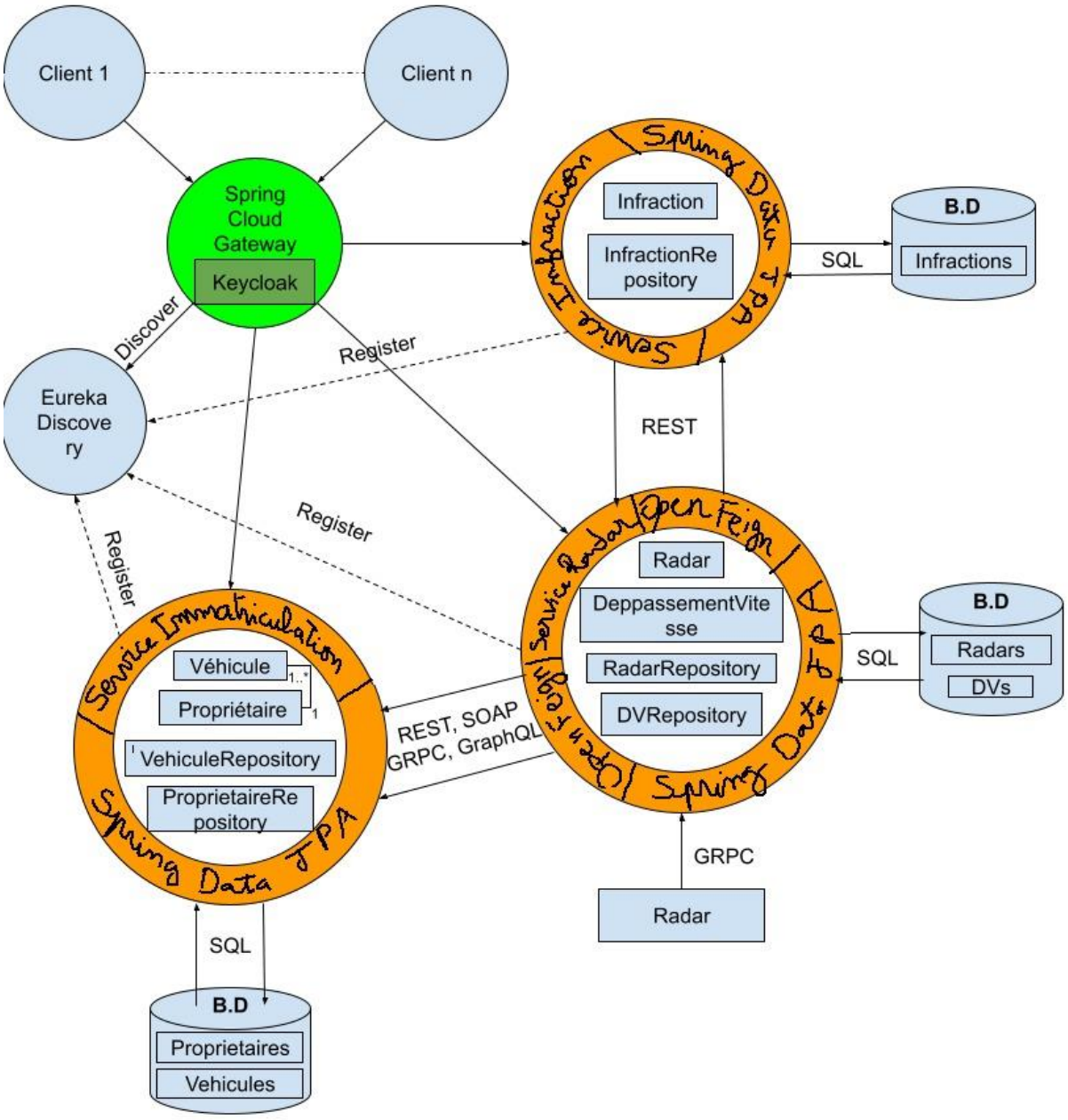
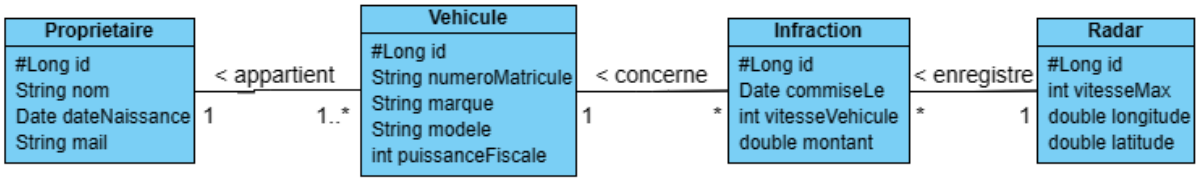
Rapport Contrôle Système Distribué

1. Établir une architecture technique du projet :

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



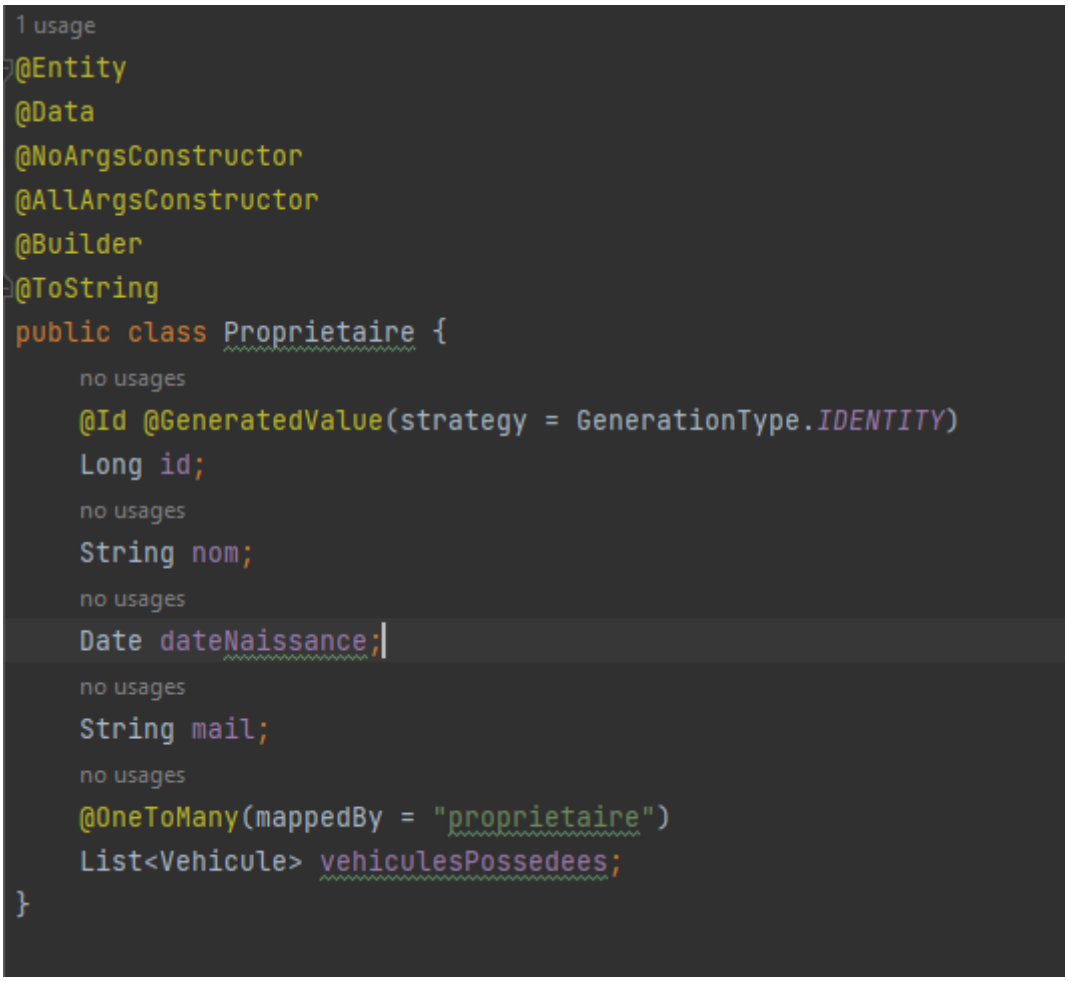
2. Établir un diagramme de classe global du projet



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;
}
```

```java
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);

}
```

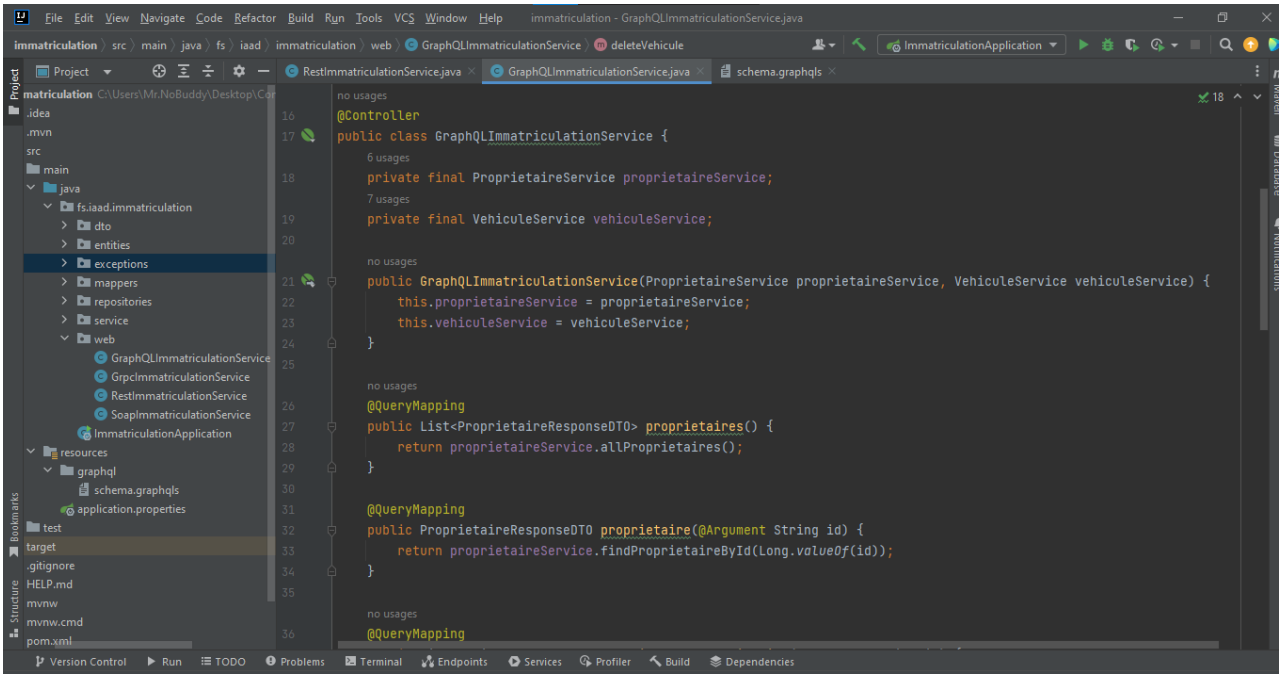
b. Les 4-web services REST, GraphQL, SOAP et GRPC

Voici l'implémentation des quatre web services :

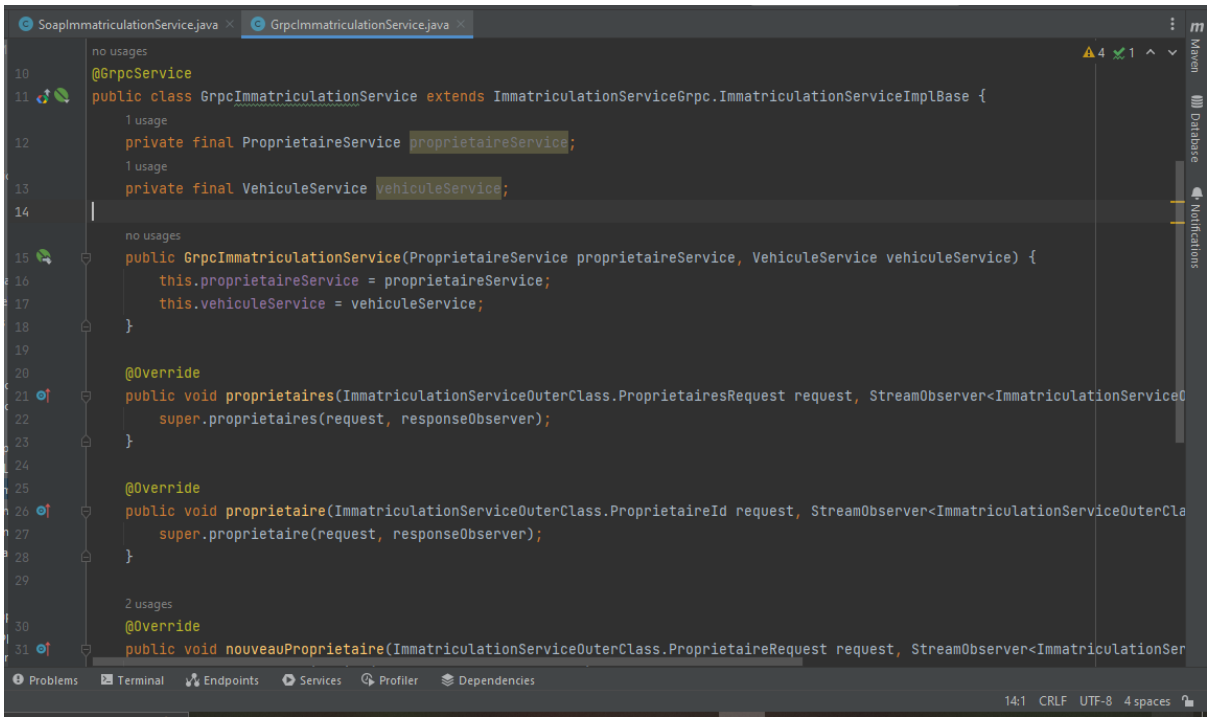
- Rest :

```java
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() {
```

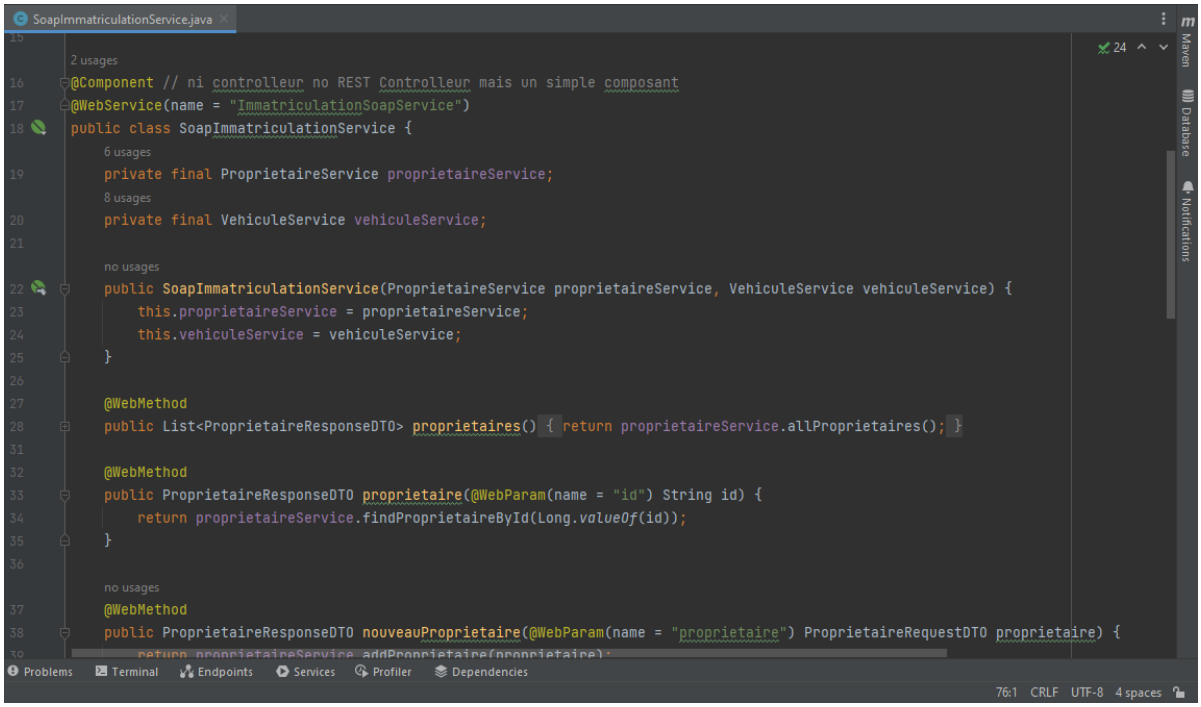
- GraphQl :


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
