  - Request 1
  - proprietaires
  - Request 1
  - updateProprietaire
  - updateVehicule
  - vehicule
  - vehicules

Request Properties

Property	Value
Name	Request 1
Description	
Message Size	307

Request 1

http://localhost:1230/services/ImmatriculationSoapService

Raw XML

```
<?xml version='1.0' encoding='UTF-8'>
<soap:Envelope xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'>
  <soap:Header/>
  <soap:Body>
    <web:proprietaire xmlns:web='http://web.immatriculation.isad.fs/'>
      <!--Optional:-->
      <id>2</id>
    </web:proprietaire>
  </soap:Body>
</soap:Envelope>
```

Response

```
<?xml version='1.0' encoding='UTF-8'>
<soap:Envelope xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'>
  <soap:Body>
    <ns2:proprietaireResponse xmlns:ns2='http://web.immatriculation.isad.fs/'>
      <return>
        <ns2:dateNaissance>1958-06-17T01:02:39.675+01:00</ns2:dateNaissance>
        <ns2:id>2</ns2:id>
        <ns2:mail>Sanae@gmail.com</ns2:mail>
        <ns2:nom>Sanae</ns2:nom>
      </return>
    </ns2:proprietaireResponse>
  </soap:Body>
</soap:Envelope>
```

response time: 52ms (373 bytes)

response time: 1038ms (1181 bytes)

✖ Ajouter un nouveau propriétaire :

Immatriculation

- SoapImmatriculationServiceServiceSoapBin
- deleteProprietaire
- deleteVehicule
- nouveauProprietaire
- nouvelleVehicule
- Request 1
- proprietaires
- Request 1
- updateProprietaire
- updateVehicule
- vehicule
- vehicules

Request Properties

Property	Value
Name	Request 1
Description	
Message Size	611

Request 1

http://localhost:1230/services/ImmatriculationSoapService

Raw XML

```
<?xml version='1.0' encoding='UTF-8'>
<soap:Envelope xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'>
  <soap:Header/>
  <soap:Body>
    <web:nouveauProprietaire xmlns:web='http://web.immatriculation.isad.fs/'>
      <!--Optional:-->
      <!--Optional:-->
      <web:dateNaissance>1970-10-17T10:06:00.928+00:00</web:dateNaissance>
      <!--Optional:-->
      <web:mail>hajji@gmail.com</web:mail>
      <!--Optional:-->
      <web:nom>amina</web:nom>
    </web:nouveauProprietaire>
  </soap:Body>
</soap:Envelope>
```

Response

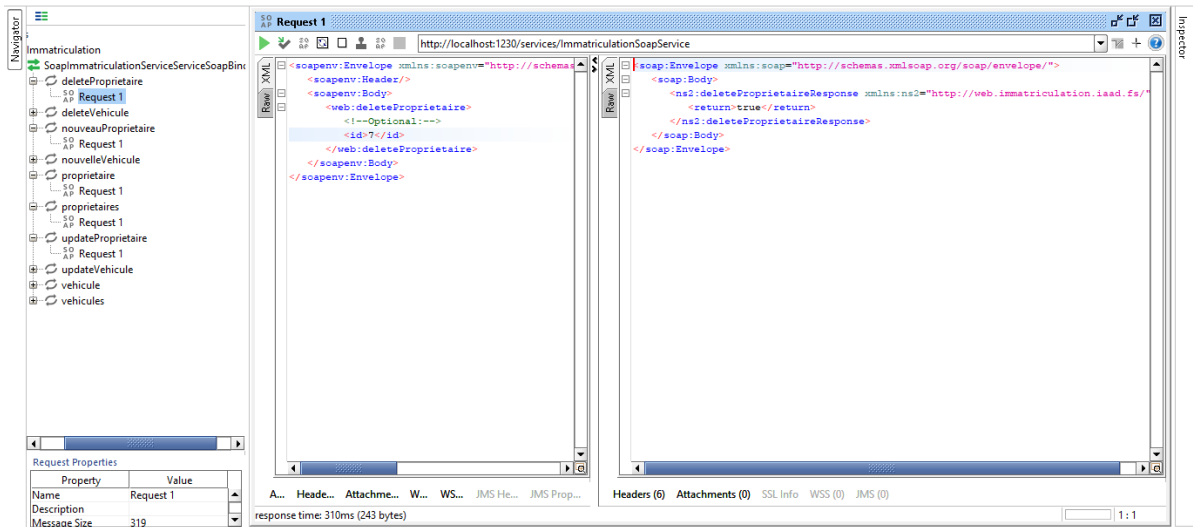
```
<?xml version='1.0' encoding='UTF-8'>
<soap:Envelope xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'>
  <soap:Body>
    <ns2:nouveauProprietaireResponse xmlns:ns2='http://web.immatriculation.isad.fs/'>
      <return>
        <ns2:dateNaissance>1970-10-17T11:06:00.928+01:00</ns2:dateNaissance>
        <ns2:id>7</ns2:id>
        <ns2:mail>hajji@gmail.com</ns2:mail>
        <ns2:nom>amina</ns2:nom>
      </return>
    </ns2:nouveauProprietaireResponse>
  </soap:Body>
</soap:Envelope>
```

