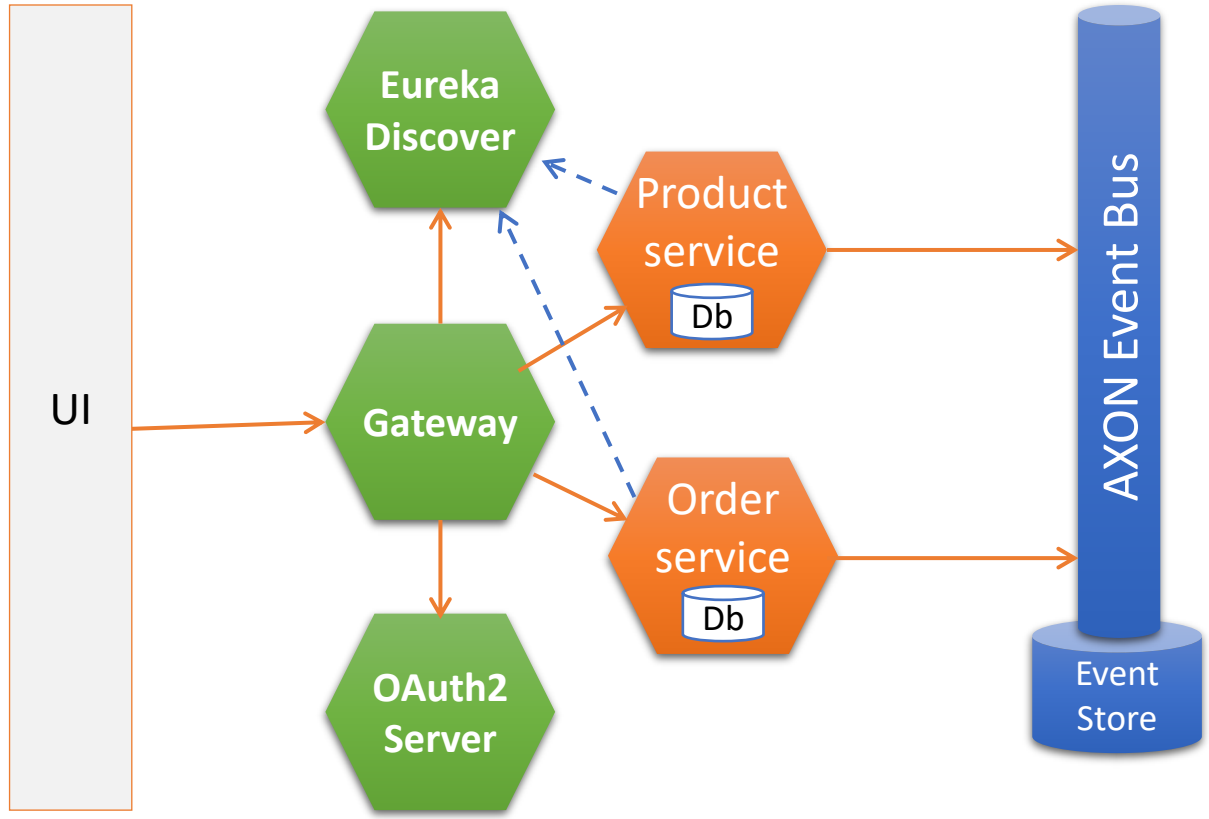
Event Driven Micro-services Architectures, CQRS & Event Sourcing Patterns : Spring Cloud & AXON Framework

Mohamed Youssfi
Enseignant Chercheur, ENSET Mohammedia
Université Hassan II de Casablanca
Directeur du Laboratoire Informatique, Intelligence Artificielle et Cyber Sécurité
Consultant R&D Ingénierie Logicielle



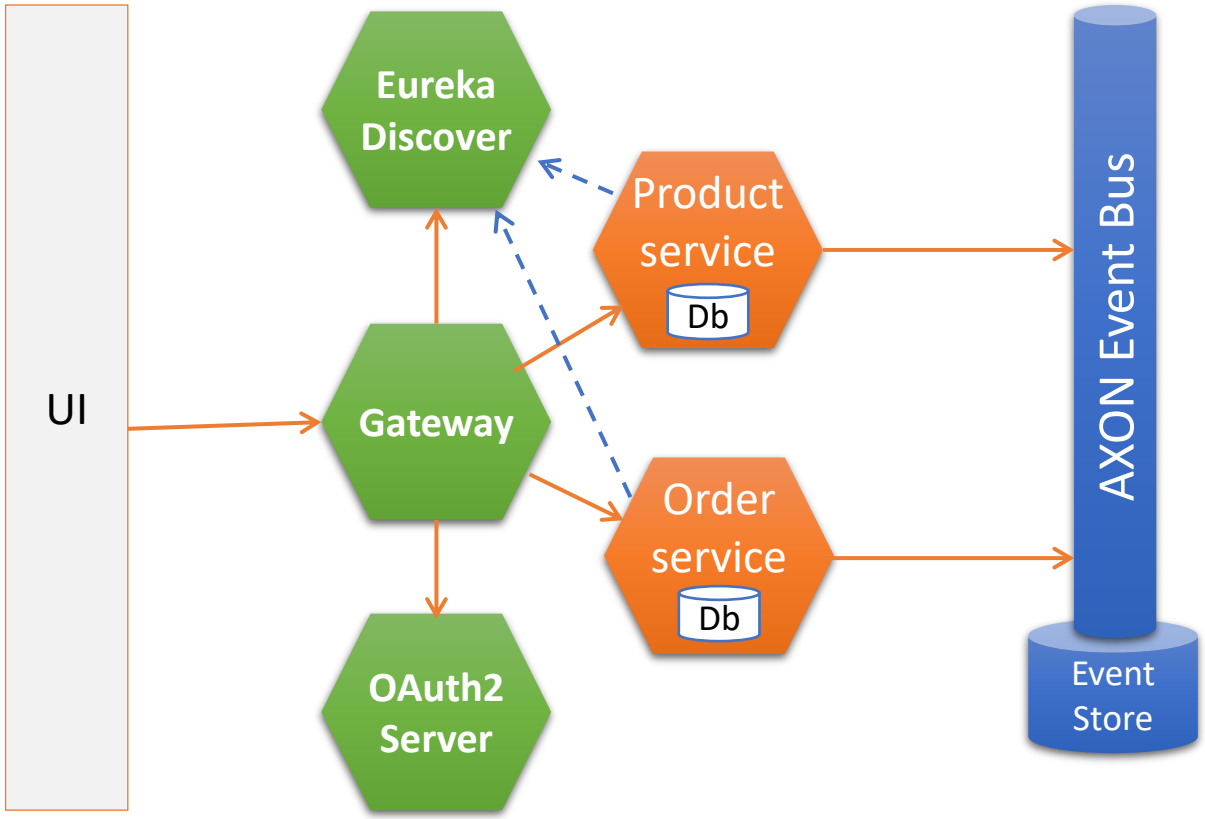
Event Driven Micro-services Architectures, CQRS & Event Sourcing Patterns : Spring Cloud & AXON Framework

