ECOLE NORMALE SUPÉRIEURE DE
L'ENSEIGNEMENT TECHNIQUE DE MOHAMMEDIA

UNIVERSITÉ HASSAN II DE CASABLANCA

ENSET

المدرسة العليا لأساتذة التعليم التقني

المحمدية

جامعة الحسن الثاني بالدار البيضاء

DEPARTEMENT MATHEMATIQUES ET INFORMATIQUE

Rapport d'examen Blanc

Filière :
« Génie du Logiciel et des Systèmes Informatiques Distribués »
GLSID

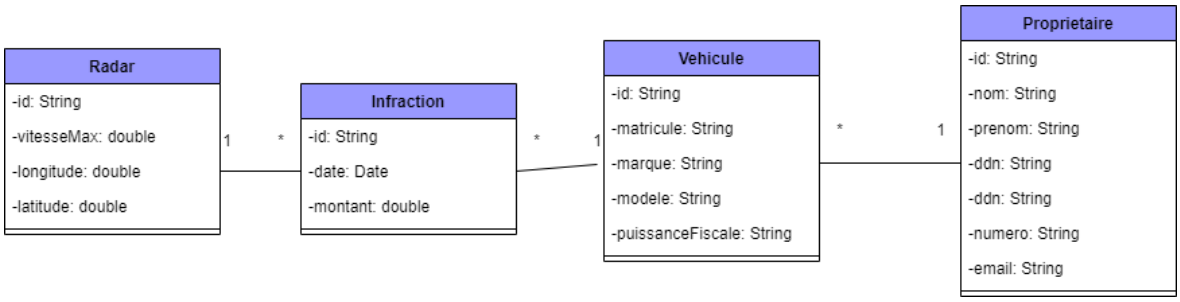
Examen Blanc Systèmes Distribués

Réalisé par :

Ikram BERRADI

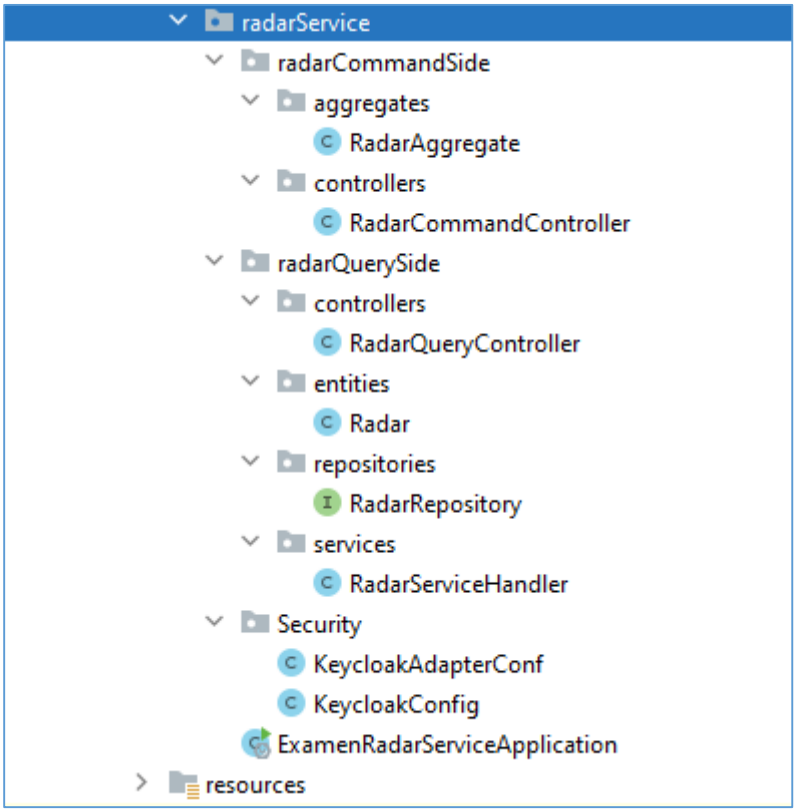
Année Universitaire : 2022-2023

1. Etablir un diagramme de classe global du projet



2. Développer le micro-service Radar

Structure du micro-service Radar



- Command

RadarAggregate

RadarAggregate.java

```java
@Aggregate
public class RadarAggregate {
    @AggregateIdentifier
    private String id;
    private double vitesseMax;
    private double longtitude;
    private double latitude;

    public RadarAggregate() {
    }

    @CommandHandler
    public RadarAggregate(CreateRadarCommand command) {
        if (command.getVitesseMax() < 0) {
            throw new RuntimeException("Vitesse max must be positive");
        }

        AggregateLifecycle.apply(new RadarCreatedEvent(
                command.getId(),
                command.getVitesseMax(),
                command.getLongtitude(),
                command.getLatitude()
        ));
    }
```

