DEPARTEMENT MATHEMATIQUES ET INFORMATIQUE

Rapport d'examen Blanc

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

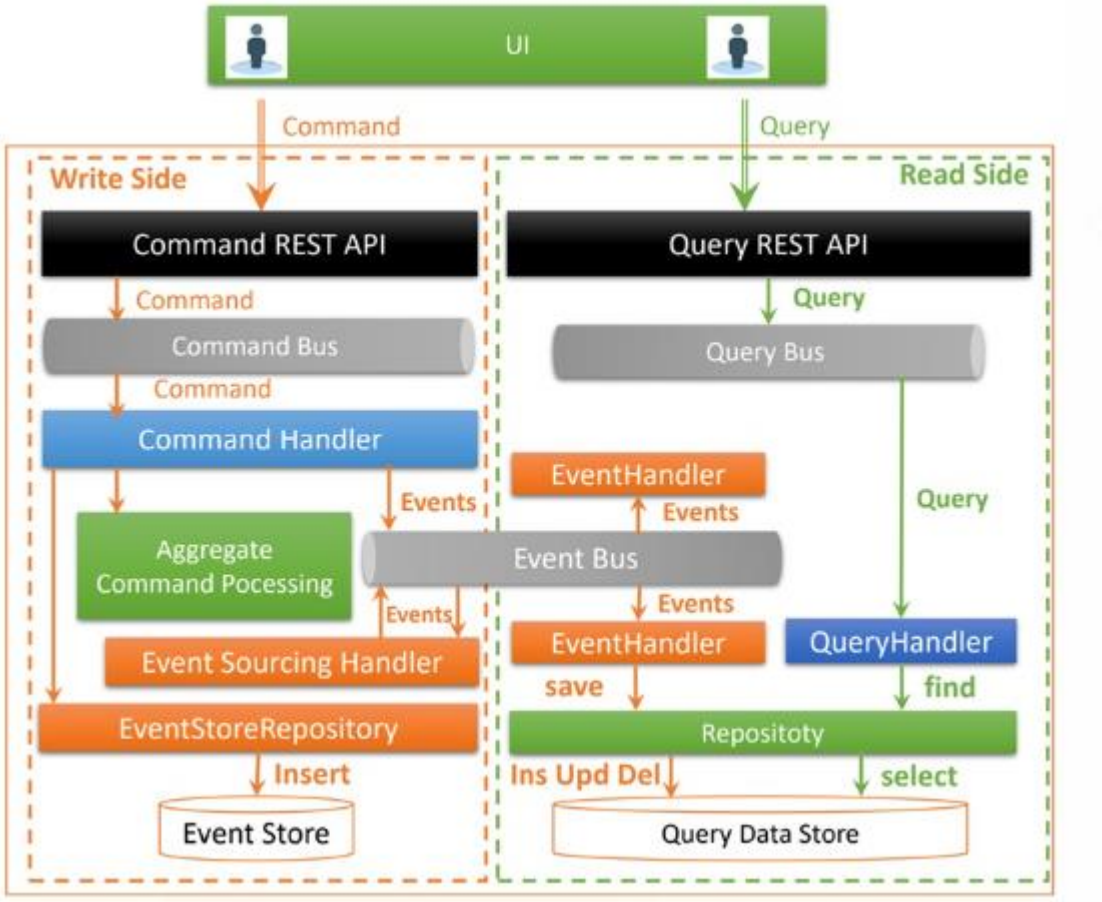
Examen Systèmes Distribués

Réalisé par :