spring boot

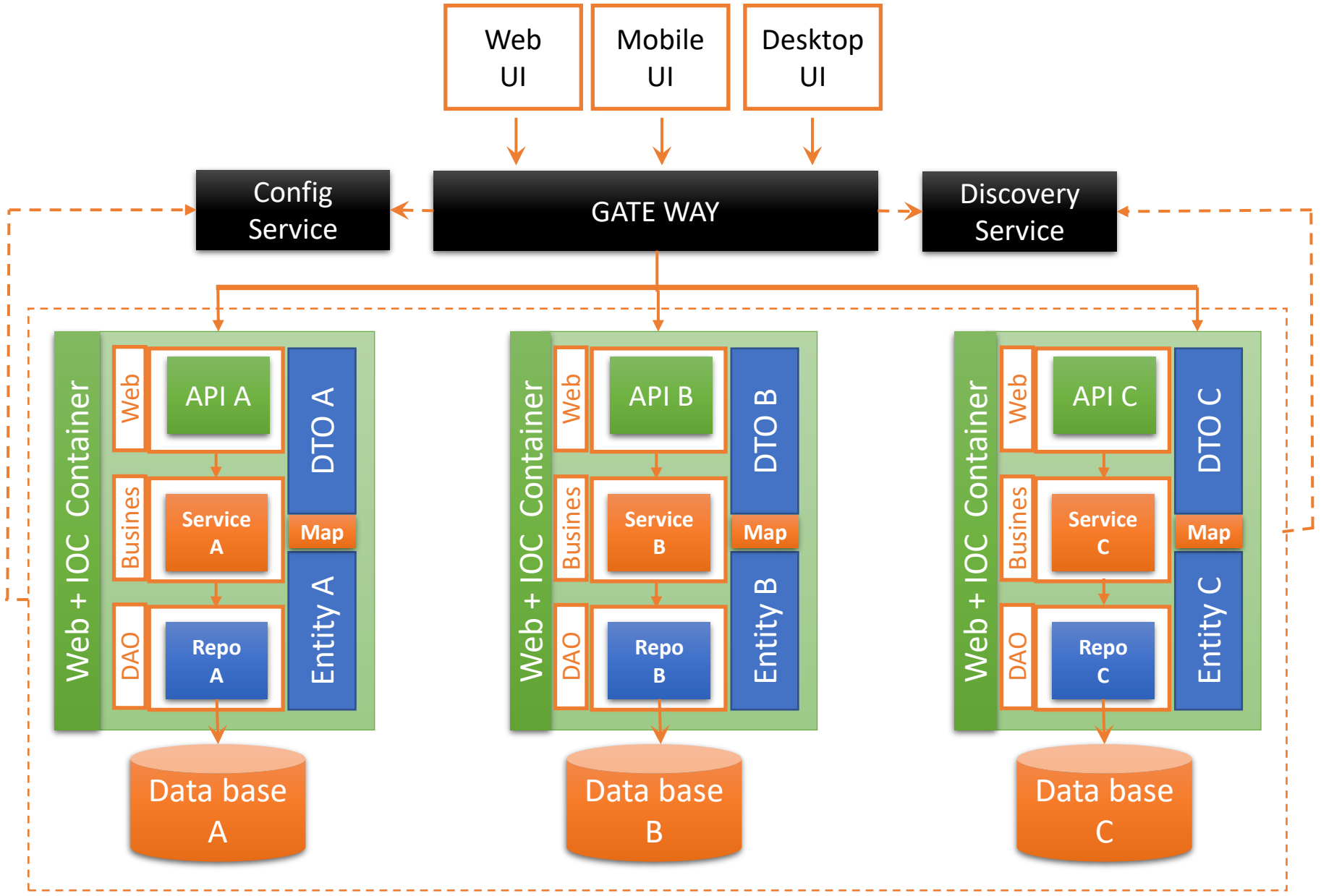
AxonIQ

Spring Cloud

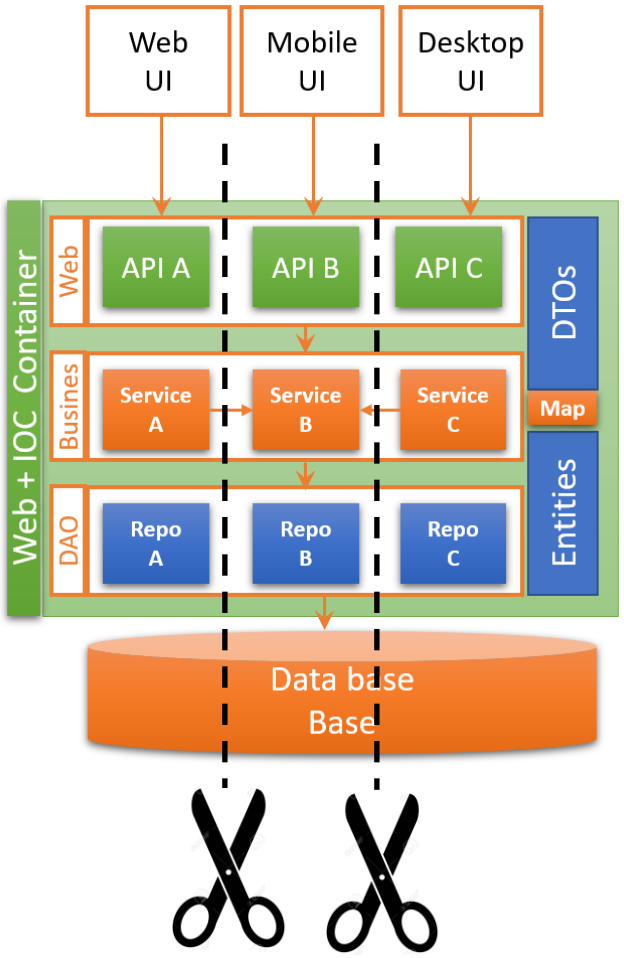
Mohamed Youssfi
Enseignant Chercheur, ENSET Mohammedia
Université Hassan II de Casablanca
Directeur du Laboratoire Informatique, Intelligence Artificielle et Cyber Sécurité
Consultant R&D Ingénierie Logicielle