immatriculation - GraphQLImmatriculationService.java
RestImmatriculationService.java
GraphQLImmatriculationService.java
schema.graphqls
@Controller
public class GraphQLImmatriculationService {
private final ProprietaireService proprietaireService;
private final VehiculeService vehiculeService;
public GraphQLImmatriculationService(ProprietaireService proprietaireService, VehiculeService vehiculeService) {
this.proprietaireService = proprietaireService;
this.vehiculeService = vehiculeService;
}
@QueryMapping
public List<ProprietaireResponseDTO> proprietaires() {
return proprietaireService.allProprietaires();
}
@QueryMapping
public ProprietaireResponseDTO proprietaire(@Argument String id) {
return proprietaireService.findProprietaireById(Long.valueOf(id));
}
@QueryMapping

- GRPC :


SoapImmatriculationService.java
GrpcImmatriculationService.java
@GrpcService
public class GrpcImmatriculationService extends ImmatriculationServiceGrpc.ImmatriculationServiceImplBase {
private final ProprietaireService proprietaireService;
private final VehiculeService vehiculeService;
public GrpcImmatriculationService(ProprietaireService proprietaireService, VehiculeService vehiculeService) {
this.proprietaireService = proprietaireService;
this.vehiculeService = vehiculeService;
}
@Override
public void proprietaires(ImmatriculationServiceOuterClass.ProprietairesRequest request, StreamObserver<ImmatriculationServiceO
super.proprietaires(request, responseObserver);
}
@Override
public void proprietaire(ImmatriculationServiceOuterClass.ProprietaireId request, StreamObserver<ImmatriculationServiceOuterCla
super.proprietaire(request, responseObserver);
}
@Override
