

**Département Mathématique Informatique**

**Filière :** Master-SDIA

**Modèle :** Programmation Distribuées et Middlewares

**Compte Rendu :**

# **Contrôle et Projet Systèmes Distribués**

**Réaliser par :**

**Année Universitaire : 2022-2023**

* **Abdelkarim AGOUJIL**

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.

1. **architecture technique**

Architecture technique :

+--------------------------------------------------+

| Client (Interface utilisateur) |

+--------------------------------------------------+

| | |

+-----------+ | +-----------------+

| | |

+-----------+ +----------------------+ +-----------------+

| Radar | | Immatriculation | | Infraction |

| Micro- | | Micro-Service | | Micro-Service |

| Service | | | | |

+-----------+ +----------------------+ +-----------------+

| | |

| | |

+-------------------------------------------------+

|

+--------------+

| Database |

+--------------+

Le client (interface utilisateur) communique avec les micro-services via des API REST ou d'autres protocoles appropriés.

Chaque micro-service possède sa propre base de données, qui peut être une base de données spécifique au micro-service ou une base de données partagée.

Les micro-services interagissent entre eux pour échanger des données et effectuer des opérations complexes.

Un système de messagerie asynchrone peut être utilisé pour la communication entre les micro-services, si nécessaire.

Diagramme de classe global :

Voici un exemple de diagramme de classe global pour votre projet :

+----------------------------------+

| 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 |

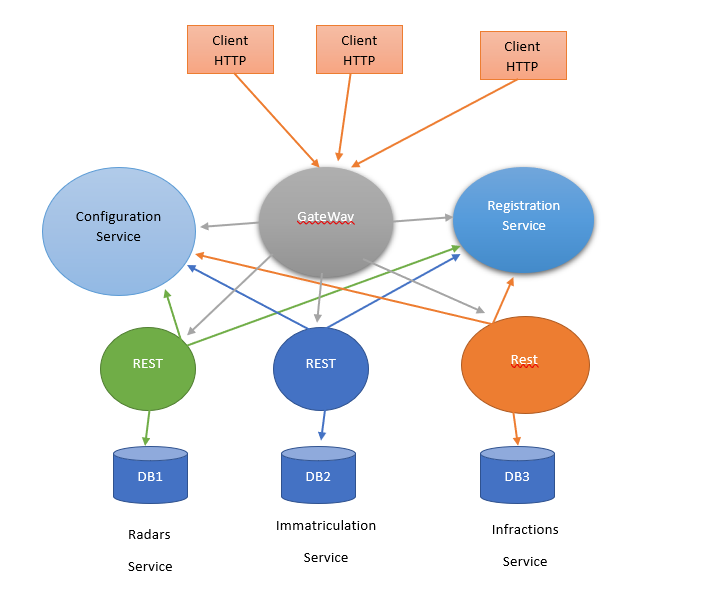
| + getVitesseMaximaleRadar(): int |

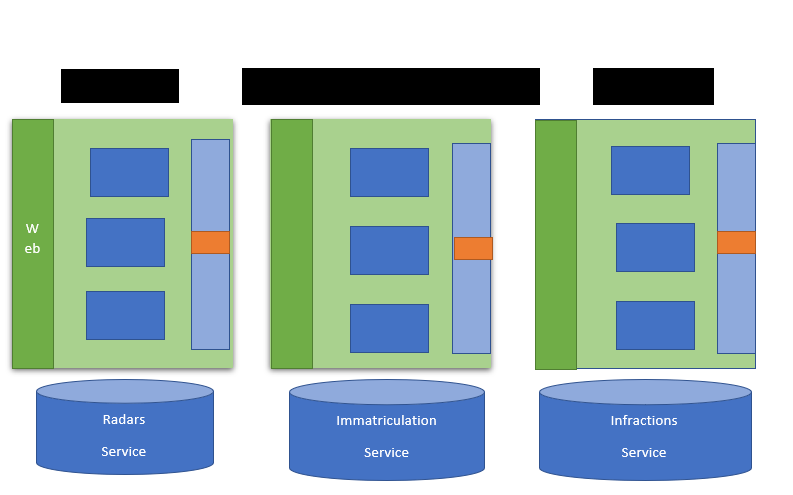
| + getMontant(): double |

+----------------------------------+

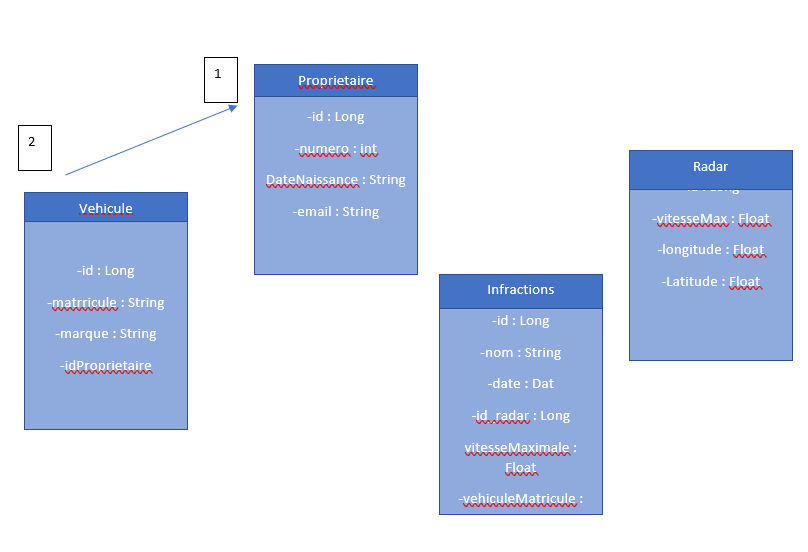
Ce diagramme de classe représente les principales entités de votre système, leurs attributs et les méthodes pour accéder à ces attributs. Les relations entre les classes sont également indiquées, notamment l'association entre les classes Vehicule et Proprietaire, ainsi qu'entre les classes Infraction, Radar et Vehicule.

Il convient de noter que ce diagramme de classe global est un exemple basé sur la description initiale de votre système. Vous devrez adapter le diagramme en fonction des besoins spécifiques de votre projet et des interactions entre les différentes entités.

****

****

1. **diagramme de classe**

****

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

a. Entités JPA et Interface JpaRepository basées sur Spring data

* **Entité Propriétaire**
* package ma.enset.immatriculation\_service.entities;  
    
  import jakarta.persistence.\*;  
  import lombok.AllArgsConstructor;  
  import lombok.Data;  
  import lombok.NoArgsConstructor;  
  import lombok.ToString;  
    
  import java.util.List;  
    
    
  @Entity  
  @Data @AllArgsConstructor @NoArgsConstructor @Builber  
  public class Proprietaire {  
   @Id @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
   private Long id;  
   private String nom;  
   private String dateNaissance;  
   private String email;  
   @OneToMany(mappedBy = "proprietaire")
* @Transient  
   private List<Vehicule> vehicules;  
  }
* **Entité Véhicule**
* package ma.enset.immatriculation\_service.entities;  
    
  import com.fasterxml.jackson.annotation.JsonProperty;  
  import jakarta.persistence.\*;  
  import lombok.AllArgsConstructor;  
  import lombok.Data;  
  import lombok.NoArgsConstructor;  
  import lombok.ToString;  
    