RadarCommandController

```java
@RestController
@RequestMapping("/radar/commands")
@AllArgsConstructor
@CrossOrigin("*")
public class RadarCommandController {

    private CommandGateway commandGateway;
    private EventStore eventStore;

    @PostMapping("/createRadar")
    public CompletableFuture<String> createRadar(@RequestBody CreateRadarRequestDTO request) {
        return commandGateway.send(new CreateRadarCommand(UUID.randomUUID().toString(),
                request.getVitesseMax(), request.getLongtitude(), request.getLatitude()));
    }

    @PutMapping("/updateRadar")
    public CompletableFuture<String> updateRadar(@RequestBody UpdateRadarRequestDTO request) {
        return commandGateway.send(new UpdateRadarCommand(request.getId(),
                request.getVitesseMax(), request.getLongtitude(), request.getLatitude()));
    }

    @ExceptionHandler(Exception.class)
    public ResponseEntity<String> handleException(Exception e) {
        ResponseEntity<String> entity = new ResponseEntity<~>(e.getMessage(), HttpStatus.INTERNAL_SERVER_ERROR);
```

- Query

Entity Radar

```java
@Entity
@Data @AllArgsConstructor @NoArgsConstructor
public class Radar {
    @Id
    private String id;
    private double vitesseMax;
    private double longtitude;
    private double latitude;
}
```

Repository RadarRepository

```java
package radarService.radarQuerySide.repositories;

import ...

public interface RadarRepository extends JpaRepository<Radar,String> {

}
```

RadarEventHandler

```java
@Service
@AllArgsConstructor
@Slf4j
public class RadarServiceHandler {
    private RadarRepository radarRepository;

    @EventHandler
    public void on(RadarCreatedEvent event){
        log.info("**************************");
        log.info("RadarCreatedEvent received");
        radarRepository.save(new Radar(event.getId(),event.getVitesseMax(),
                event.getLongtitude(),event.getLatitude()));
    }

    @EventHandler
    public void on(RadarUpdatedEvent event){
        log.info("**************************");
        log.info("RadarUpdatedEvent received");
        Radar radar = radarRepository.findById(event.getId()).get();
        radar.setVitesseMax(event.getVitesseMax());
        radar.setLongtitude(event.getLongtitude());
        radar.setLatitude(event.getLatitude());
        radarRepository.save(radar);
    }
```