public void nouveauProprietaire(ImmatriculationServiceOuterClass.ProprietaireRequest request, StreamObserver<ImmatriculationSer

- SOAP :



c. Tester les 4 web services

Voici les tests concernant ces web services :

- Rest :
  - Afficher tous les propriétaires :

Swagger /v3/api-docs Explore

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

Response body


```
[
  {
    "id": 1,
    "nom": "Kawtar",
    "dateNaissance": "1981-11-13T14:22:30.424+00:00",
    "mail": "Kawtar@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": "XE"
```


Download

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

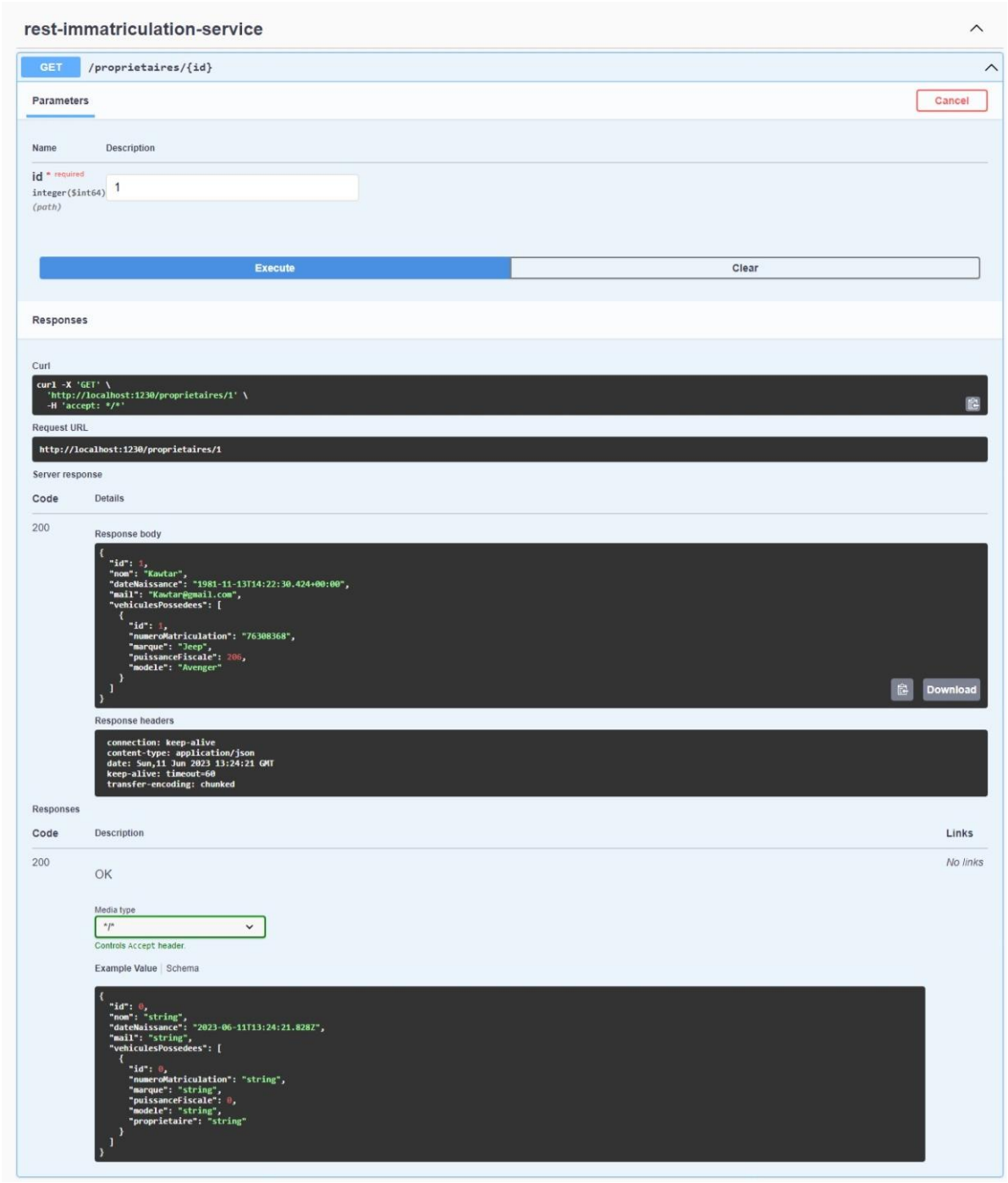
ProprietaireRequestDTO >

ProprietaireResponseDTO >

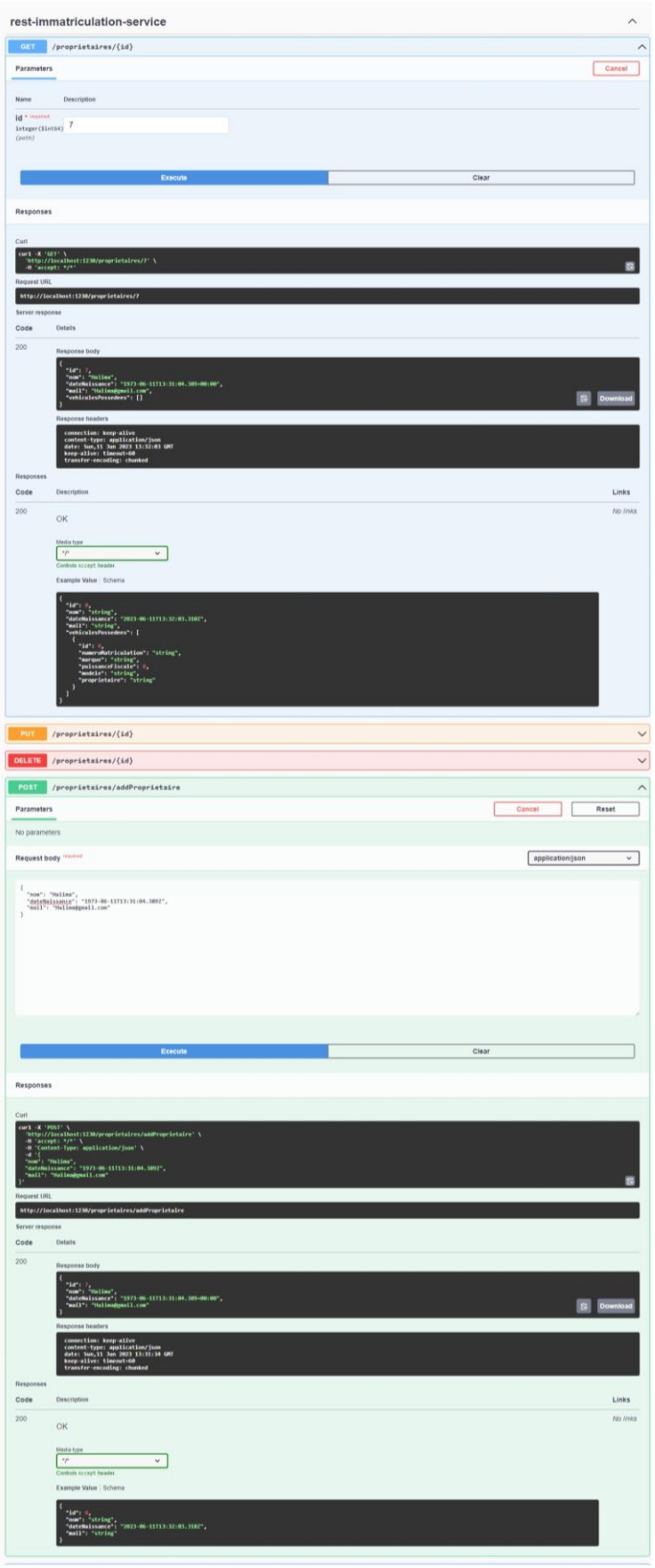
Proprietaire >

Vehicule >

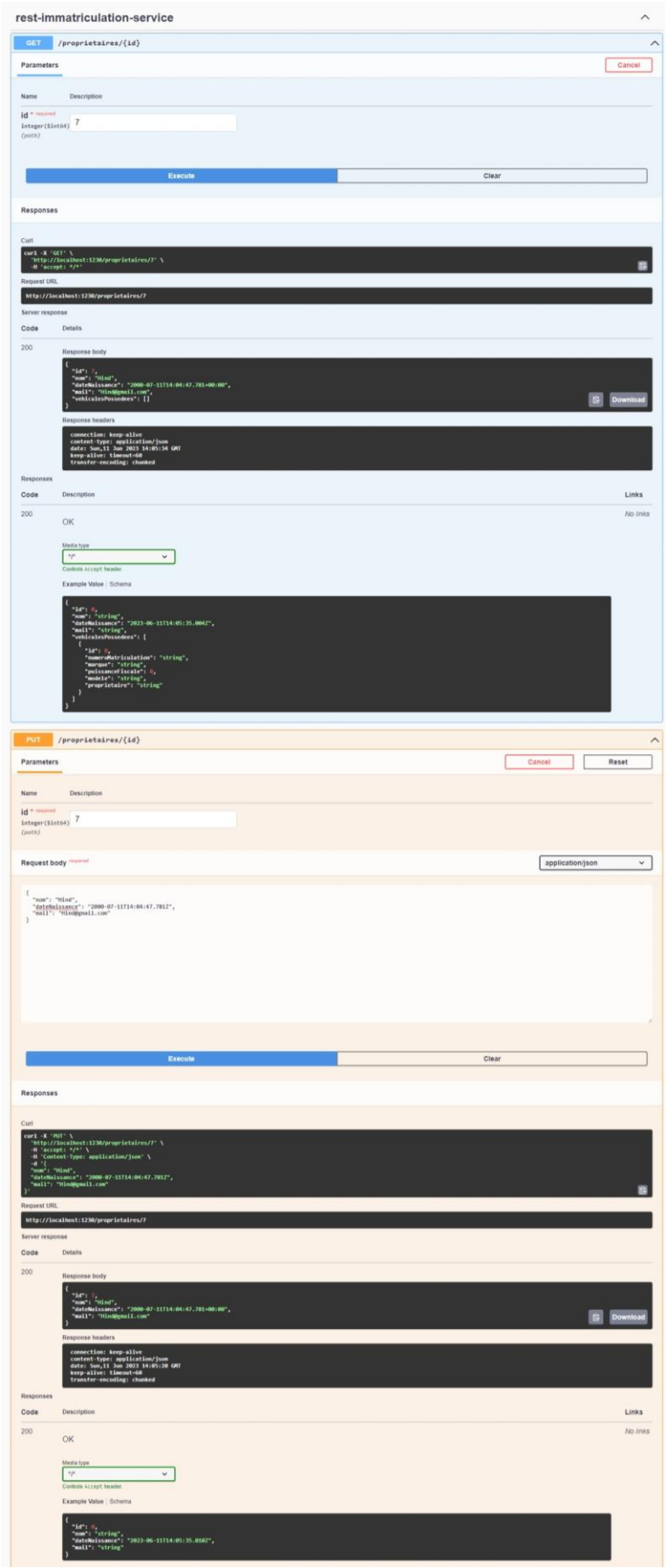