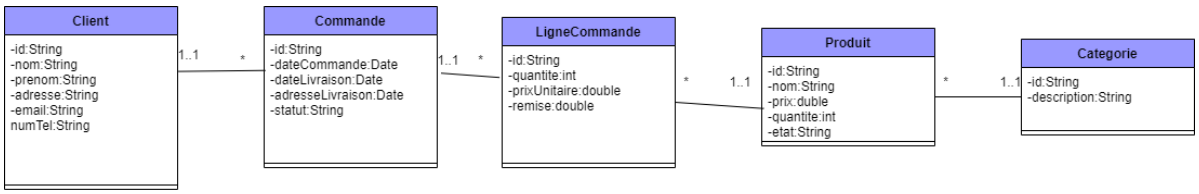
Ikram BERRADI

Année Universitaire : 2022-2023

1. **Établir une architecture technique du projet**

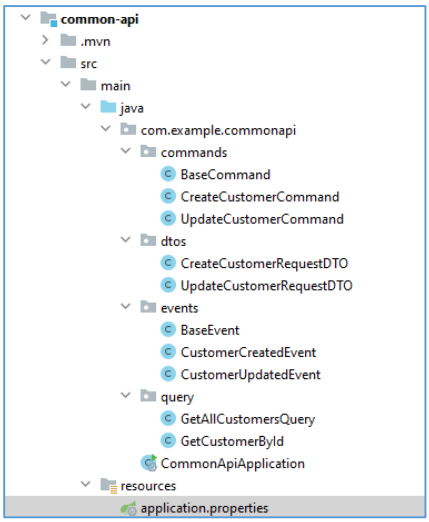


2. Etablir un diagramme de classe global du projet



3. Déployer le serveur AXON Server

- **Structure Commen-api**



BaseCommand

```java
package com.example.commonapi.commands;

import org.axonframework.modelling.command.TargetAggregateIdentifier;

public class BaseCommand <T>{
    @TargetAggregateIdentifier
    private T id;

    public BaseCommand(T id) { this.id = id; }
```

UpdateCustomerCommand

```java
package com.example.commonapi.commands;

public class UpdateCustomerCommand extends BaseCommand<String> {

    private  String nom;
    private String adresse;
    private String email;
    private  String numTel;
    public UpdateCustomerCommand(String id, String nom, String adresse, String email, String numTel) {
        super(id);
        this.nom = nom;
        this.adresse = adresse;
        this.email = email;
        this.numTel = numTel;
    }

    public String getNom() {
        return nom;
    }

    public String getAdresse() {
        return adresse;
    }

    public String getEmail() {
        return email;
    }
```

CreateCustomerCommand

```java
public class CreateCustomerCommand extends BaseCommand<String>{
private  String nom;
private String adresse;
private String email;
private  String numTel;
    public CreateCustomerCommand(String id, String nom, String adresse, String email) {
        super(id);
        this.nom=nom;
        this.adresse=adresse;
        this.email=email;
        this.numTel=numTel;
    }

    public CreateCustomerCommand(String id) { super(id); }

    public String getNom() { return nom; }

    public String getAdresse() { return adresse; }

    public String getEmail() { return email; }

    public String getNumTel() { return numTel; }
}
```

Dtos

```java
@Data
@NoArgsConstructor
@AllArgsConstructor
public class CreateCustomerRequestDTO {

    private String nom ;
    private String adresse ;
    private  String email  ;
     private String telephone ;

}
```

```java
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
public class UpdateCustomerRequestDTO {
    private String id ;
    private String nom ;
    private String adresse ;
    private  String email  ;
    private String telephone ;

}
```

BaseEvent

```java
package com.example.commonapi.events;

public class BaseEvent <T>{
    private  T id ;
    public BaseEvent(T id ) { this.id =id ; }
    public T getId() { return id; }

}
```

CustomerCreatEvent

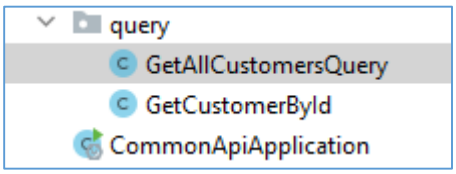
```java
import lombok.Getter;

public class CustomerCreatedEvent extends BaseEvent<String> {
    @Getter
    private String nom ;
    @Getter  private String adresse ;
    @Getter  private  String email  ;
    @Getter private String telephone ;

    public CustomerCreatedEvent(String id, String nom, String adresse, String email, String numTel) {
        super(id);
        this.nom = nom;
        this.adresse = adresse;
        this.email = email;
        this.telephone = telephone;
    }
}
```

CustomerUpdatedEvent

```java
import lombok.Getter;

public class CustomerUpdatedEvent extends BaseEvent<String> {
  @Getter private String nom ;
  @Getter  private String adresse ;
  @Getter  private  String email  ;
  @Getter private String telephone ;

    public CustomerUpdatedEvent(String id, String nom, String adresse, String email, String numTel) {
        super(id);
        this.nom = nom;
        this.adresse = adresse;
        this.email = email;
        this.telephone = telephone;
    }
}
```