  @Entity  
  @Data @AllArgsConstructor @NoArgsConstructor @Builder  
  public class Vehicule {  
   @Id  
   @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
   private Long id;  
   private String matrecule;  
   private String marque;  
   private String fiscalePuissance;  
   private String model;  
   @ManyToOne  
   @JsonProperty(access = JsonProperty.Access.*WRITE\_ONLY*)  
   private Proprietaire proprietaire;  
  }
* **Propriétaire Repository**

package ma.enset.immatriculation\_service.repositories;  
  
  
import ma.enset.immatriculation\_service.entities.Proprietaire;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.rest.core.annotation.RepositoryRestResource;  
  
  
@RepositoryRestResource  
public interface ProprietaireRepository extends JpaRepository<Proprietaire,Long> {

Optional<Proprietaire> findByNom(String nom);

}

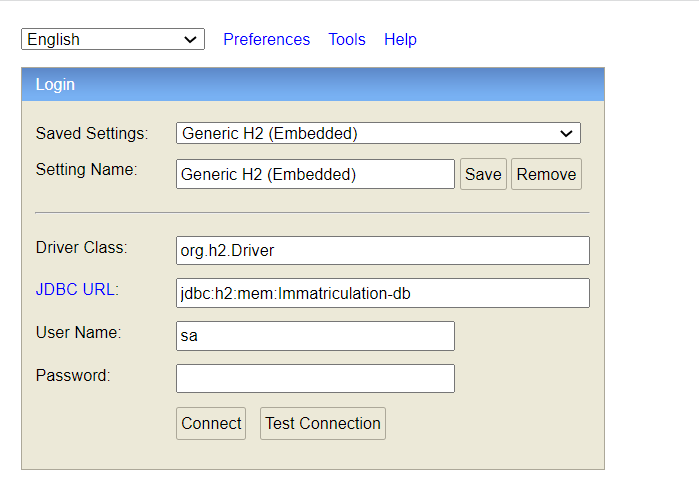
* **Véhicule Repository**
* package ma.enset.immatriculation\_service.repositories;  
    
    
  import ma.enset.immatriculation\_service.entities.Vehicule;  
  import org.springframework.data.jpa.repository.JpaRepository;  
  import org.springframework.data.rest.core.annotation.RepositoryRestResource;  
    
  import java.util.List;  
    
  @RepositoryRestResource  
  public interface VehiculeRepository extends JpaRepository<Vehicule,Long> {  
     
  }
* **ImmatriculationServiceApplication**

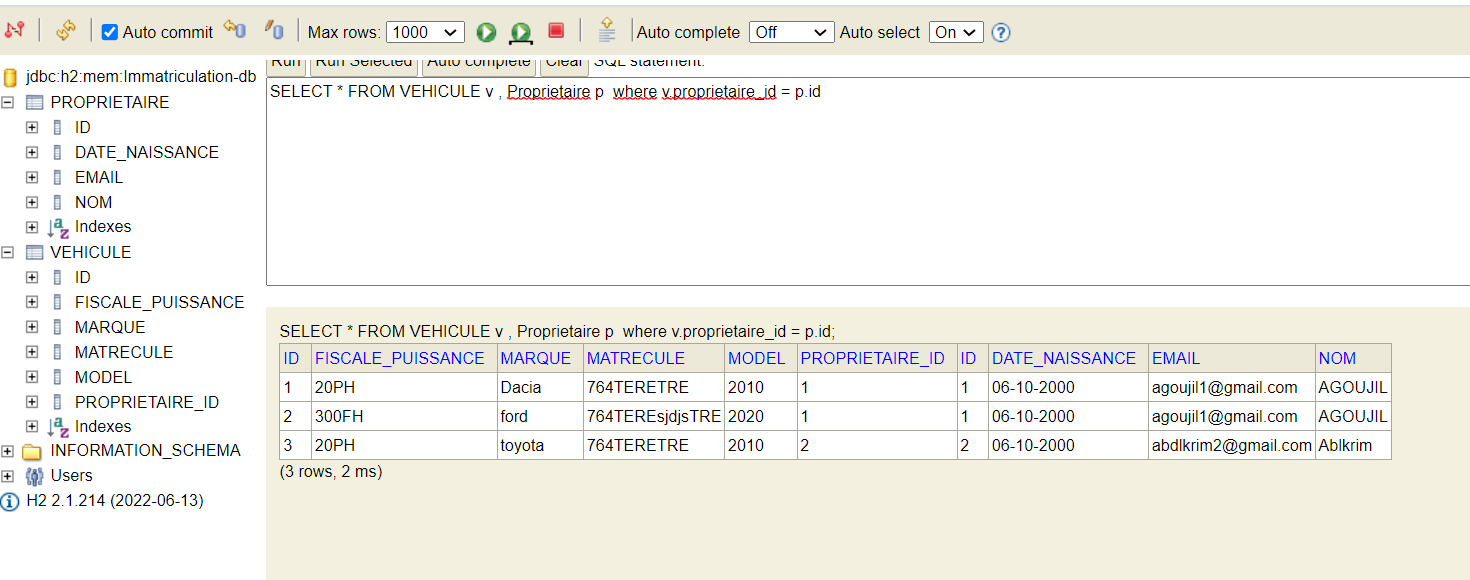
Pour tester les deux entités j’ai créé deux propriétaires et trois véhicule, j’ais associer les deux premiers véhicules et la troisième a le deuxième propriétaire :

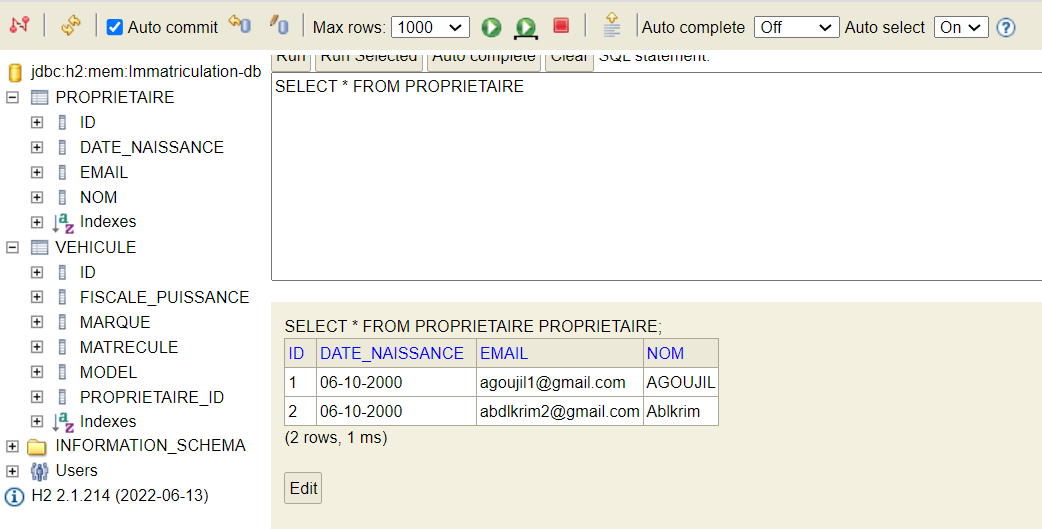
package ma.enset.immatriculation\_service;  
  
import ma.enset.immatriculation\_service.entities.Proprietaire;  
import ma.enset.immatriculation\_service.entities.Vehicule;  
import ma.enset.immatriculation\_service.repositories.ProprietaireRepository;  
import ma.enset.immatriculation\_service.repositories.VehiculeRepository;  
import org.springframework.boot.CommandLineRunner;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.context.annotation.Bean;  
import org.springframework.data.rest.core.config.RepositoryRestConfiguration;  
  
import java.util.ArrayList;  
  