- Chercher un propriétaire par son identifiant :



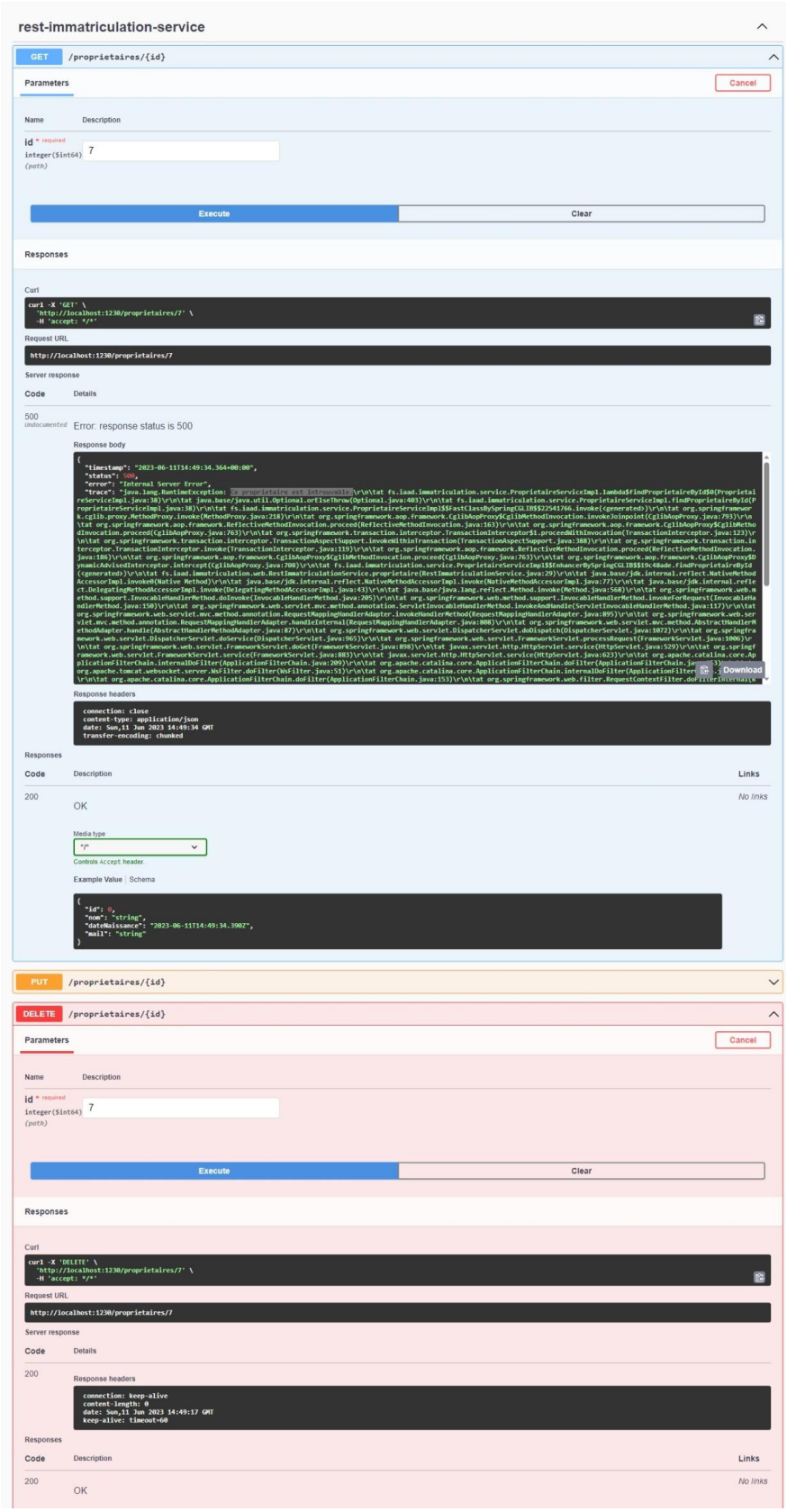
- Ajouter un nouveau propriétaire :



- Mettre à jour un propriétaire :

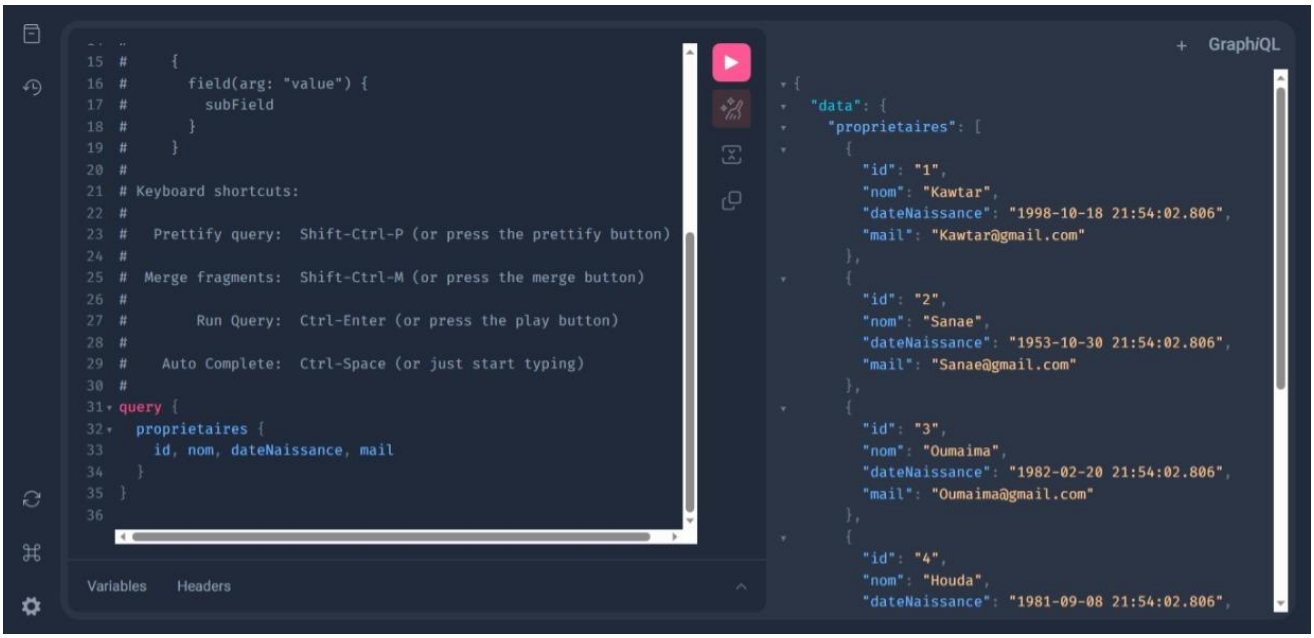


- Supprimer un propriétaire :


rest-immatriculation-service
GET /proprietaires/{id}
Parameters
Cancel
Name
Description
id * required
integer($int64)
(path)
7
Execute
Clear
Responses
Curl
curl -X 'GET' \
'http://localhost:1230/proprietaires/7' \
-H 'accept: */*'
Request URL
http://localhost:1230/proprietaires/7
Server response
Code
Details
500
Undocumented
Error: response status is 500
Response body
Download
Response headers
connection: close
content-type: application/json
date: Sun,11 Jun 2023 14:49:34 GMT
transfer-encoding: chunked
Responses
Code
Description
Links
200
OK
No links
Media type
/
Controls Accept header.