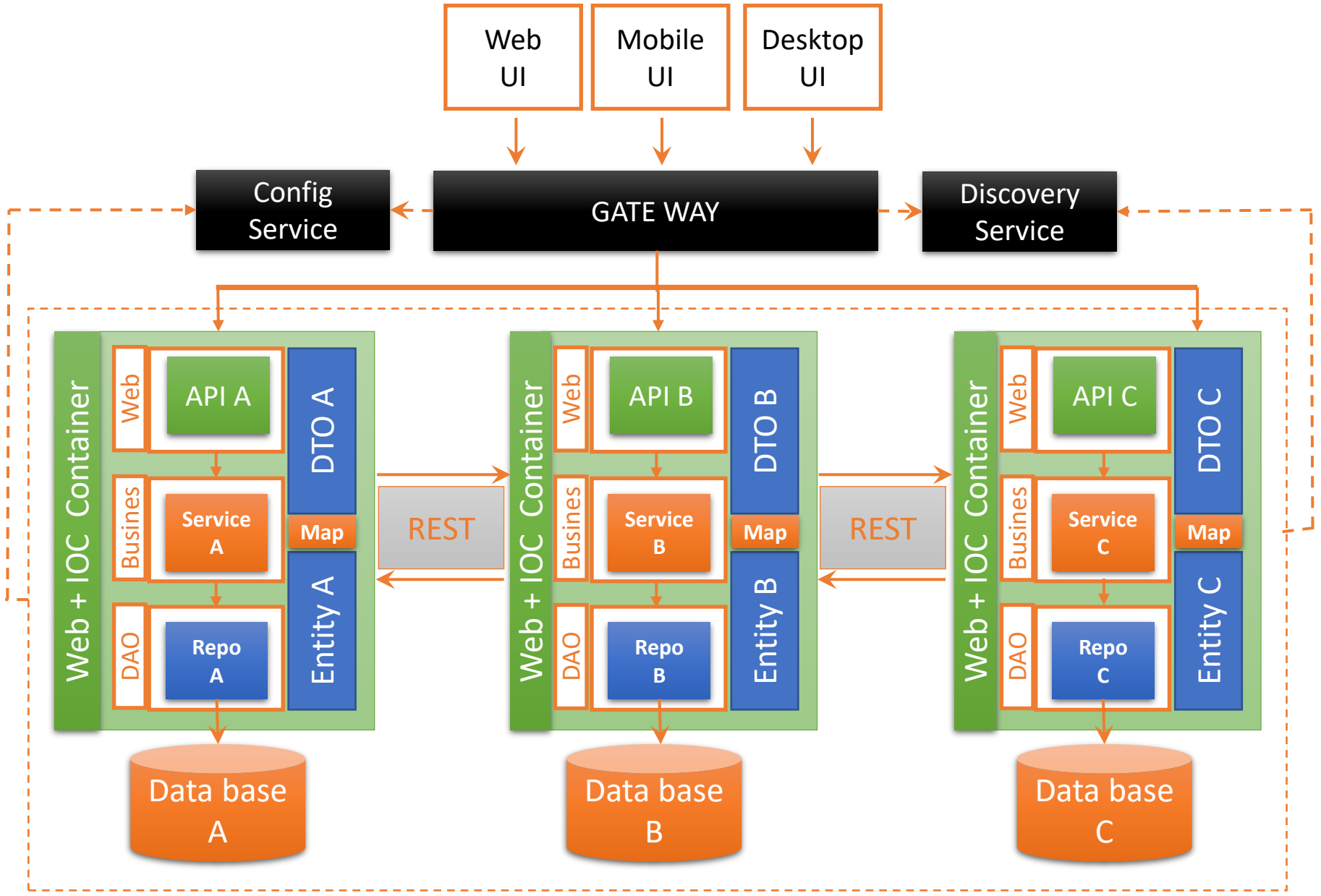
Architecture Micro-Service



Monolith

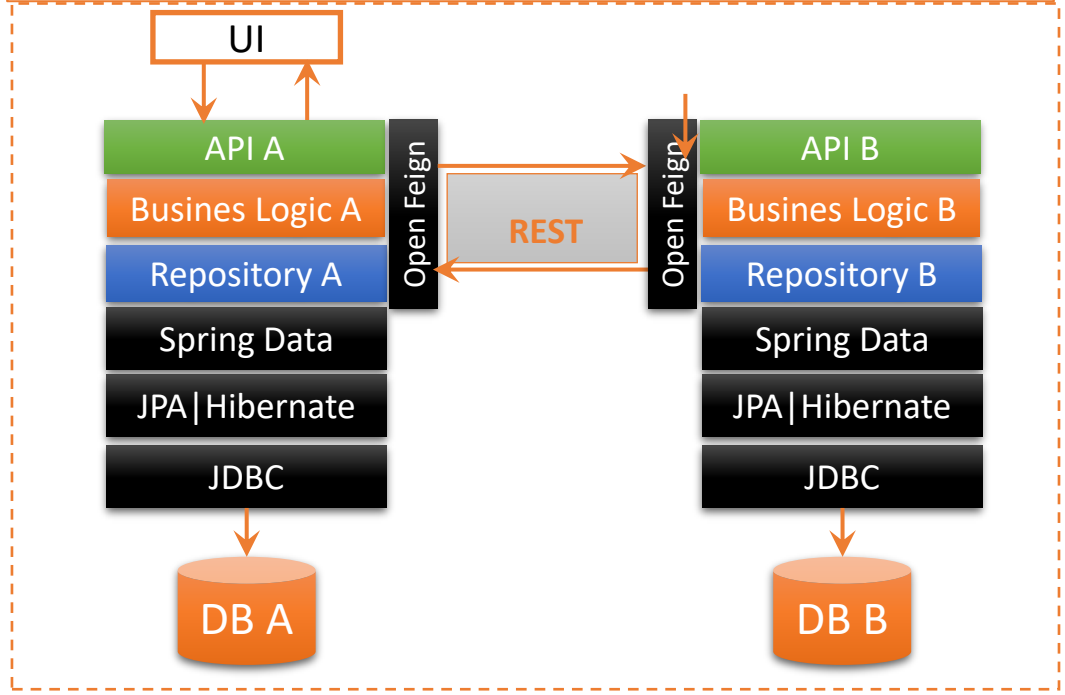


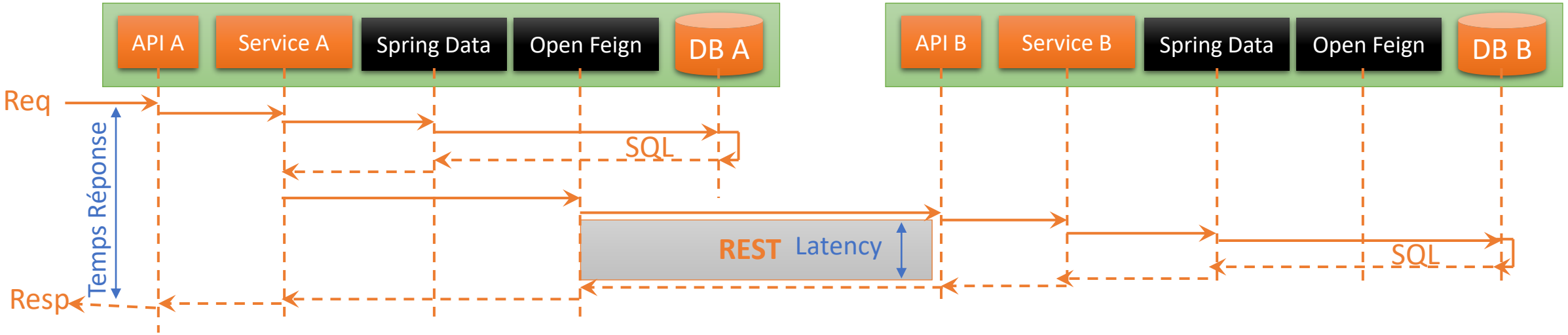
Synchronous Communication Model : REST



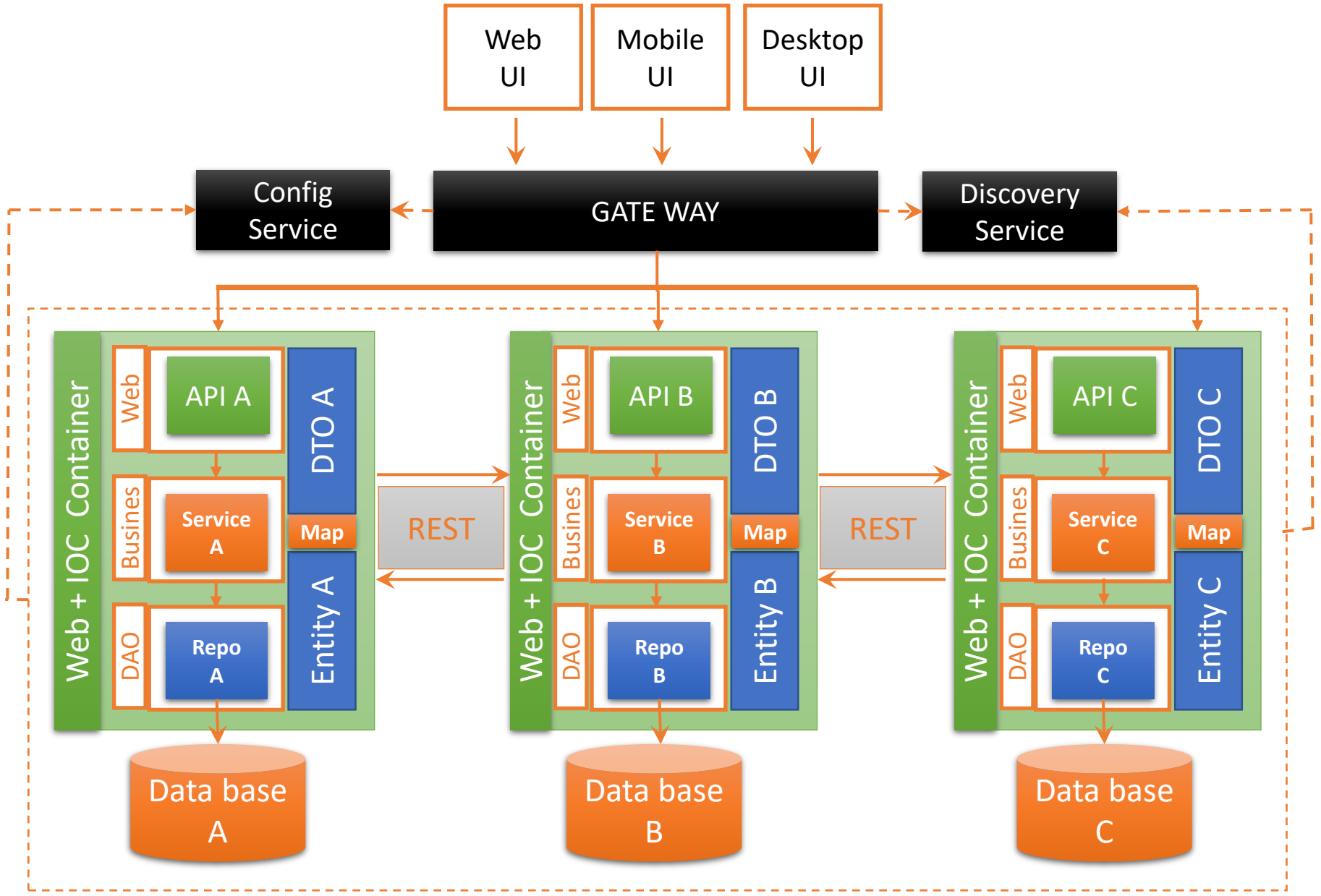