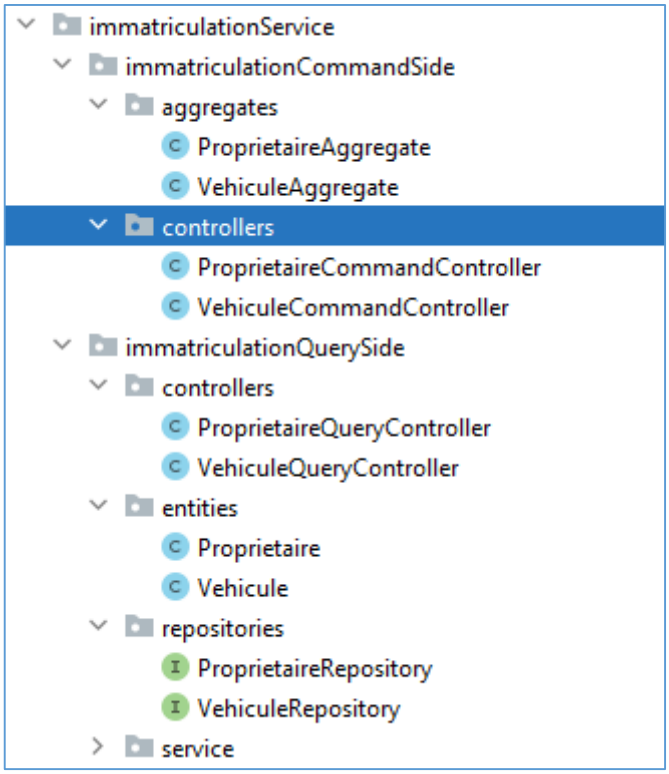
Base de données

+ Options

			id	latitude	longtitude	vitesse_max
Éditer	Copier	Supprimer	AERGYHJIKL	1233	7888	169
Éditer	Copier	Supprimer	UJKOLMPUJN	346	7789	189
Éditer	Copier	Supprimer	YJIKLOMPHG	2929	5656	220

3. Développer le micro-service Immatriculation

Structure du micro- service Immatriculation



- Command

ProprietaireAggregate

```java
@Aggregate
public class ProprietaireAggregate {
    @AggregateIdentifier
    private String id;
    private String nom;
    private String prenom;
    private Date ddn;
    private String email;
    private String numTel;

    public ProprietaireAggregate() {
    }

    @CommandHandler
    public ProprietaireAggregate(CreateProprietaireCommand command) {
        AggregateLifecycle.apply(
                new ProprietaireCreatedEvent(
                        command.getId(),
                        command.getNom(),
                        command.getPrenom(),
                        command.getDdn(),
                        command.getEmail(),
                        command.getNumTel()
                )
```

VehiculeAggregate

```java
@Aggregate
public class VehiculeAggregate {
    @AggregateIdentifier
    private String id;
    private String matricule;
    private String marque;
    private String modele;
    private String puissanceFiscale;
    private String proprietaireId;

    public VehiculeAggregate() {
    }

    @CommandHandler
    public VehiculeAggregate(CreateVehiculeCommand command) {
        AggregateLifecycle.apply(
                new VehiculeCreatedEvent(
                        command.getId(),
                        command.getMatricule(),
                        command.getMarque(),
                        command.getModele(),
                        command.getPuissanceFiscale(),
                        command.getProprietaireId()
                )
```

ProprietaireCommandController

```java
@RestController
@RequestMapping("/proprietaire/commands")
@AllArgsConstructor
@CrossOrigin("*")
public class ProprietaireCommandController {
    private CommandGateway commandGateway;
    private EventStore eventStore;

    @PostMapping("/create")
    public CompletableFuture<String> createProprietaire(@RequestBody CreateProprietaireRequestDT
        return commandGateway.send(new CreateProprietaireCommand(
                UUID.randomUUID().toString(),
                request.getNom(),
                request.getPrenom(),
                request.getDdn(),
                request.getEmail(),
                request.getNumTel()
        ));
    }

    @PutMapping("/update")
    public CompletableFuture<String> updateProprietaire(@RequestBody UpdateProprietaireRequestDT
        return commandGateway.send(new UpdateProprietaireCommand(
```

VehiculeCommandController

```java
@RequestMapping("/vehicule/commands")
@AllArgsConstructor
@CrossOrigin("*")
public class VehiculeCommandController {
    private final CommandGateway commandGateway;
    private final EventStore eventStore;

    @PostMapping("/create")
    public CompletableFuture<String> create(@RequestBody CreateVehiculeRequestDTO request) {
        return commandGateway.send(new CreateVehiculeCommand(
                UUID.randomUUID().toString(),
                request.getMatricule(),
                request.getMarque(),
                request.getModele(),
                request.getPuissanceFiscale(),
                request.getProprietaireId()
        ));
    }

    @PutMapping("/update")
    public CompletableFuture<String> update(@RequestBody UpdateVehiculeRequestDTO request) {
        return commandGateway.send(new UpdateVehiculeCommand(
                request.getId(),
                request.getMatricule(),
```

- Query

Entity Proprietaire

```java
@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Proprietaire {
    @Id
    private String id;
    private String nom;
    private String prenom;
    private Date ddn;
    private String email;
    private String numTel;

    @OneToMany(mappedBy = "proprietaire")
    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private List<Vehicule> vehicules = new ArrayList<>();
}
```