Example Value
Schema
{
"id": 0,
"nom": "string",
"dateNaissance": "2023-06-11T14:49:34.390Z",
"mail": "string"
}
PUT /proprietaires/{id}
DELETE /proprietaires/{id}
Parameters
Cancel
Name
Description
id * required
integer($int64)
(path)
7
Execute
Clear
Responses
Curl
curl -X 'DELETE' \
'http://localhost:1230/proprietaires/7' \
-H 'accept: */*'
Request URL
http://localhost:1230/proprietaires/7
Server response
Code
Details
200
Response headers
connection: keep-alive
content-length: 0
date: Sun,11 Jun 2023 14:49:17 GMT
keep-alive: timeout=60
Responses
Code
Description
Links
200
OK
No links

- GraphQl :

- Afficher tous les propriétaires :



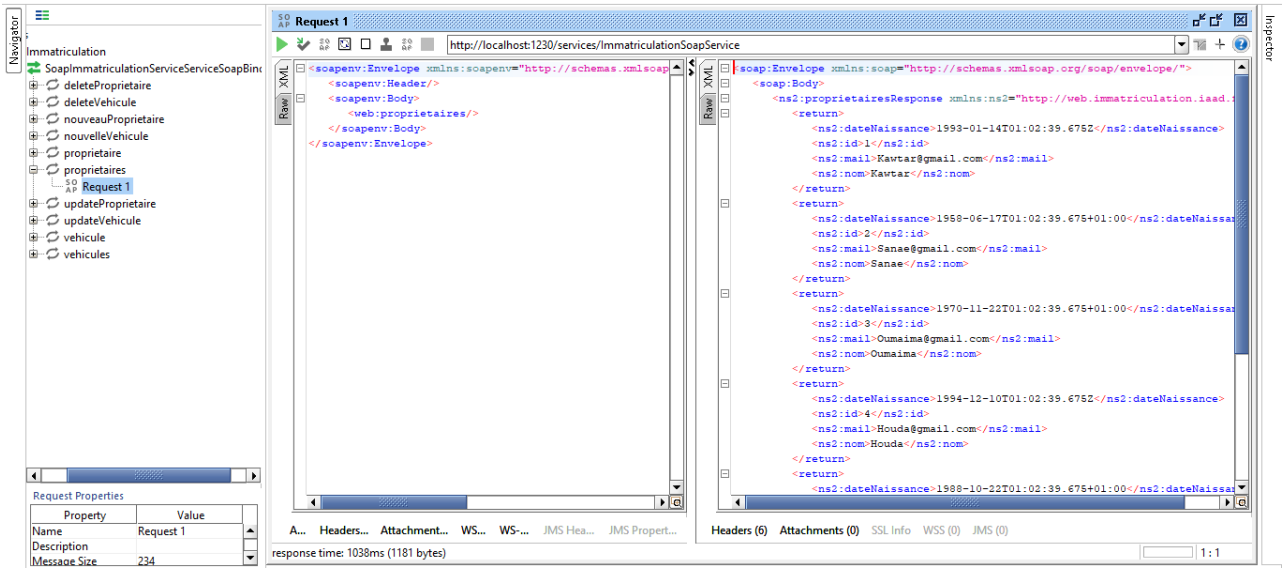