@SpringBootApplication  
public class ImmatriculationServiceApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(ImmatriculationServiceApplication.class, args);  
 }  
 @Bean  
 CommandLineRunner start(ProprietaireRepository proprietaireRepository, VehiculeRepository vehiculeRepository, RepositoryRestConfiguration repositoryRestConfiguration){  
 repositoryRestConfiguration.exposeIdsFor(Proprietaire.class);  
 repositoryRestConfiguration.exposeIdsFor(Vehicule.class);  
 return args -> {  
 Proprietaire agoujil = new Proprietaire(null, "AGOUJIL", "06-10-2000", "agoujil1@gmail.com",null);  
 Proprietaire ablkrim = new Proprietaire(null, "Ablkrim", "06-10-2000", "abdlkrim2@gmail.com",null);  
 proprietaireRepository.save(agoujil);  
 proprietaireRepository.save(ablkrim);  
 Vehicule vehicule1 = new Vehicule(null,"764TERETRE","Dacia","20PH","2010",agoujil);  
 Vehicule vehicule2 = new Vehicule(null,"764TEREsjdjsTRE","ford","300FH","2020",agoujil);  
 vehiculeRepository.save(vehicule1);  
 vehiculeRepository.save(vehicule2);  
 vehiculeRepository.save(new Vehicule(null,"764TERETRE","toyota","20PH","2010",ablkrim));  
  
 };  
 }  
  
}

* **Teste l’application :**

Voire les résultats dans la base de données H2 console :







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

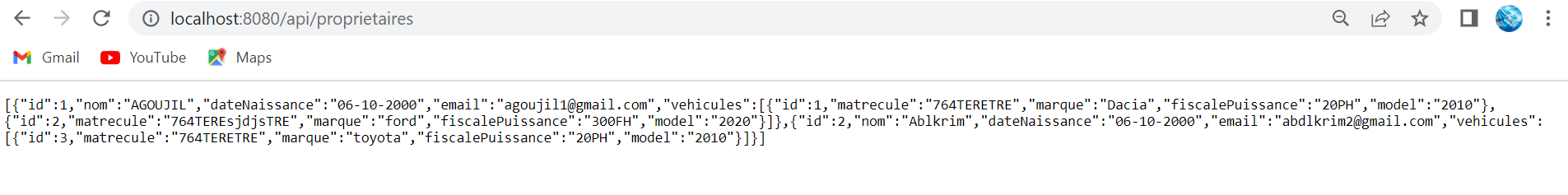
* **Rest controller**
* **ProprietaireRestController**
* package ma.enset.immatriculation\_service.web;  
    
  import ma.enset.immatriculation\_service.entities.Proprietaire;  
  import ma.enset.immatriculation\_service.repositories.ProprietaireRepository;  
  import org.springframework.web.bind.annotation.\*;  
    
  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 proprietaire1=proprietaireRepository.findById(Long.*valueOf*(id)).orElseThrow();  
   if(proprietaire.getNom()!=null) proprietaire1.setNom(proprietaire.getNom());  
   if(proprietaire.getDateNaissance()!=null) proprietaire1.setDateNaissance (proprietaire.getDateNaissance());  
   if(proprietaire.getEmail()!=null) proprietaire1.setEmail (proprietaire.getEmail());  
   if(proprietaire.getVehicules()!=null) proprietaire1.setVehicules(proprietaire.getVehicules());  
   return proprietaireRepository.save(proprietaire1);  
   }  
   @DeleteMapping("/proprietaires/{id}")  
   public void delete(@PathVariable String id){  
   proprietaireRepository.deleteById(Long.*valueOf*(id));  
   }  
  }
* **VéhiculeRestController**
* package ma.enset.immatriculation\_service.web;  
    
  import ma.enset.immatriculation\_service.dto.VehiculeDto;  
  import ma.enset.immatriculation\_service.entities.Proprietaire;  
  import ma.enset.immatriculation\_service.entities.Vehicule;  
  import ma.enset.immatriculation\_service.repositories.ProprietaireRepository;  
  import ma.enset.immatriculation\_service.repositories.VehiculeRepository;  
  import org.springframework.web.bind.annotation.\*;  
    
  import java.util.List;  
    
    
  @RestController  
  @RequestMapping("/api")  
  public class VehiculeRestController {  
   private VehiculeRepository vehiculeRepository;  
   private ProprietaireRepository proprietaireRepository;  
   private VehiculeDto vehiculeDto;  
    
   public VehiculeRestController(VehiculeRepository vehiculeRepository, ProprietaireRepository proprietaireRepository) {  
   this.vehiculeRepository = vehiculeRepository;  
   this.proprietaireRepository = proprietaireRepository;  
   }  
    
   @GetMapping("/vehicules")  
   public List<Vehicule> vehicules(){  
   return vehiculeRepository.findAll();  
    
   }  
   @GetMapping("/vehicules/{id}")  
   public Vehicule vehicule(@PathVariable String id){  
   return vehiculeRepository.findById(Long.*valueOf*(id))  
   .orElseThrow(()->new RuntimeException(String.*format*("Vehicule % not found",id)));  
   }  
   @PostMapping("/vehicules")  
   public Vehicule save(@RequestBody VehiculeDto vehiculeDto){  
   Proprietaire proprietaire=proprietaireRepository.findByNom(vehiculeDto.getNom()).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)).orElseThrow();  
   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

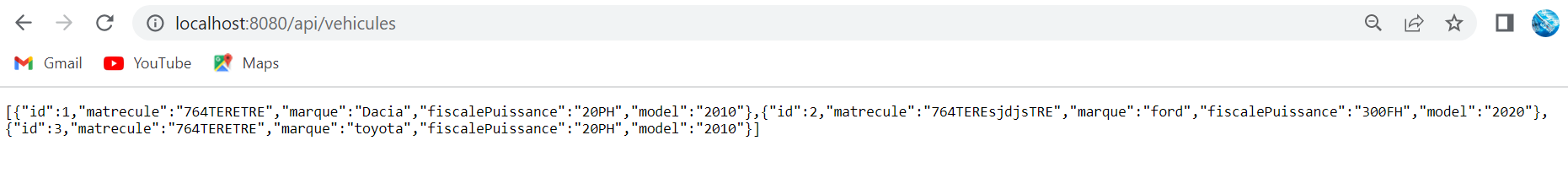
package ma.enset.immatriculation\_service.dto;  
  
import lombok.Data;  
  
@Data  
public class VehiculeDto {  
 String nom;  
 private String matrecule;  
 private String marque;  
 private String fiscalePuissance;  
 private String model;  
  