Entity Vehicule

```java
import ...

@Entity
@Data @AllArgsConstructor @NoArgsConstructor
public class Vehicule {
    @Id
    private String id;
    private String marque;
    private String modele;
    private String matricule;
    private String puissanceFiscale;

    @ManyToOne
    private Proprietaire proprietaire;
}
```

Repository ProprietaireRepository

```java
import ...

public interface ProprietaireRepository extends JpaRepository<Proprietaire, String> {
}
```

Repository VehiculeRepository

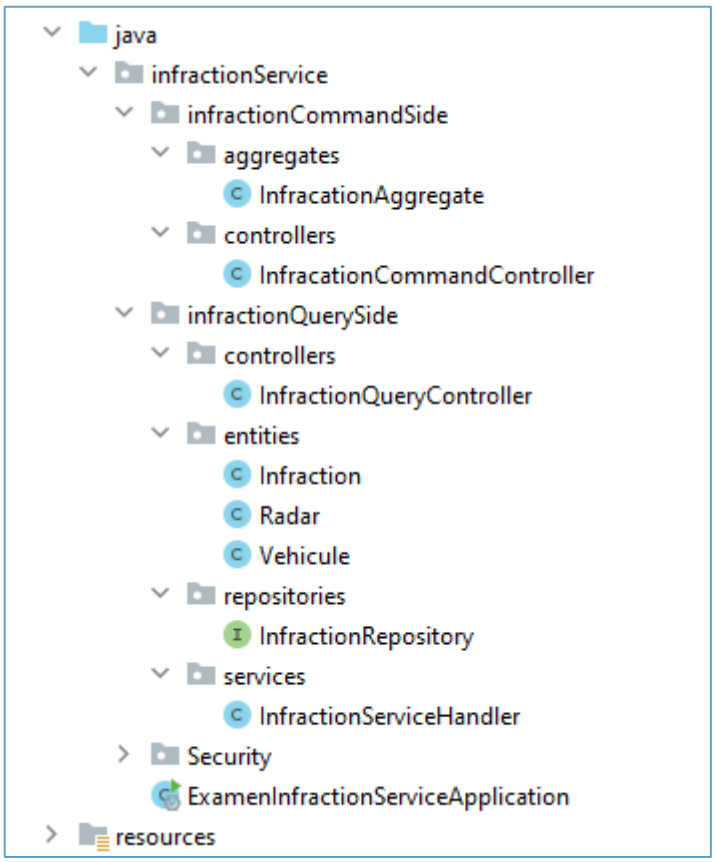
```java
import ...

public interface VehiculeRepository extends JpaRepository<Vehicule, String> {
}
```

ProprietaireQueryController

```java
@RestController
@AllArgsConstructor
@RequestMapping("/proprietaire/queries")
@CrossOrigin("*")
public class ProprietaireQueryController {
    private QueryGateway queryGateway;
    private ProprietaireRepository proprietaireRepository;

    @GetMapping("/allProprietaires")
    public List<Proprietaire> getAllProprietaires() {
        return queryGateway.query(new GetAllProprietairesQuery(), ResponseTypes.multipleInstances(
    }

    @QueryHandler
    public List<Proprietaire> on(GetAllProprietairesQuery query) { return proprietaireRepository.

    @GetMapping("/getProprietaire/{id}")
    public Proprietaire getProprietaire(@PathVariable String id) {
        return queryGateway.query(new GetProprietaireById(id),ResponseTypes.instanceOf(Proprietair
```

5. Développer le micro-service Infractions

Structure du micro-service Infractions



- Command

InfracationAggregate

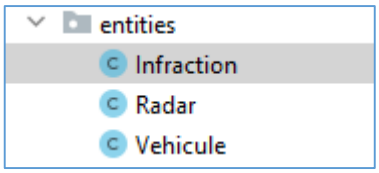
```java
@Aggregate
public class InfracationAggregate {
    @AggregateIdentifier
    private String id;
    private Date date;
    private double montant;
    private String vehiculeId;
    private String radarId;

    public InfracationAggregate() {
    }

    @CommandHandler
    public InfracationAggregate(CreateInfractionCommand command) {
        AggregateLifecycle.apply(new InfractionCreatedEvent(
                command.getId(),
                command.getDate(),
                command.getMontant(),
                command.getVehiculeId(),
                command.getRadarId()
        ));
    }
```