Synchronous Communication Model : REST

Modèle Synchrone

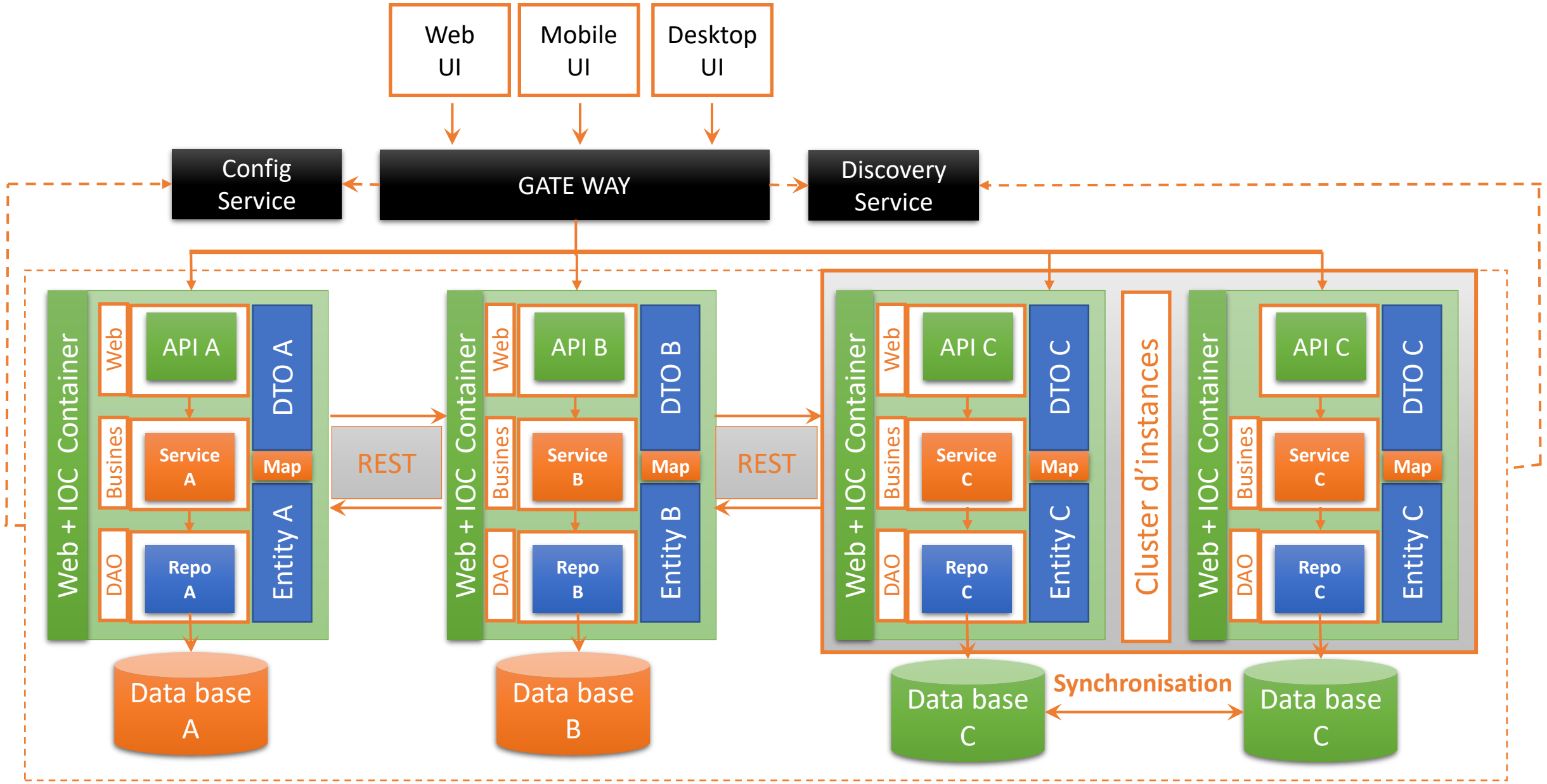




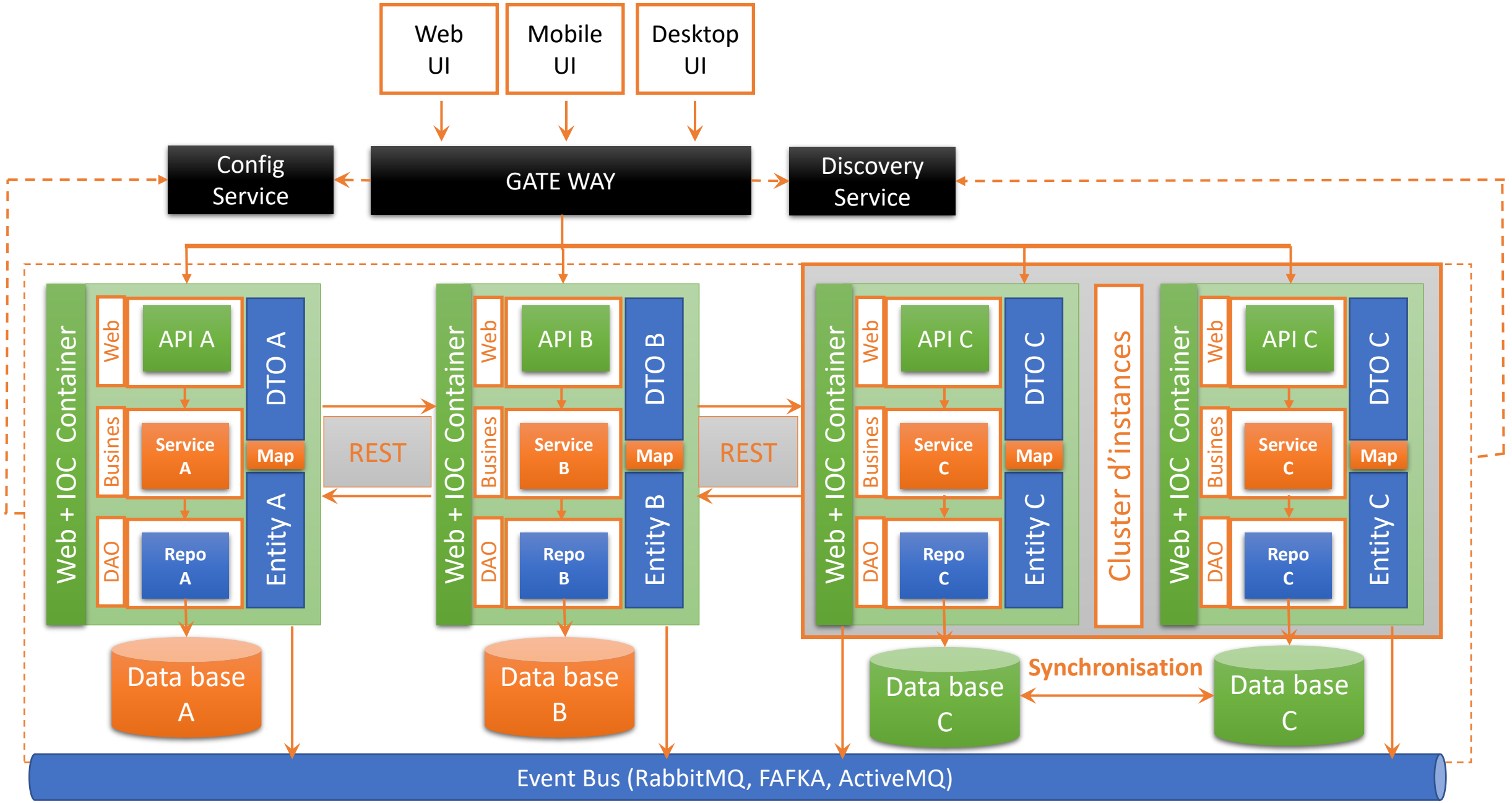
Synchronous Communication Model : REST



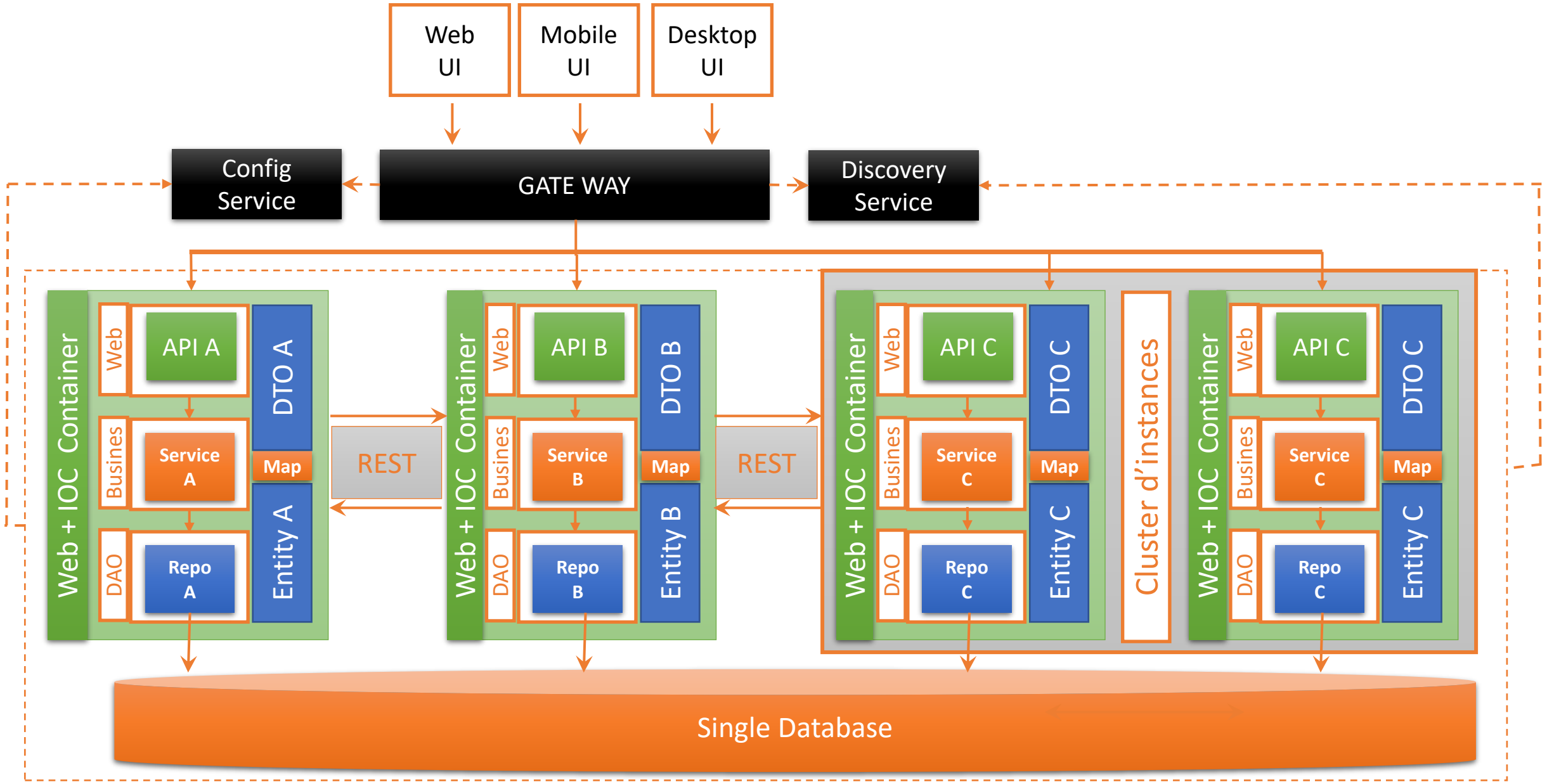