- Chercher un propriétaire par son identifiant :

- Ajouter un nouveau propriétaire :

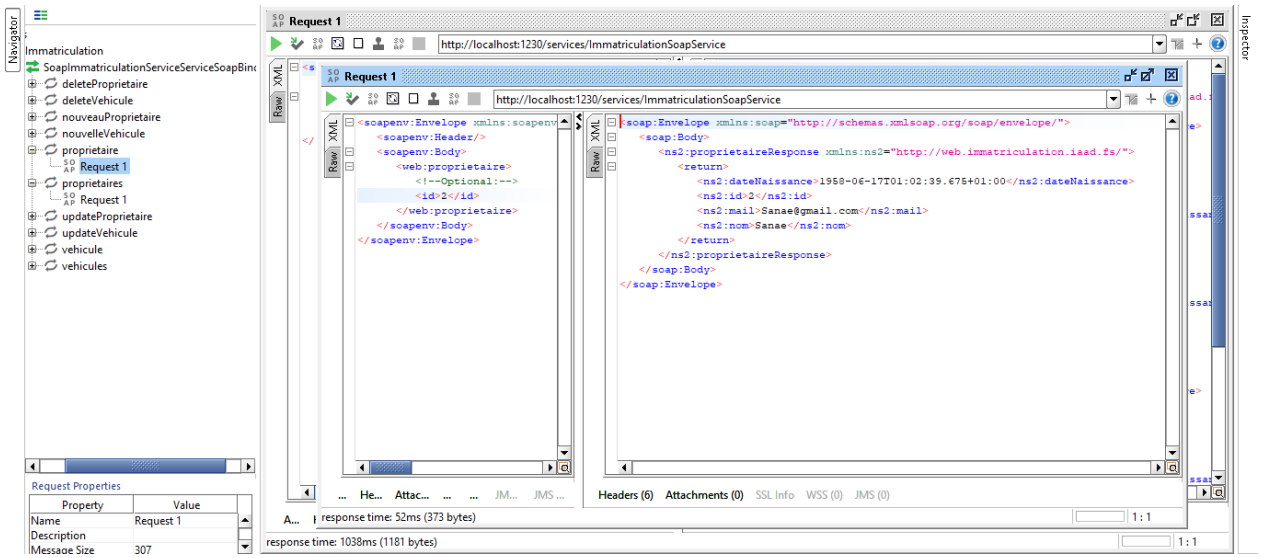
- Mettre à jour un propriétaire :

- Supprimer un propriétaire :

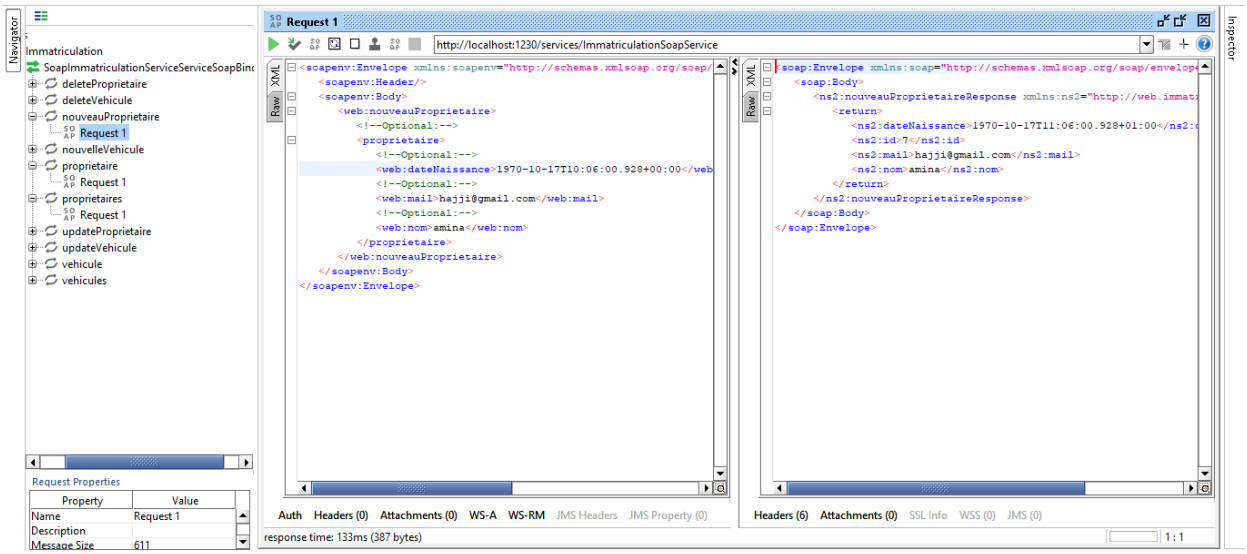
- GRPC :
  - Afficher tous les propriétaires :
  - Chercher un propriétaire par son identifiant :
  - Ajouter un nouveau propriétaire :
  - Mettre à jour un propriétaire :
  - Supprimer un propriétaire :
- SOAP :
  - Afficher tous les propriétaires :



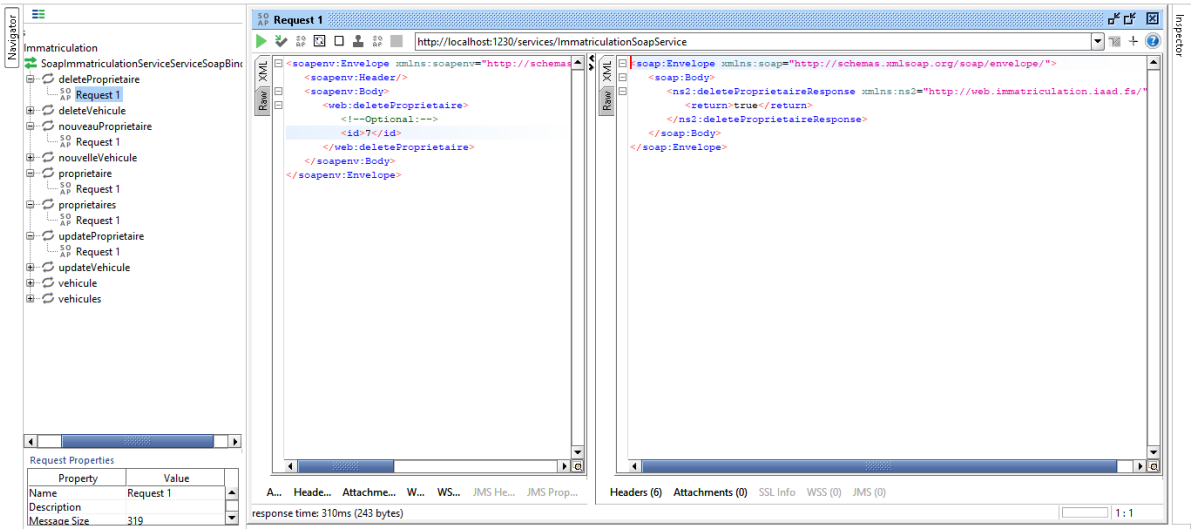
  - Chercher un propriétaire par son identifiant :



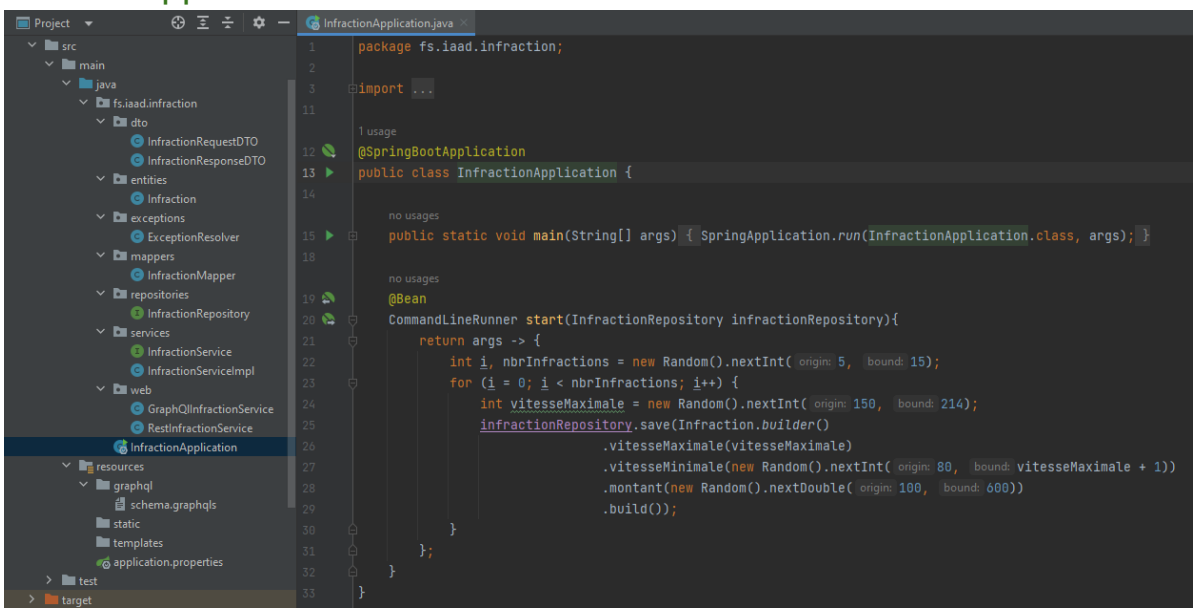
- Ajouter un nouveau propriétaire :



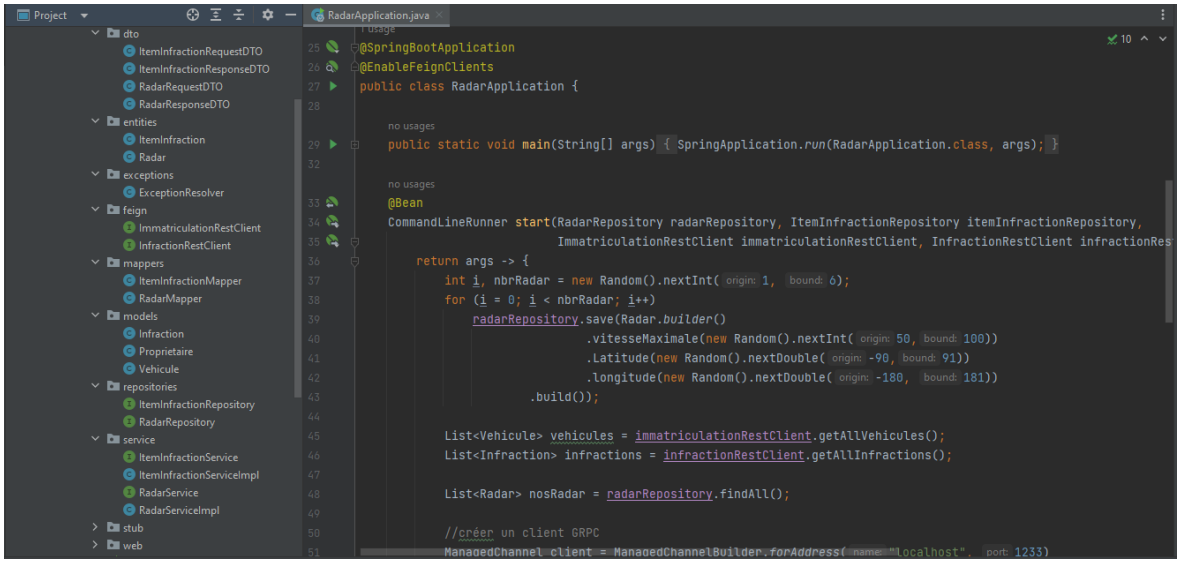
- Supprimer un propriétaire :