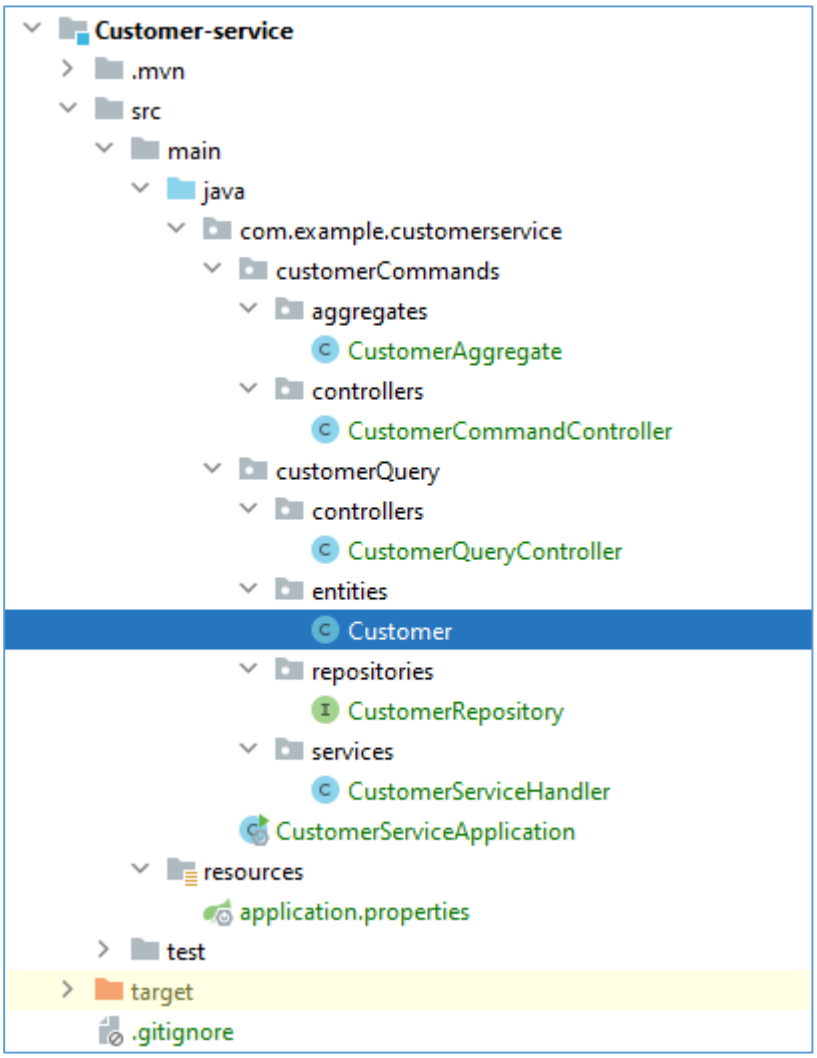
Query

```java
import ...

@Data
@AllArgsConstructor
@NoArgsConstructor
public class GetCustomerById {
    private String id ;
}
```



4. Développer le micro-service Customer-service

Structure du Customer-service



- Command

Aggregate

```java
@Aggregate
public class CustomerAggregate {
    @AggregateIdentifier
    private String id;
    private String nom ;
    private String adresse ;
    private String mail;
    private String tele;

    public CustomerAggregate() {
    }

    @CommandHandler
    public CustomerAggregate(CreateCustomerCommand command) {

        AggregateLifecycle.apply(new CustomerCreatedEvent(
                command.getId(),
                command.getNom(),
                command.getAdresse(),
                command.getMail(),
                command.getTele()
        ));
    }

    @EventSourcingHandler
    public void on(CustomerCreatedEvent event) {
        this.id = event.getId();
```