Scalability : How to synchronize multiples instances database ?



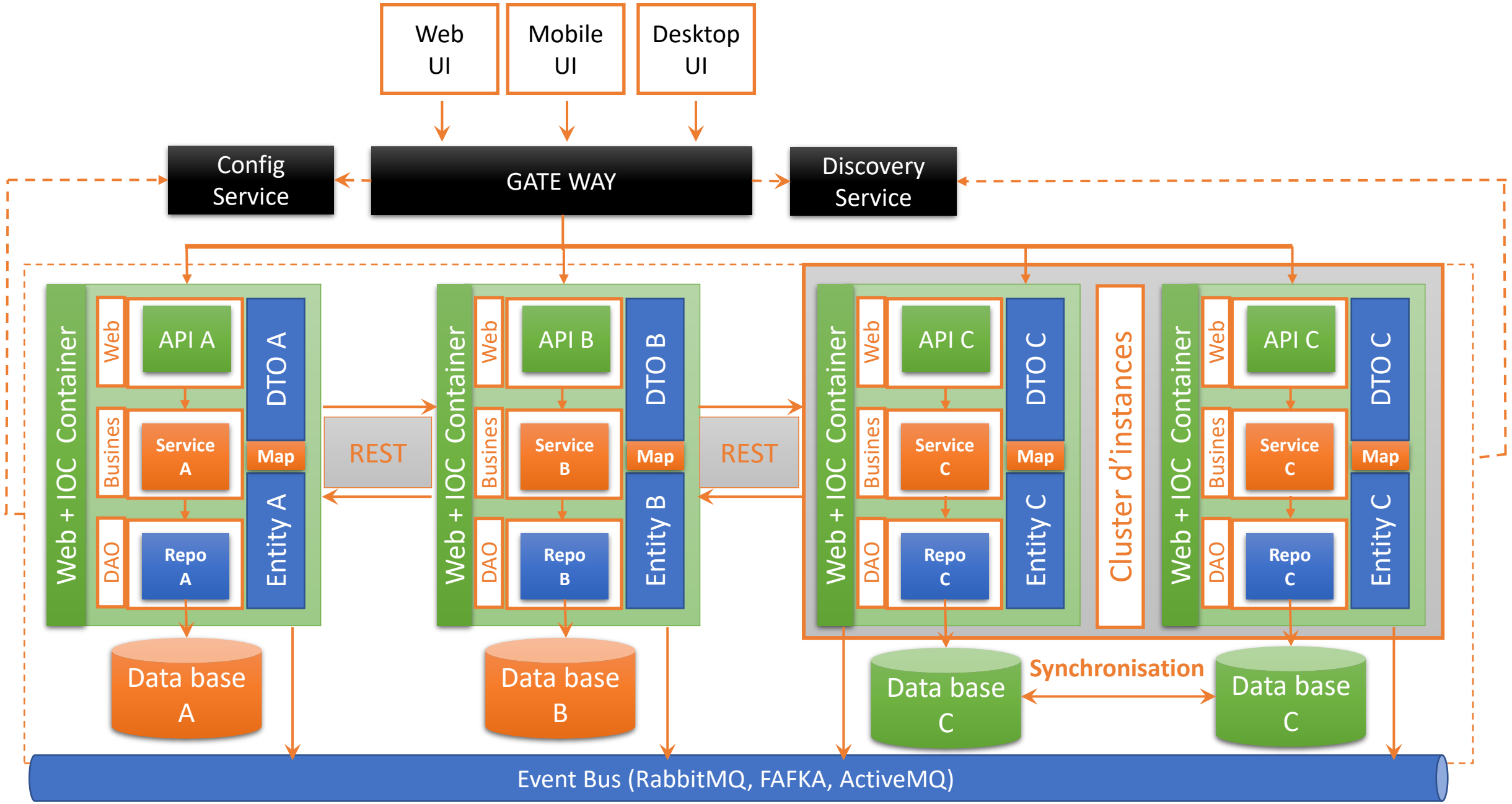
Scalability : How to synchronize multiples instances database ?



Micro-service Architecture with Single database?