Auth Headers (0) Attachments (0) WS-A WS-RM JMS Headers JMS Property (0)

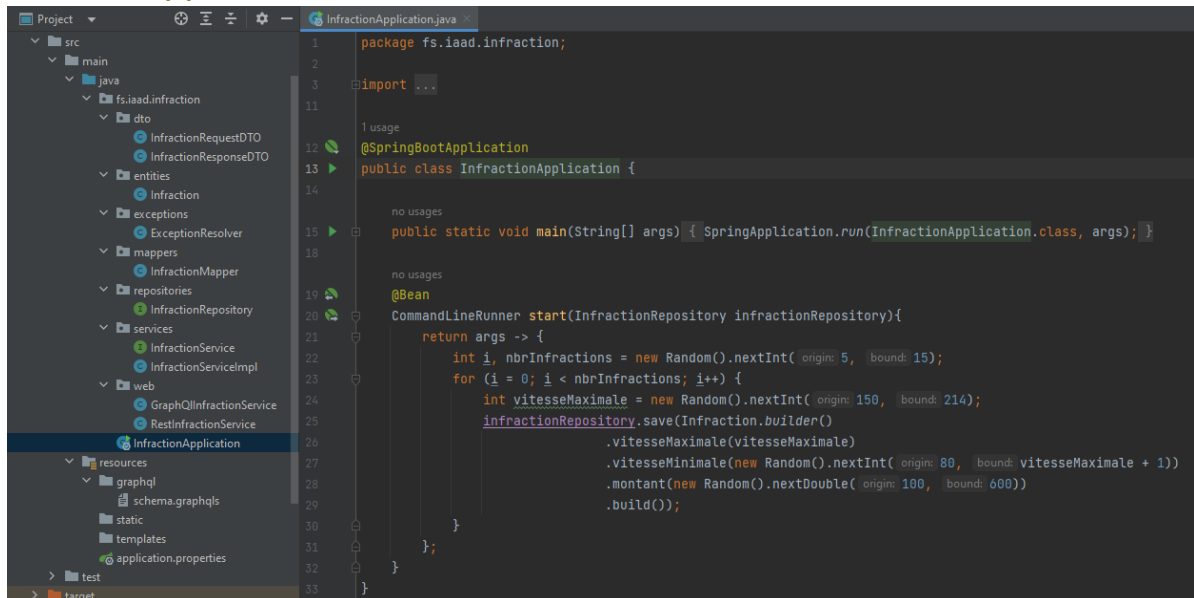
response time: 133ms (387 bytes)