}

c. Tester les 4 web services

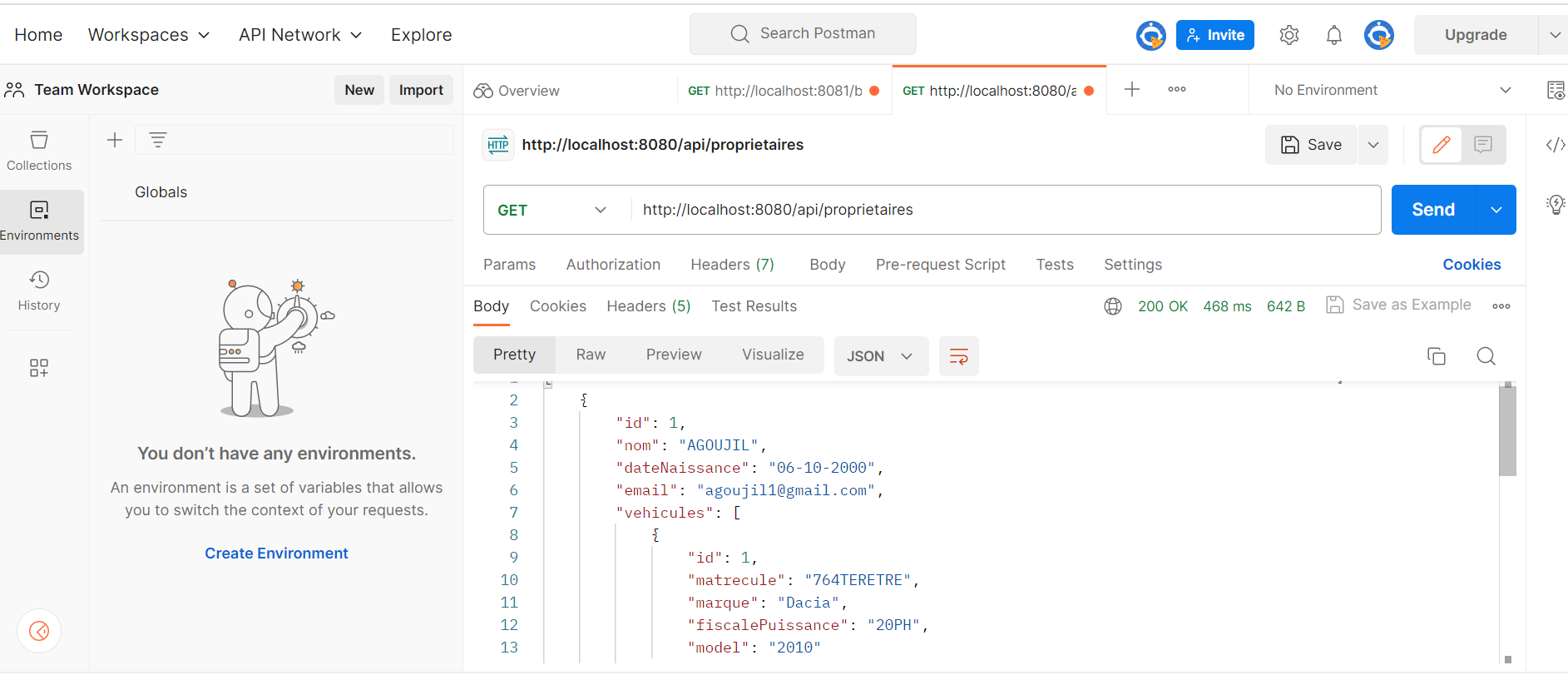
* **Tester Rest**
* Afficher les proprietaires