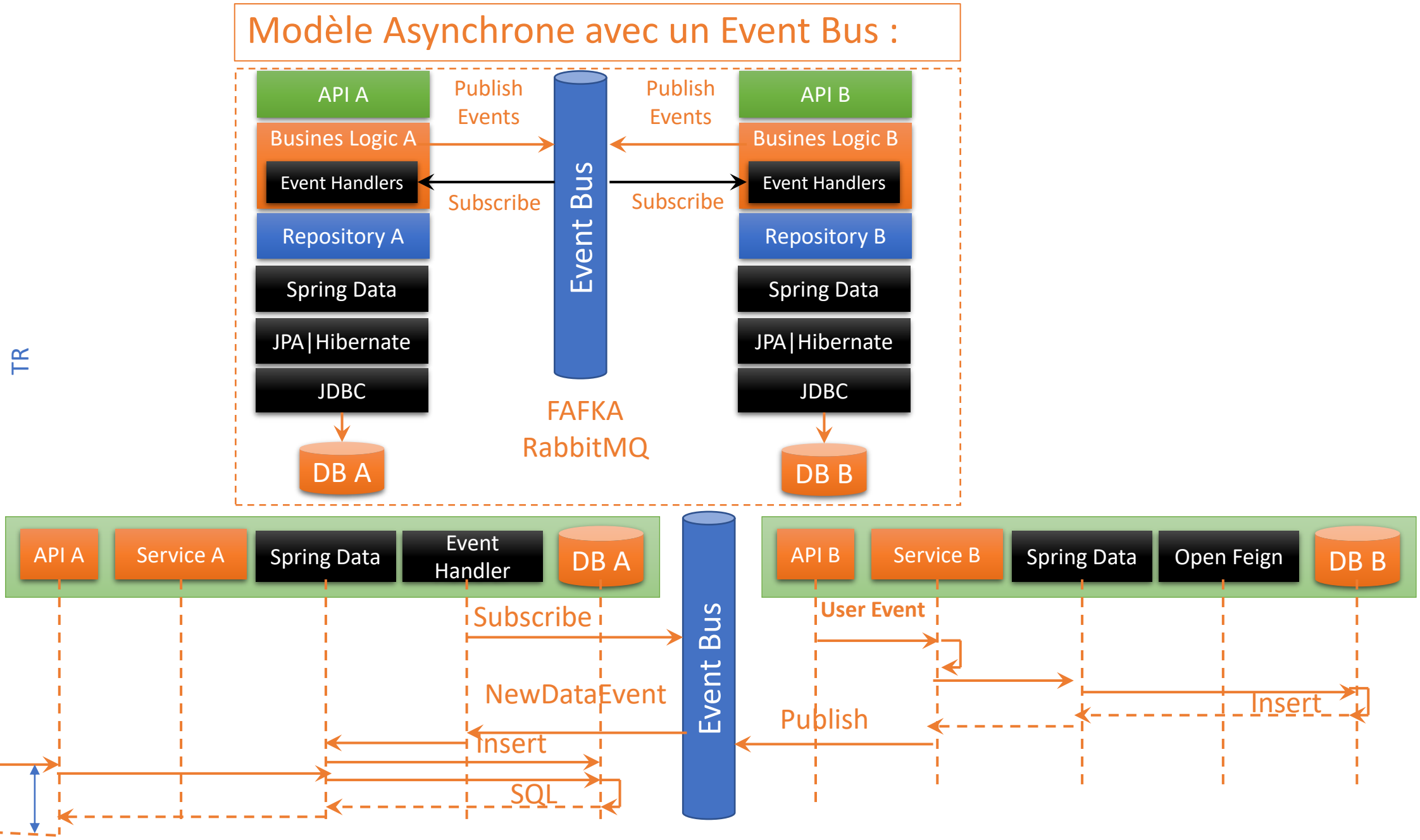


Scalability ?



Quel modèle de communication entre les micro-services

Modèle Asynchrone avec un Event Bus :



CQRS & Event Sourcing

Event Sourcing

- Pattern d'architecture
- Capturer tous les changements de l'état d'une application Comme Séquence d'Evénements
- Ne pas se concentrer sur l'état courant de l'application, mais sur la séquence de changements d'états (Evénements métiers) qui ont abouti à l'état courant
- À partir de cette séquences d'événements, on pourra agréger l'état courant de l'application
- Tout changement de l'état de l'application a une cause unique (Evénement),
- Exemple : Opérations effectuées sur un compte bancaire

			Initial Account Balance	**8000**
N°	**Account_Id**	**date**	**type**	**Amount**
1	001	2021-1-1	DEBIT	**1 200**
2	001	2021-1-1	CREDIT	**6 000**
3	001	2021-1-3	CREDIT	**12 000**
			Current Account Balance	**17 600**

Avantages de l'Event Sourcing

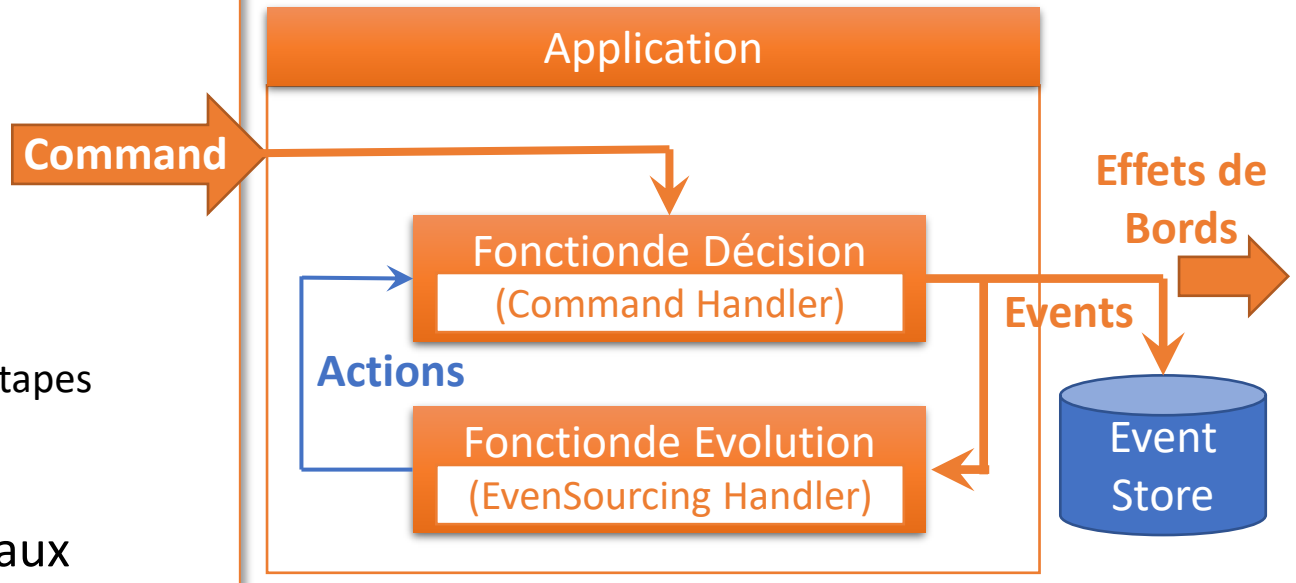
- Base de d'Audit
- Analyse et Debug : Retrouver facilement l'origine des Bugs en Prod.
- Reprise de données : En cas de panne, Rejouer tous les événements métiers enregistrées pour retrouver l'état de l'application
- Performances : Asynchrone avec des Bus de messages qui se mettent bien à l'échelle (Scalability)
- À partir des événements, on peut créer plusieurs projections avec des modèles de données différents

			Initial Account Balance	**8000**
N°	**Account_Id**	**date**	**type**	**Amount**
1	001	2021-1-1	DEBIT	**1 200**
2	001	2021-1-1	CREDIT	**6 000**
3	001	2021-1-3	CREDIT	**12 000**
			Current Account Balance	**17 600**