InfracationCommandController

```java
@RestController
@RequestMapping("/infraction/commands")
@AllArgsConstructor
@CrossOrigin("*")
public class InfracationCommandController {
    private final CommandGateway commandGateway;
    private final EventStore eventStore;

    @PostMapping("/create")
    public CompletableFuture<String> createInfraction(@RequestBody CreateInfractionRequestDTO createInfract
        return commandGateway.send(new CreateInfractionCommand(
                UUID.randomUUID().toString(),
                createInfractionRequestDTO.getDate(),
                createInfractionRequestDTO.getMontant(),
                createInfractionRequestDTO.getVehiculeId(),
                createInfractionRequestDTO.getRadarId()
        ));
    }

    @PutMapping("/update")
    public CompletableFuture<String> updateInfraction(@RequestBody UpdateInfractionRequestDTO updateInfract
        return commandGateway.send(new UpdateInfractionCommand(
                updateInfractionRequestDTO.getId(),
```

- Query

Entity Infracation

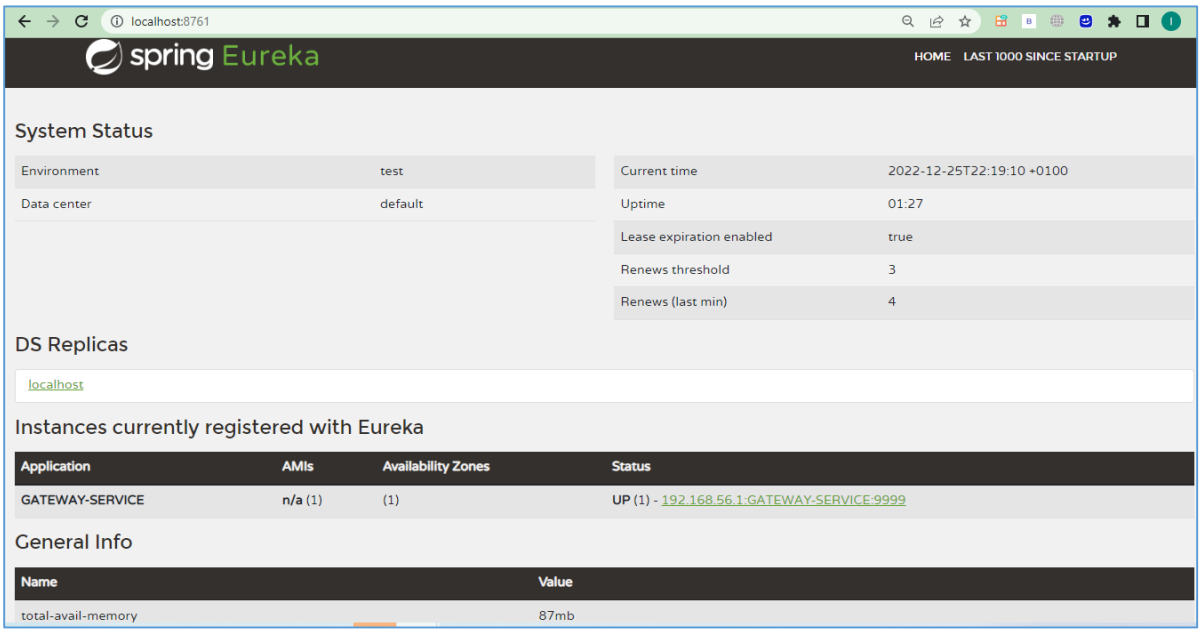



```java
@Entity
@Data @AllArgsConstructor @NoArgsConstructor
public class Infraction {
    @Id
    private String id;
    private Date date;
    private double montant;
    private String vehiculeId;
    private String radarId;

    @Transient
    private Vehicule vehicule;

    @Transient
    private Radar radar;
}
```