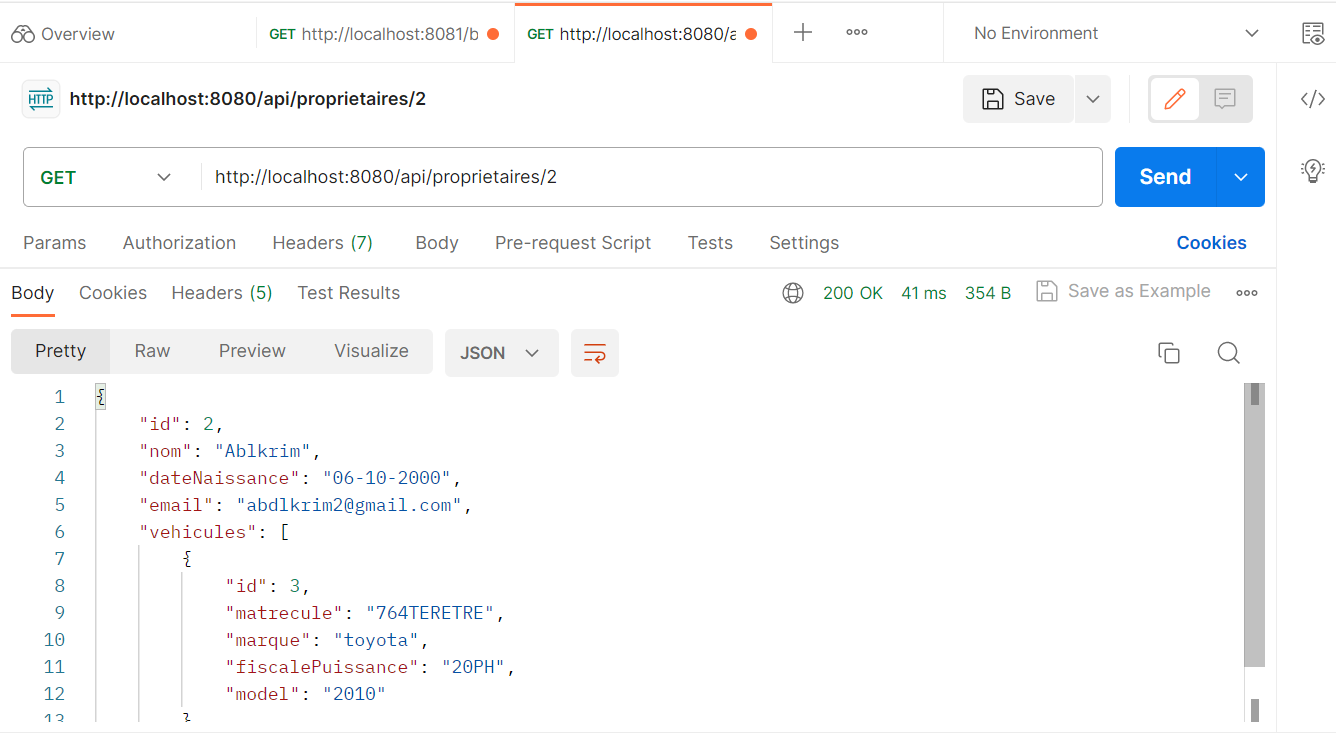


* Afficher les vehicules

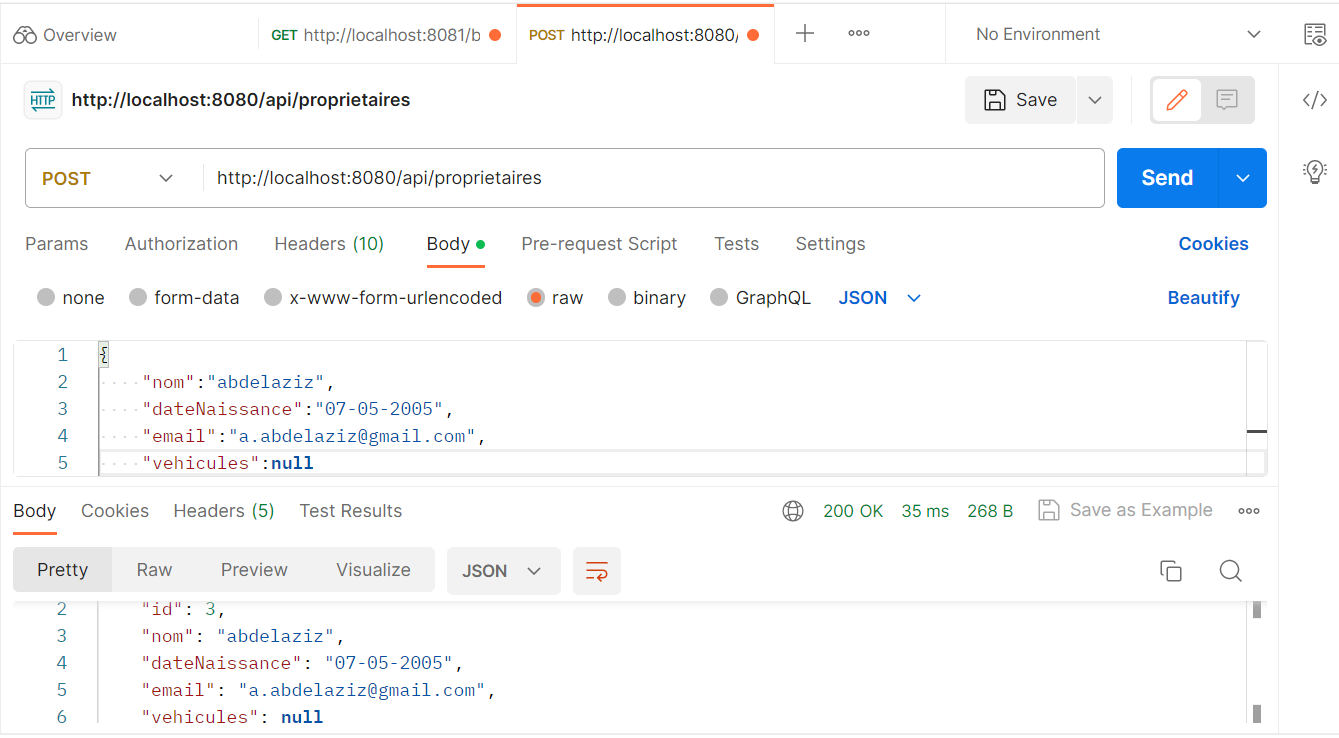


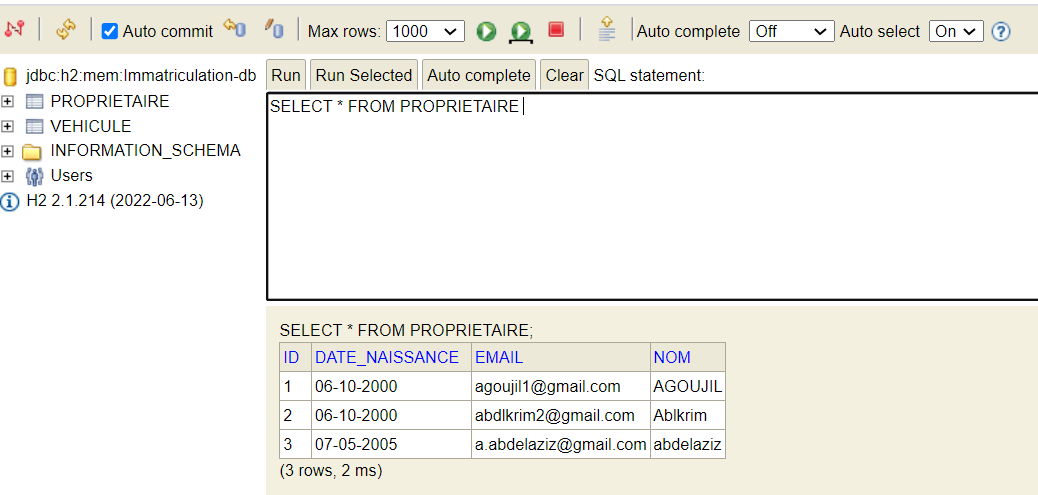
* Tester les méthodes avec postman
* Table Proprietaire :
* Méthode GET