4. Développer le micro-service Infractions

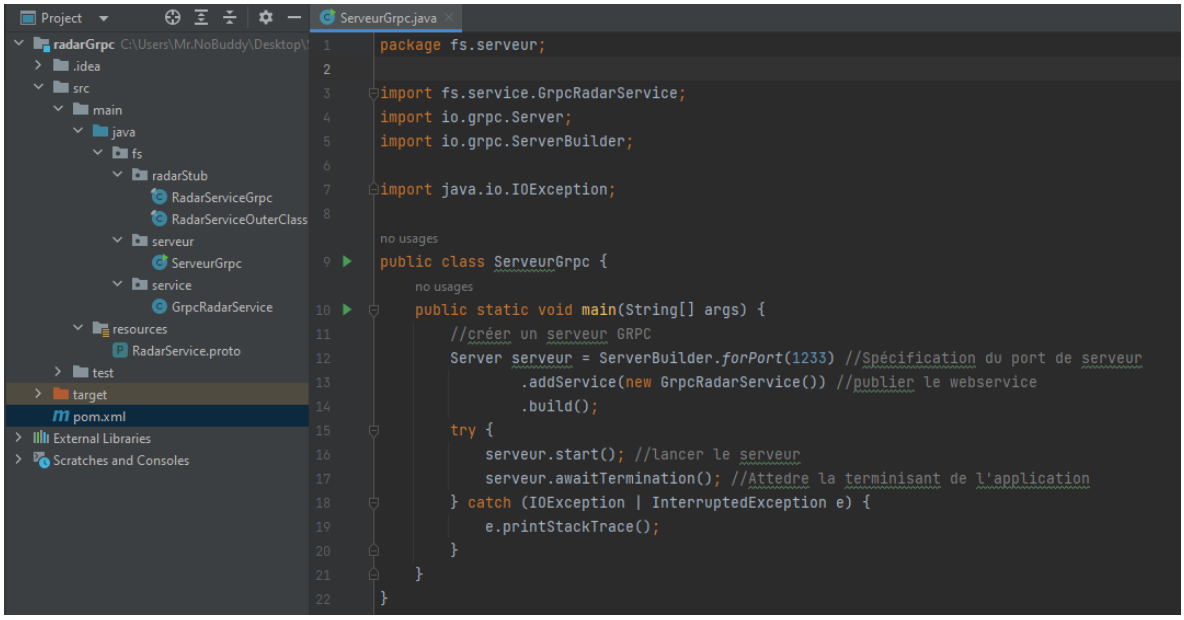


5. Développer le micro-service Radar



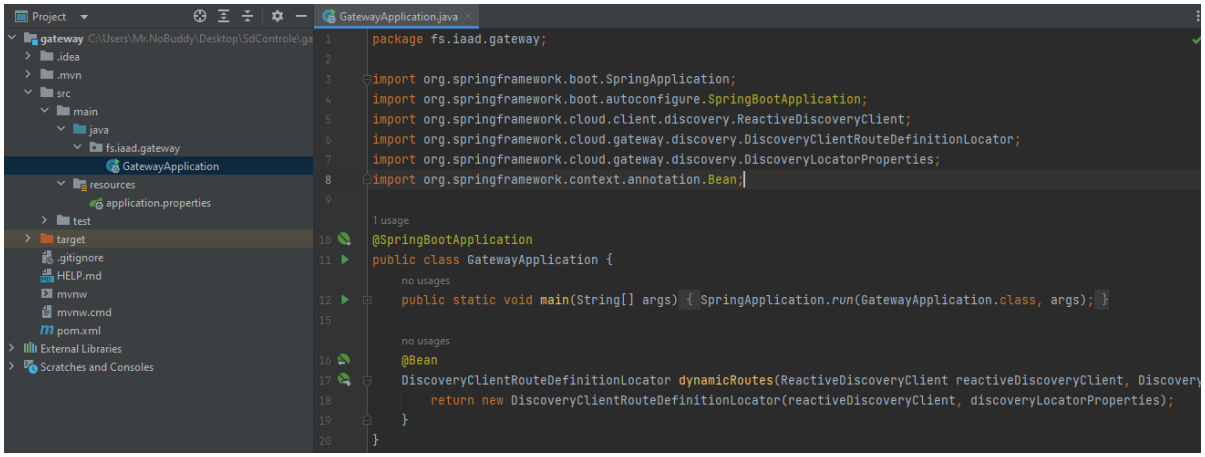
6. Créer une application java qui permet de simuler un radar qui génère aléatoirement des dépassements de vitesses et de les envoyer, via GRPC, au service Radar-Service

Voici l'implémentation de l'application radar :

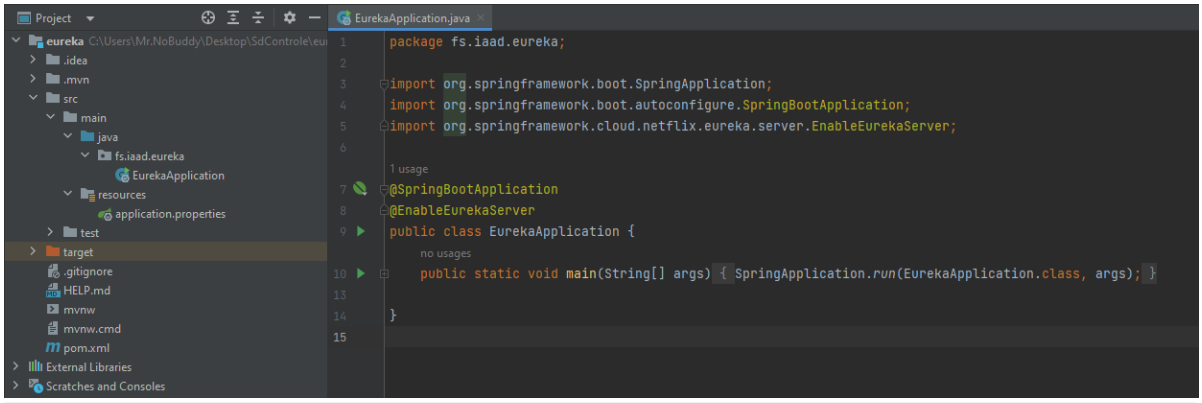


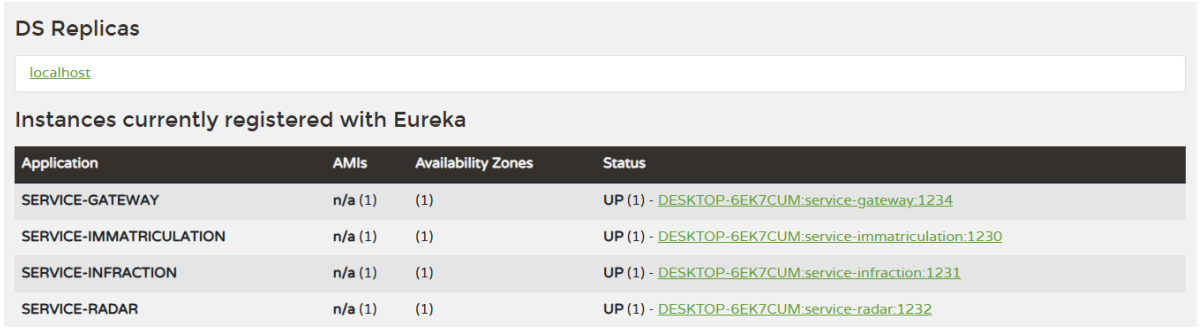
7. Mettre en place les services techniques de l'architecture micro-service (Gateway, Eureka Discovery service)

Voici Gateway service :



et Eureka service :





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.