InfracationQueryController

```java
@RestController
@RequestMapping("/infraction/queries")
@AllArgsConstructor
@CrossOrigin("*")
public class InfractionQueryController {
    private QueryGateway queryGateway;
    private InfractionRepository infractionRepository;

    @GetMapping("/getAllInfractions")
    public List<Infraction> getAllInfractions(){
        return queryGateway.query(new GetAllInfractionsQuery(),
                ResponseTypes.multipleInstancesOf(Infraction.class)).join();
    }

    @QueryHandler
    public List<Infraction> on(GetAllInfractionsQuery query) { return infractionRepository.findAll()

    @GetMapping("/getInfraction/{id}")
    public Infraction getInfraction(@PathVariable String id){
        return queryGateway.query(new GetInfractionById(id),
                ResponseTypes.instanceOf(Infraction.class)).join();
    }

    @QueryHandler
```

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



- **Discovery service**

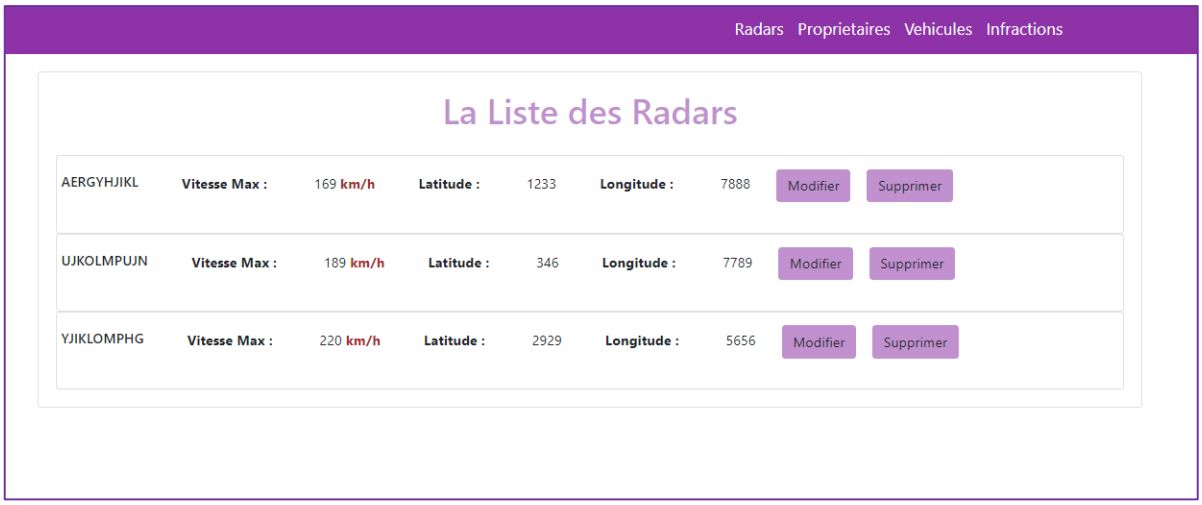
```java
@EnableEurekaServer
public class DiscoveryServiceApplication {

    public static void main(String[] args) { SpringApplication.run(DiscoveryServiceApplication.cla

}
```

- **Gateway-config**

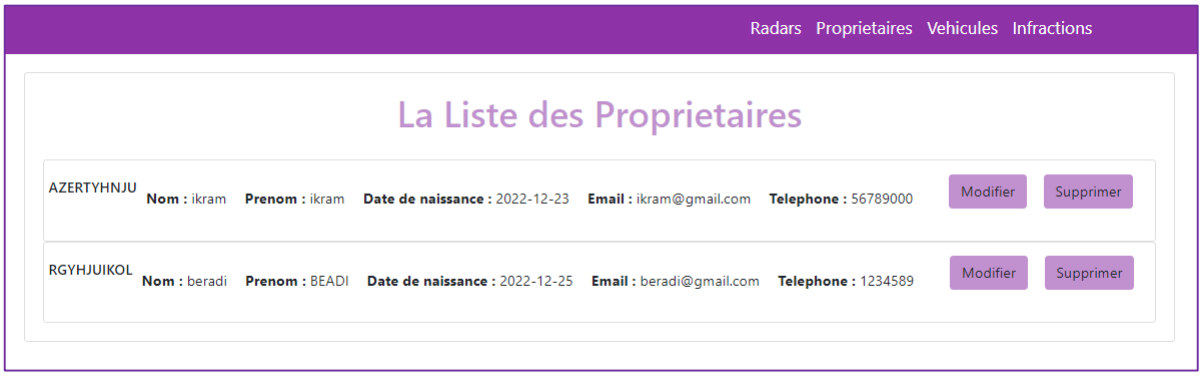
```java
@Configuration
public class GatewayConfig {
    @Bean
    DiscoveryClientRouteDefinitionLocator discoveryClientRouteDefinitionLocator (ReactiveDiscoveryCl
        return new DiscoveryClientRouteDefinitionLocator(rdc,dlp);
    }
}
```