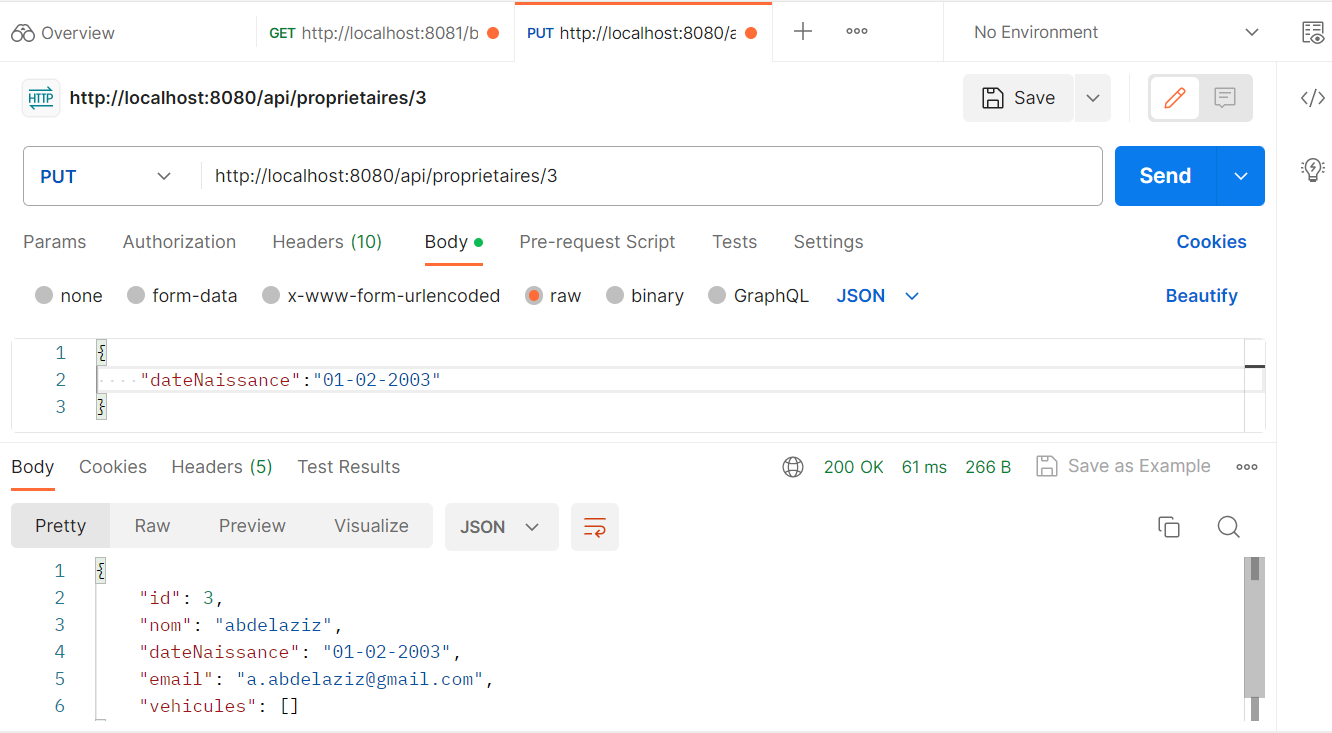


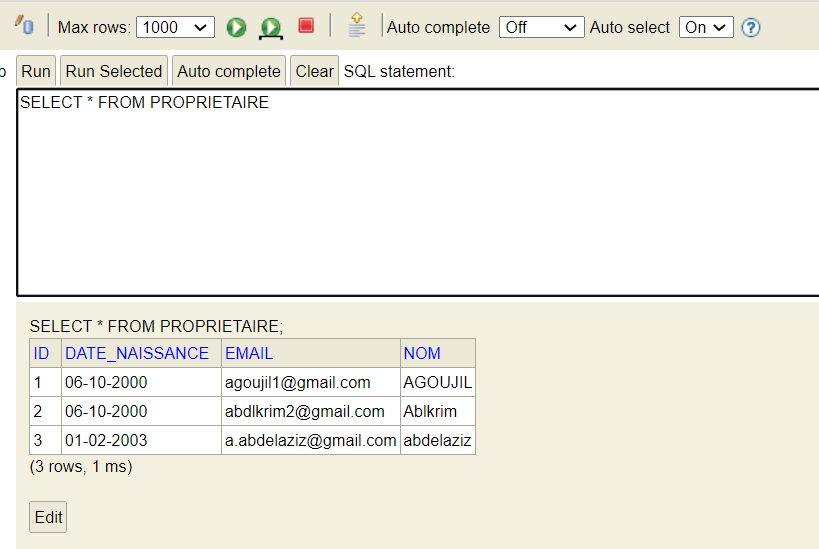
* Méthode POST



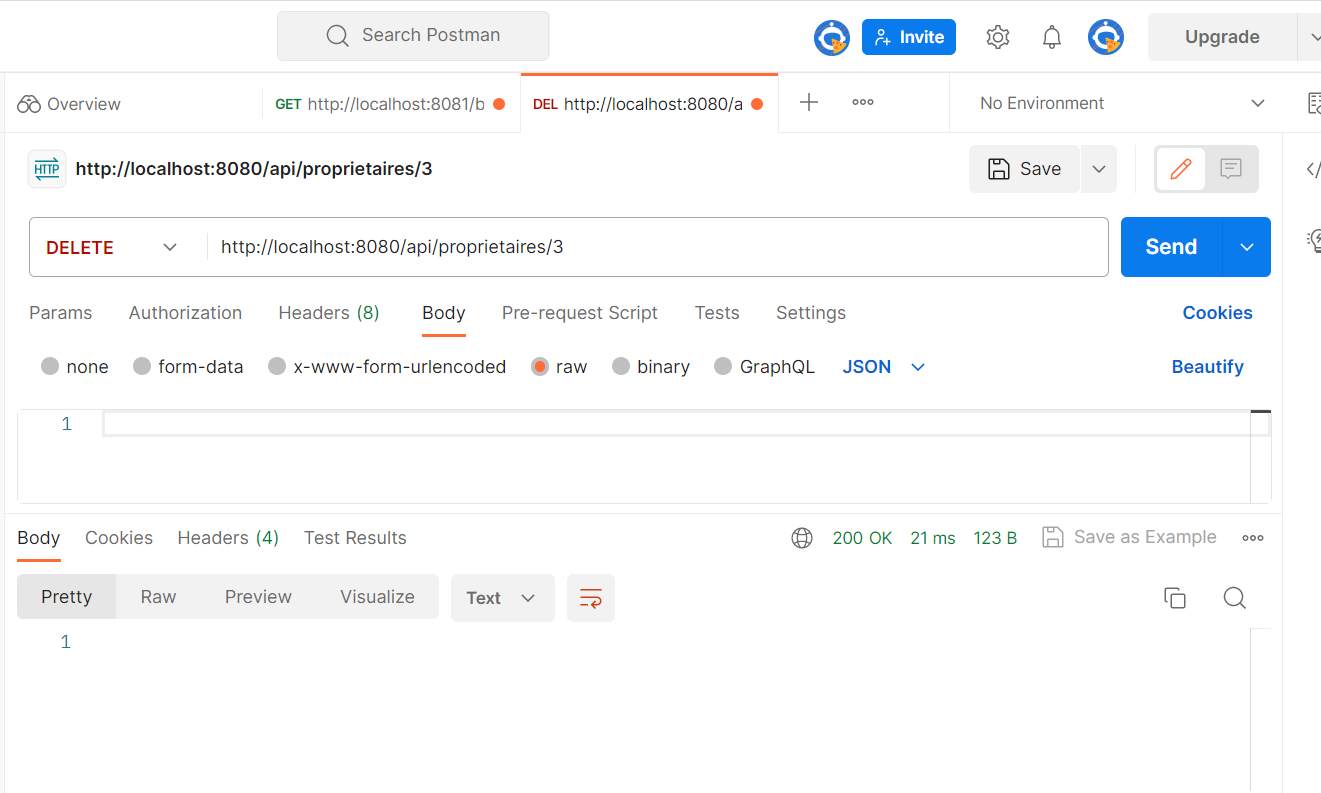


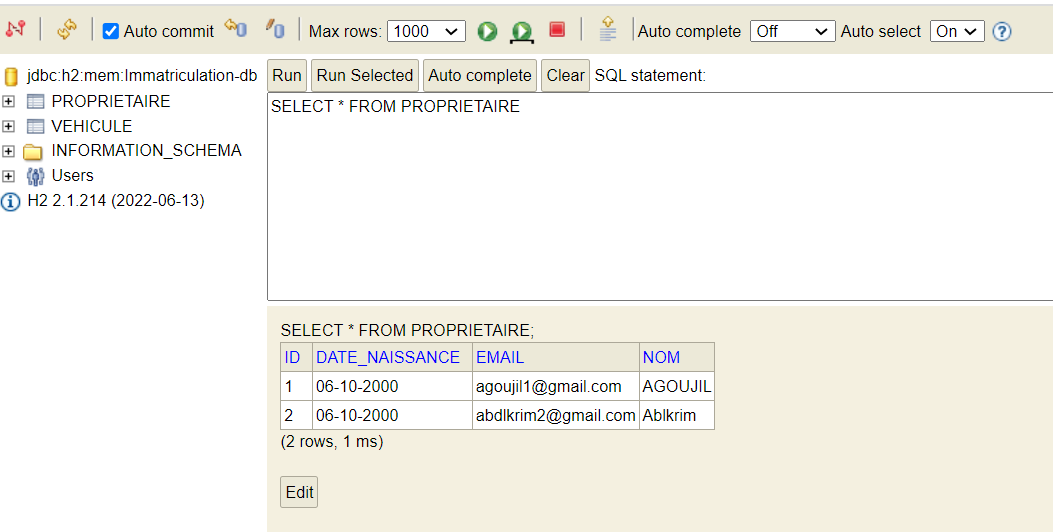
* Méthode PUT