CommandController

```java
@RestController
@RequestMapping("/client/commands")

@CrossOrigin("*")
public class CustomerCommandController {

    private CommandGateway commandGateway;

    private EventStore eventStore;

    public CustomerCommandController(CommandGateway commandGateway, EventStore eventStore) {
        this.commandGateway = commandGateway;
        this.eventStore = eventStore;
    }

    @PostMapping("/createClient")
    public CompletableFuture<String> createClient(@RequestBody CreateClientRequestDTO request) {
        return commandGateway.send(new CreateCustomerCommand(UUID.randomUUID().toString(),
                request.getNom(),request.getAdresse(),request.getMail(),request.getTele()));
    }

    @PutMapping("/updateClient")
    public CompletableFuture<String> updateClient(@RequestBody UpdateClientRequestDTO request) {
        return commandGateway.send(new UpdateCustomerCommand(request.getId(),
                request.getNom(),request.getAdresse(),request.getMail(),request.getTele()));
    }
```

- Query

CustomerQueryController

```java
@RestController
@RequestMapping("/client/queries")
@AllArgsConstructor
@CrossOrigin("*")
public class CustomerQueryController {

    private QueryGateway queryGateway;
    private CustomerRepository customerRepository;

    @GetMapping("/getAllClients")
    public List<Customer> getAllClients(){
        return queryGateway.query(new GetAllClientsQuery(),
                ResponseTypes.multipleInstancesOf(Customer.class)).join();
    }

    @QueryHandler
    public List<Customer> on(GetAllClientsQuery query) { return customerRepository.findAll(); }

    @GetMapping("/getClient/{id}")
    public Customer getClient(@PathVariable String id){
        return queryGateway.query(new GetClientById(id),
                ResponseTypes.instanceOf(Customer.class)).join();
    }

    @QueryHandler
    public Customer on(GetClientById query){
        return customerRepository.findById(query
```

Entity Customer

```java
@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Customer {
    @Id
    private String id;
    private String nom ;
    private String adresse ;
    private String mail;
    private String tele;

}
```

Repository CustomerRepository

```java
import ...

public interface CustomerRepository extends JpaRepository<Customer,String> {

}
```

RadarEventHandler

```java
@Service
@AllArgsConstructor
@Slf4j
public class CustomerServiceHandler {
    private CustomerRepository  customerRepository;
    @EventHandler
    public void on(CustomerCreatedEvent event){
        log.info("**************************");
        log.info("CustomerCreatedEvent received");
        customerRepository.save(new Customer(event.getId(), event.getNom(), event.getAdresse(), even
    }

    @EventHandler
    public void on(CustomerUpdatedEvent event){
        log.info("**************************");
        log.info("CustomerUpdatedEvent received");
        Customer c = customerRepository.findById(event.getId()).get();
        c.setNom(event.getNom());
        c.setAdresse(event.getAdresse());
        c.setMail(event.getMail());
        c.setTele(event.getTele());
        customerRepository.save(c);
    }
}
```

Base de données

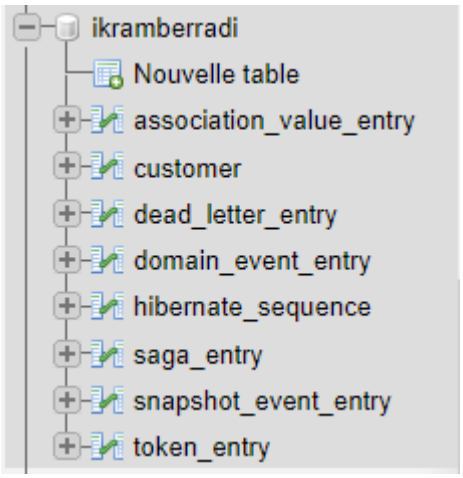


	Table	Action	Lignes	Type	Interclassement	Taille	P
☐	association_value_entry	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_general_ci	48,0 kio	
☐	customer	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_general_ci	16,0 kio	
☐	dead_letter_entry	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_general_ci	64,0 kio	
☐	domain_event_entry	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_general_ci	48,0 kio	
☐	hibernate_sequence	Parcourir Structure Rechercher Insérer Vider Supprimer	1	InnoDB	utf8mb4_general_ci	16,0 kio	
☐	saga_entry	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_general_ci	16,0 kio	
☐	snapshot_event_entry	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_general_ci	32,0 kio	
☐	token_entry	Parcourir Structure Rechercher Insérer Vider Supprimer	1	InnoDB	utf8mb4_general_ci	16,0 kio