Architecture et Terminologie Event Sourcing

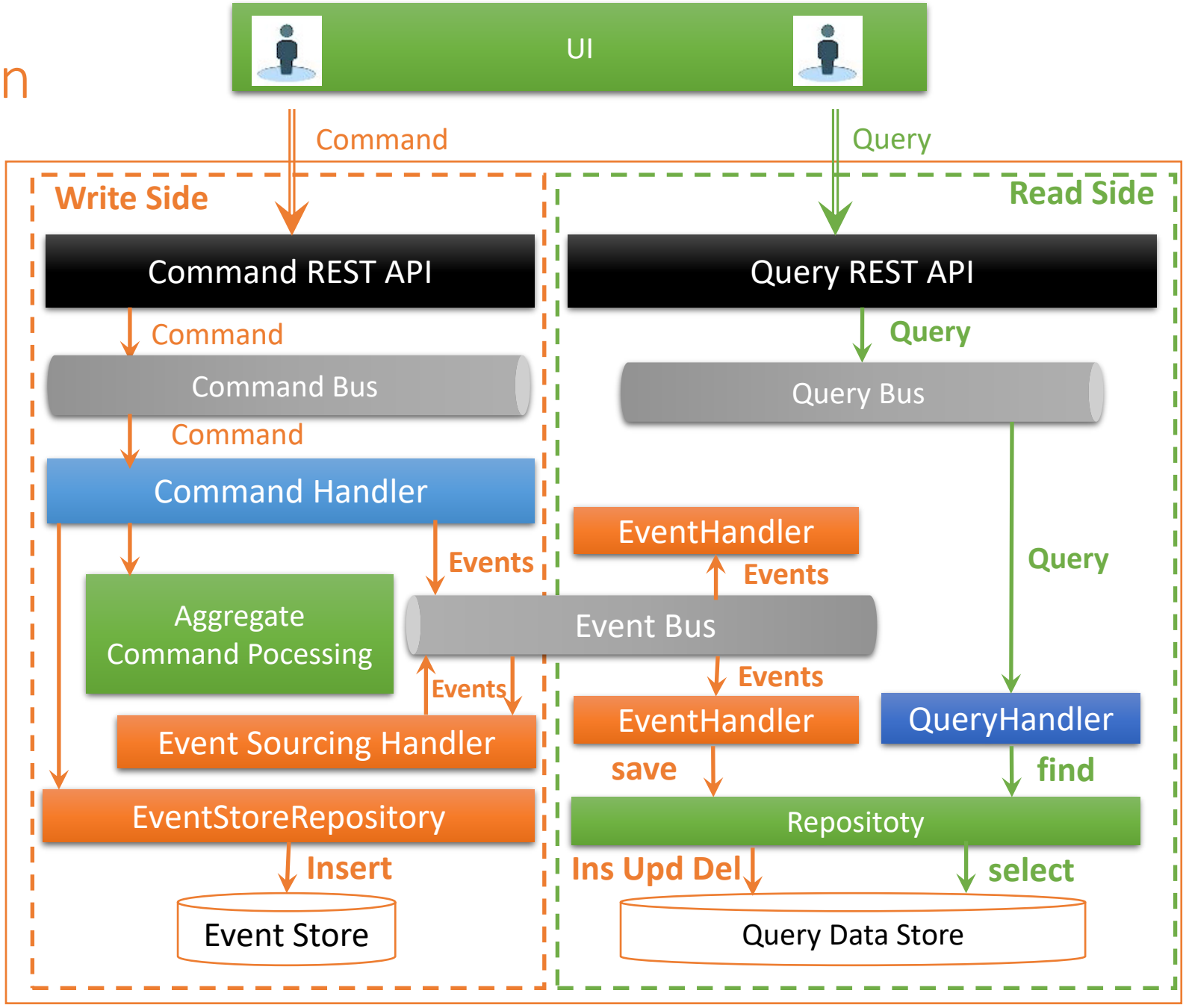
- Com
  -
  -
- Fonc
  mét
  -
  -
- Evér
  -
  -
- Ever
  -

- Fonction d'évolution :
  - Déclenché par les événements émis par la fonction de décision
  - Faire Mutter l'état de l'application
  - Pas de règles métiers
  - (State, Event) => State
- Action : Une commande Interne
  - Commande Produite par l'application
  - Dans le cas ou la décision se fait en plusieurs étapes
  - Le système décide d'agir sur lui-même
- Effets de Bords : Publier les événements aux applications partenaires
  - Une fois qu'on est dans un état stable pour ne pas polluer nos partenaires



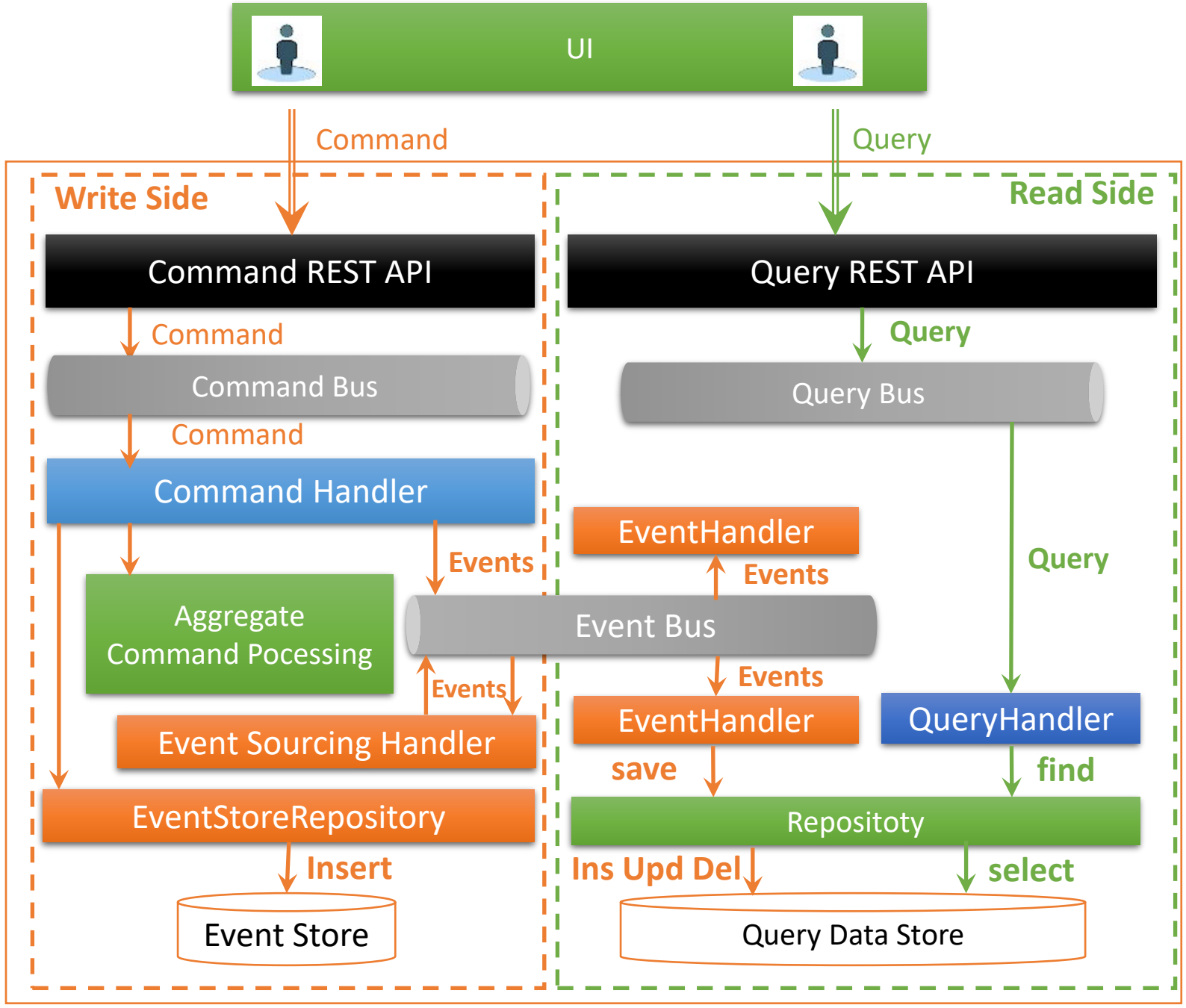
CQRS : Command Query Responsability Segregation

- Pattern qui consiste à séparer la partie lecture de la partie écriture de l'application
- Terminologies :
  - **Command :** Une intension externe de modification de l'état d'un objet (Insert, update, delete)
  - **Query :** Une intention de consultation d'une information ou l'état d'un objet (Select)
  - **Event :** Symbolise une action qui s'est produite dans le système
  - **Event Bus** : un mécanisme qui permet de dispatcher les événements vers les écouteurs des événements (Event Handlers)
    - Peut être n'importe quel système de messagerie comme KAFKA ou RabbitMQ
  - **Event Store :** Base de persistance des événements publiés dans l'application.
  -