5. Développer votre application Frontend avec Angular ou React

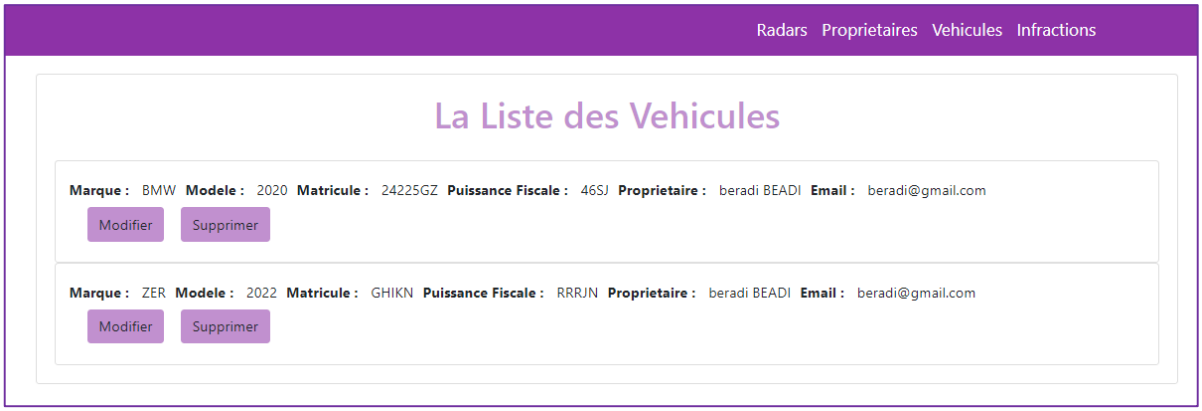
La liste des radars



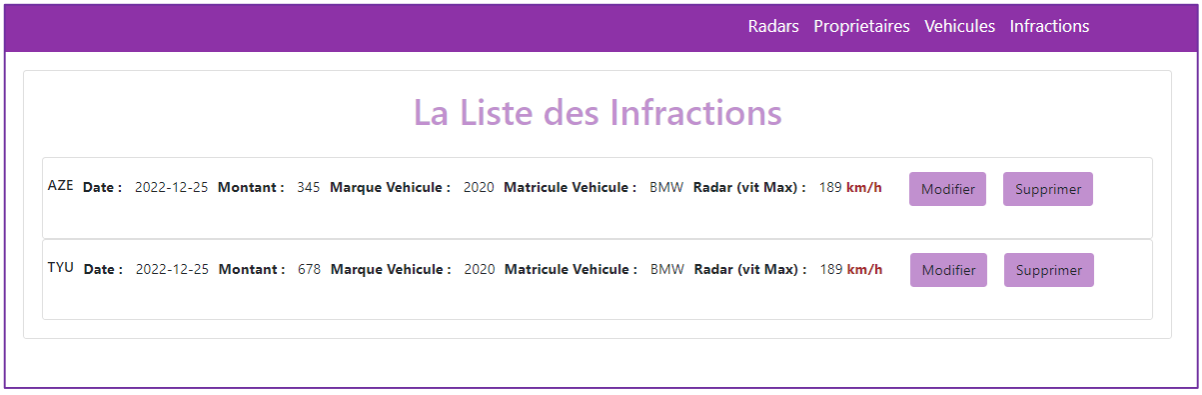
La liste des propriétaires



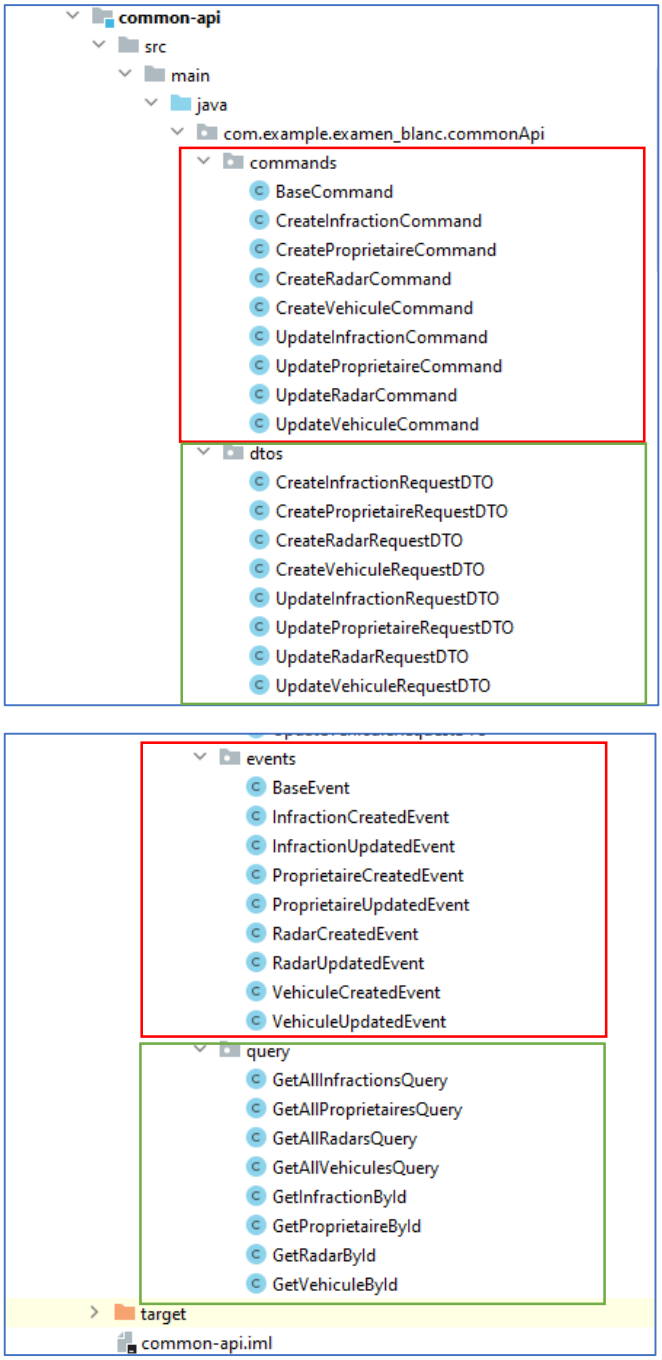
La liste des Vehicles



la liste des infracations



6. Common-Api


common-api
src
main
java
com.example.examen_blanc.commonApi
commands
BaseCommand
CreateInfractionCommand
CreateProprietaireCommand
CreateRadarCommand
CreateVehiculeCommand
UpdateInfractionCommand
UpdateProprietaireCommand
UpdateRadarCommand
UpdateVehiculeCommand
dtos
CreateInfractionRequestDTO
CreateProprietaireRequestDTO
CreateRadarRequestDTO
CreateVehiculeRequestDTO
UpdateInfractionRequestDTO
UpdateProprietaireRequestDTO
UpdateRadarRequestDTO
UpdateVehiculeRequestDTO
events
BaseEvent
InfractionCreatedEvent
InfractionUpdatedEvent
ProprietaireCreatedEvent
ProprietaireUpdatedEvent
RadarCreatedEvent
RadarUpdatedEvent
VehiculeCreatedEvent
VehiculeUpdatedEvent
query
GetAllInfractionsQuery
GetAllProprietairesQuery
GetAllRadarsQuery
GetAllVehiculesQuery
GetInfractionById
GetProprietaireById
GetRadarById
GetVehiculeById
target
common-api.iml