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

## ✕ Supprimer un propriétaire :



## 4. Développer le micro-service Infractions



## 5. Développer le micro-service Radar



```

25 @SpringBootApplication
26 @EnableFeignClients
27 public class RadarApplication {
28
29     no usages
30     public static void main(String[] args) { SpringApplication.run(RadarApplication.class, args); }
31
32     no usages
33     @Bean
34     CommandLineRunner start(RadarRepository radarRepository, ItemInfractionRepository itemInfractionRepository,
35                             ImmatriculationRestClient immatriculationRestClient, InfractionRestClient infractionRes
36
37     return args -> {
38         int i, nbrRadar = new Random().nextInt( origin: 1, bound: 6);
39         for (i = 0; i < nbrRadar; i++)
40             radarRepository.save(Radar.builder()
41                                 .vitesseMaximale(new Random().nextInt( origin: 50, bound: 100))
42                                 .Latitude(new Random().nextDouble( origin: -90, bound: 91))
43                                 .Longitude(new Random().nextDouble( origin: -180, bound: 181))
44                                 .build());
45
46         List<Vehicule> vehicules = immatriculationRestClient.getAllVehicules();
47         List<Infraction> infractions = infractionRestClient.getAllInfractions();
48
49         List<Radar> nosRadar = radarRepository.findAll();
50
51         //créer un client GRPC
52         ManagedChannel client = ManagedChannelBuilder.forAddress("localhost", 1233)

```

6. Créer une application java qui permet de simuler un radar qui génère aléatoirement des dépassements de vitesses et de les envoyer, via GRPC, au service Radar-Service

Voici l'implémentation de l'application radar :

```

1 package fs.serveur;
2
3 import fs.service.GrpcRadarService;
4 import io.grpc.Server;
5 import io.grpc.ServerBuilder;
6
7 import java.io.IOException;
8
9 no usages
10 public class ServeurGrpc {
11     no usages
12     public static void main(String[] args) {
13         //créer un serveur GRPC
14         Server serveur = ServerBuilder.forPort(1233) //Spécification du port de serveur
15             .addService(new GrpcRadarService()) //publier le webservice
16             .build();
17
18         try {
19             serveur.start(); //lancer le serveur
20             serveur.awaitTermination(); //Attendre la terminaison de l'application
21         } catch (IOException | InterruptedException e) {
22             e.printStackTrace();
23         }
24     }
25 }

```

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

Voici Gateway service :

```

1 package fs.iaad.gateway;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.cloud.client.discovery.ReactiveDiscoveryClient;
6 import org.springframework.cloud.gateway.discovery.DiscoveryClientRouteDefinitionLocator;
7 import org.springframework.cloud.gateway.discovery.DiscoveryLocatorProperties;
8 import org.springframework.context.annotation.Bean;
9
10 @SpringBootApplication
11 public class GatewayApplication {
12     public static void main(String[] args) { SpringApplication.run(GatewayApplication.class, args); }
13
14     @Bean
15     DiscoveryClientRouteDefinitionLocator dynamicRoutes(ReactiveDiscoveryClient reactiveDiscoveryClient, DiscoveryLocatorProperties discoveryLocatorProperties) {
16         return new DiscoveryClientRouteDefinitionLocator(reactiveDiscoveryClient, discoveryLocatorProperties);
17     }
18 }

```

et Eureka service :

```

1 package fs.iaad.eureka;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;
6
7 @SpringBootApplication
8 @EnableEurekaServer
9 public class EurekaApplication {
10     public static void main(String[] args) { SpringApplication.run(EurekaApplication.class, args); }
11 }

```

## DS Replicas

localhost

## Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
SERVICE-GATEWAY	n/a (1)	(1)	UP (1) - <a href="#">DESKTOP-6EK7CUM:service-gateway:1234</a>
SERVICE-IMMATRICULATION	n/a (1)	(1)	UP (1) - <a href="#">DESKTOP-6EK7CUM:service-immatriculation:1230</a>
SERVICE-INFRACTION	n/a (1)	(1)	UP (1) - <a href="#">DESKTOP-6EK7CUM:service-infraction:1231</a>
SERVICE-RADAR	n/a (1)	(1)	UP (1) - <a href="#">DESKTOP-6EK7CUM:service-radar:1232</a>

8. Développer votre application Frontend avec Angular ou React

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

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

J'ai pas pu faire cette partie.