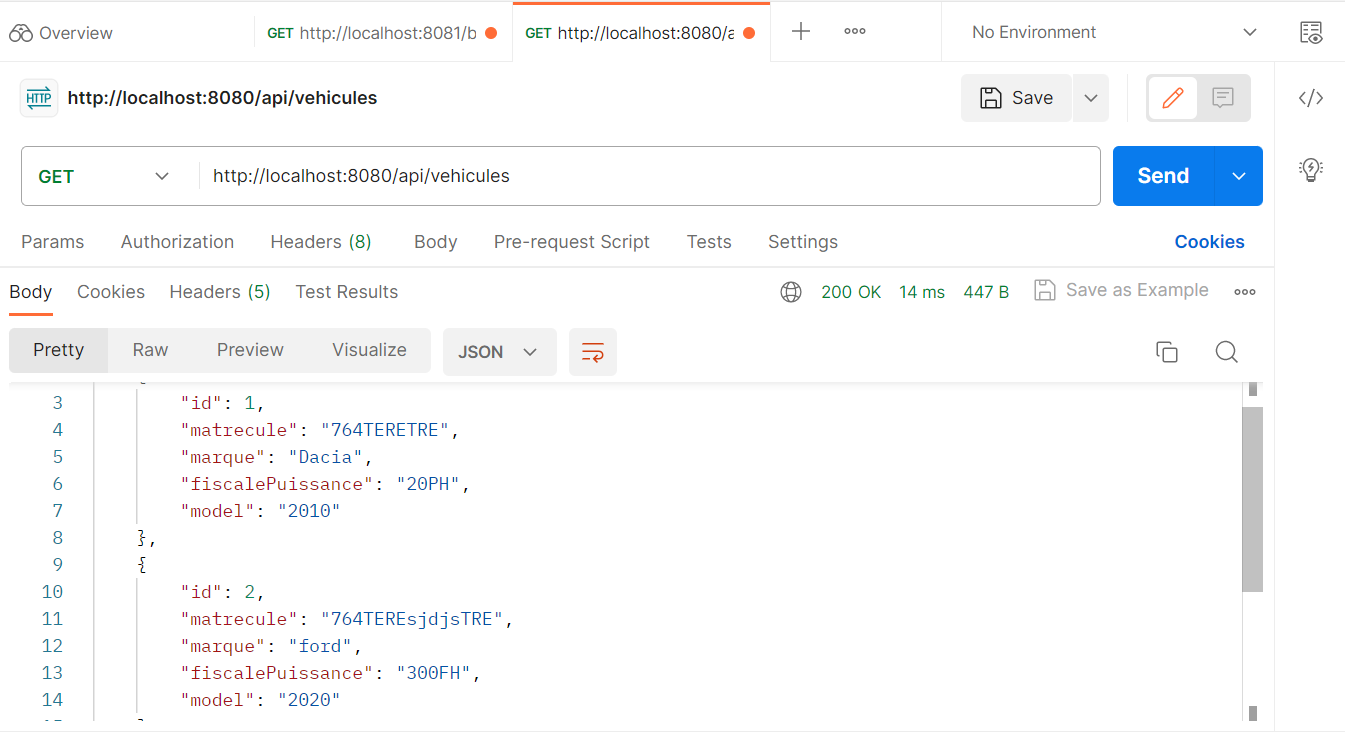


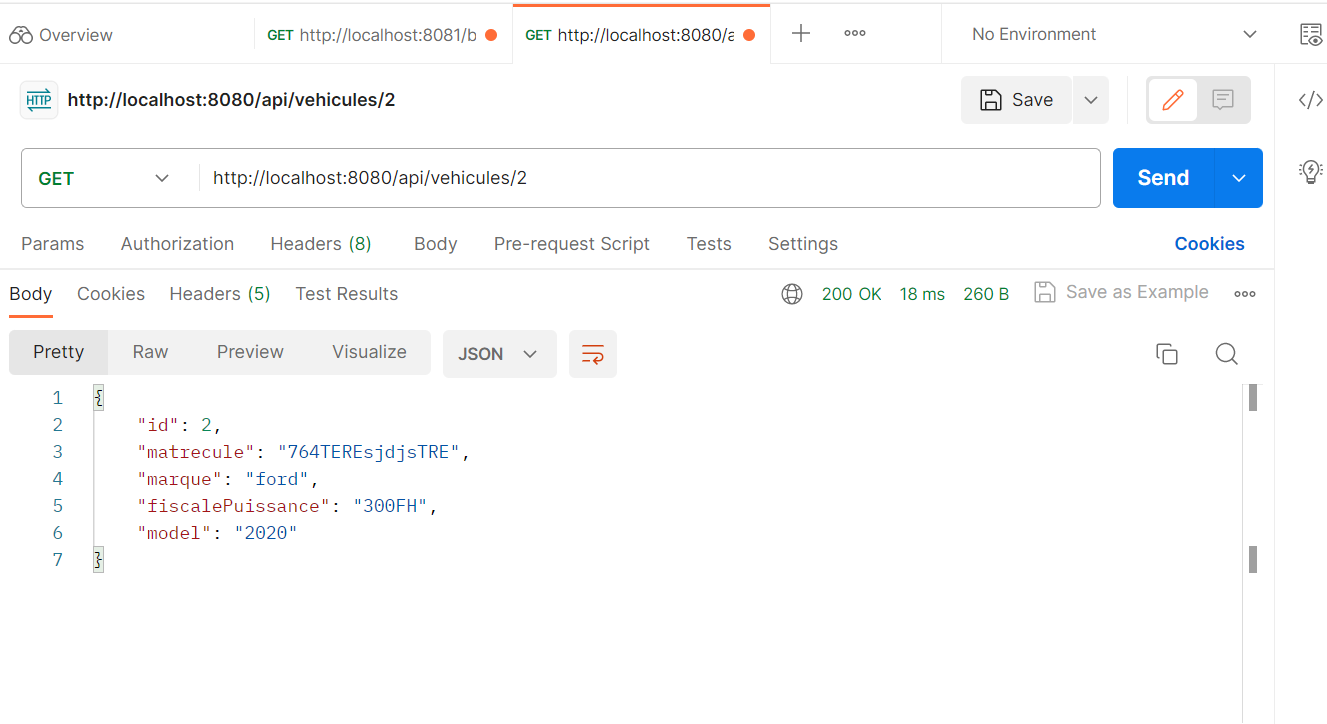
* Méthode DELETE



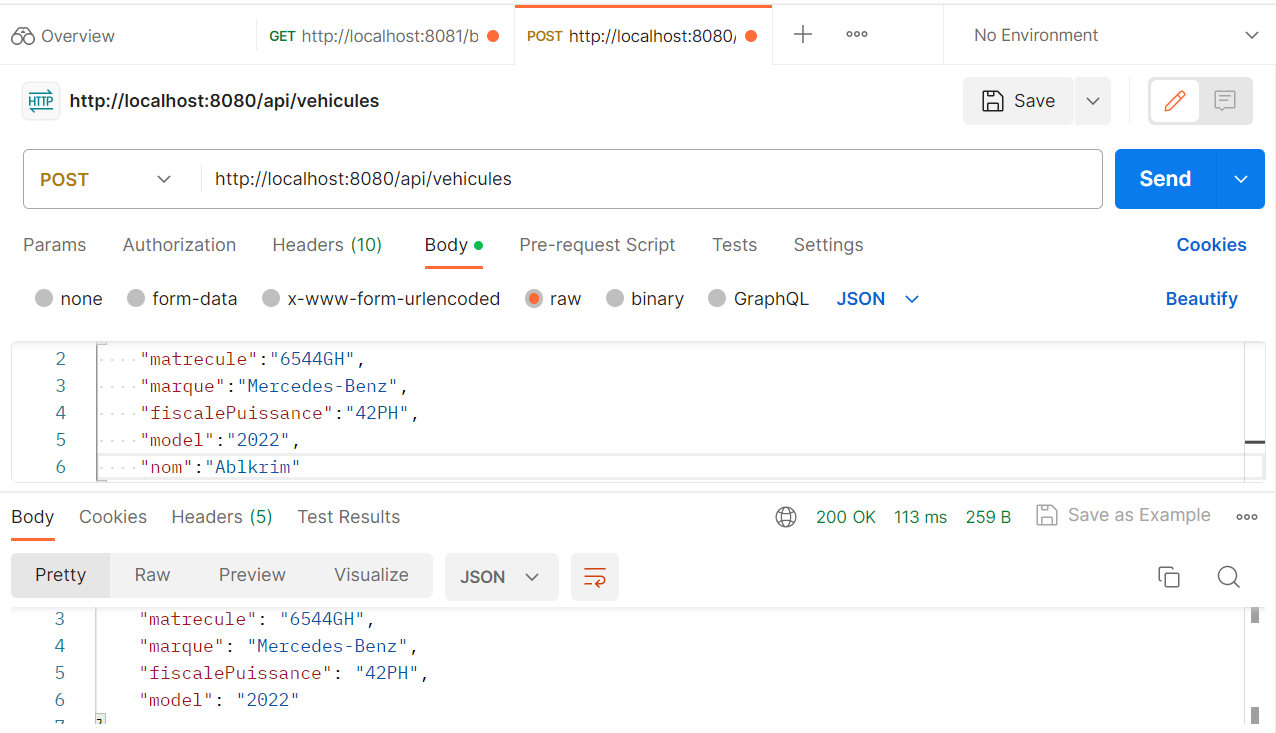


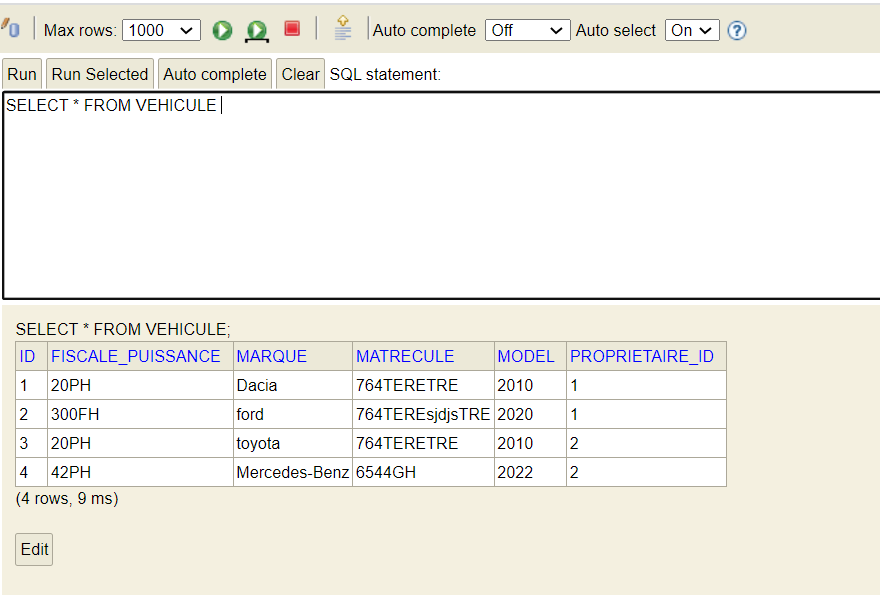
* Table Véhicule :
* Méthode GET





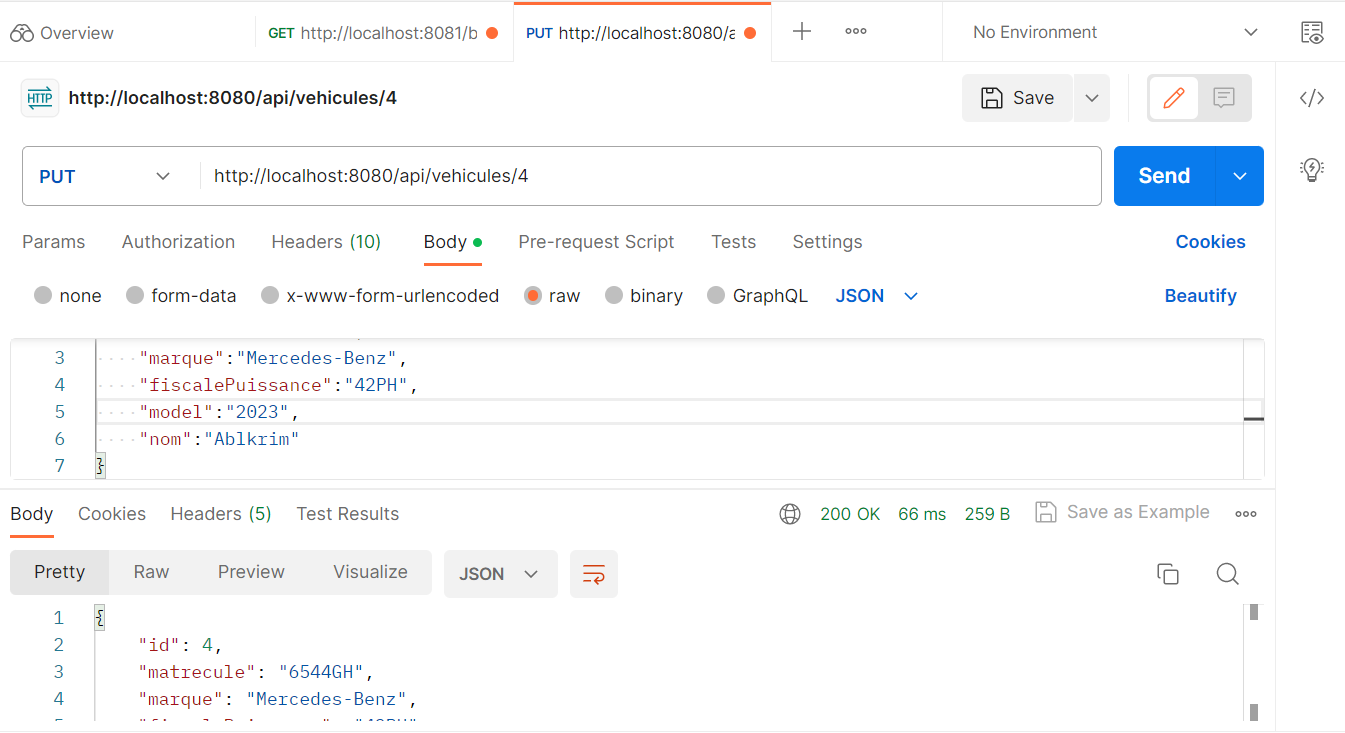
* Méthode POST

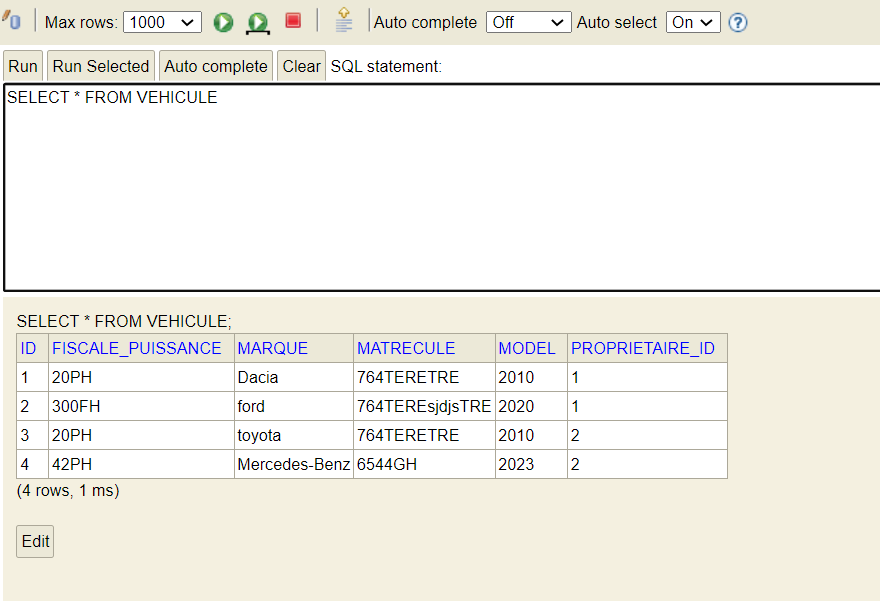




* Méthode POST

Changer le model





* Méthode POST

Supprimer le dernier véhicule ajouter :