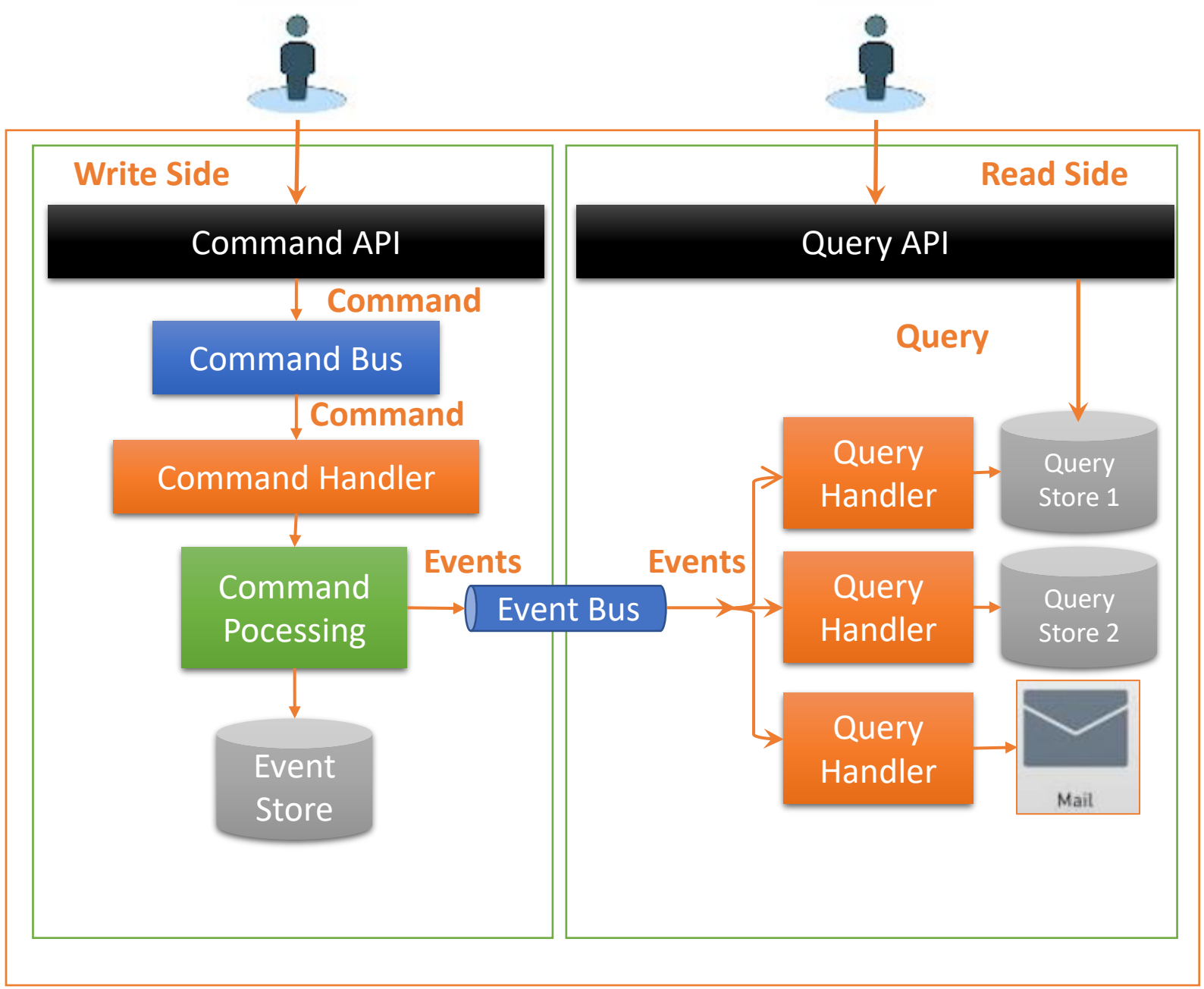
Avantages de CQRS

- On peut agréger facilement les données de plusieurs micro-services
- Mise à l'échelle (Scale) séparément des deux parties Lecture (90%) et Ecriture (10%)
- Liberté de choisir des types de SGBD différents pour l'écriture et pour les lectures
  - PostGres pour Event Store
  - MySQL pour lectures
  - ElasticSerach pour le moteur de recherche
- Faciliter la séparation des aspects de lectures et d'écritures
- Faciliter la séparation des deux modèles pour la lecture et écriture



Inconvénients de CQRS

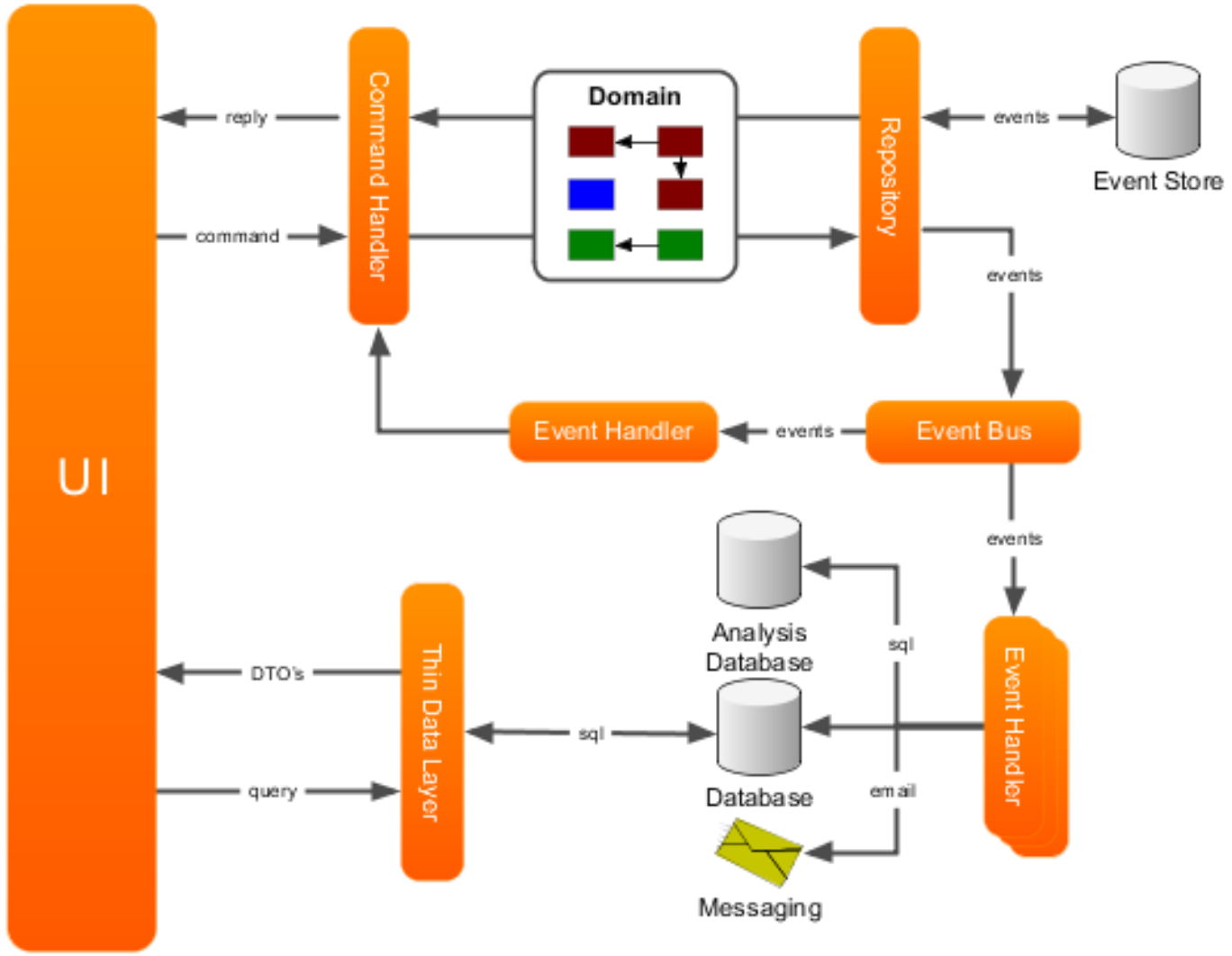
- Amplifie la complexité du système
- Duplication du code :
  - Penser à créer des Core-Laibrairies pour le code en commun.
- Contraintes de consistance entre les bases de lectures et d'écriture



AXON Framework

La partie UI :

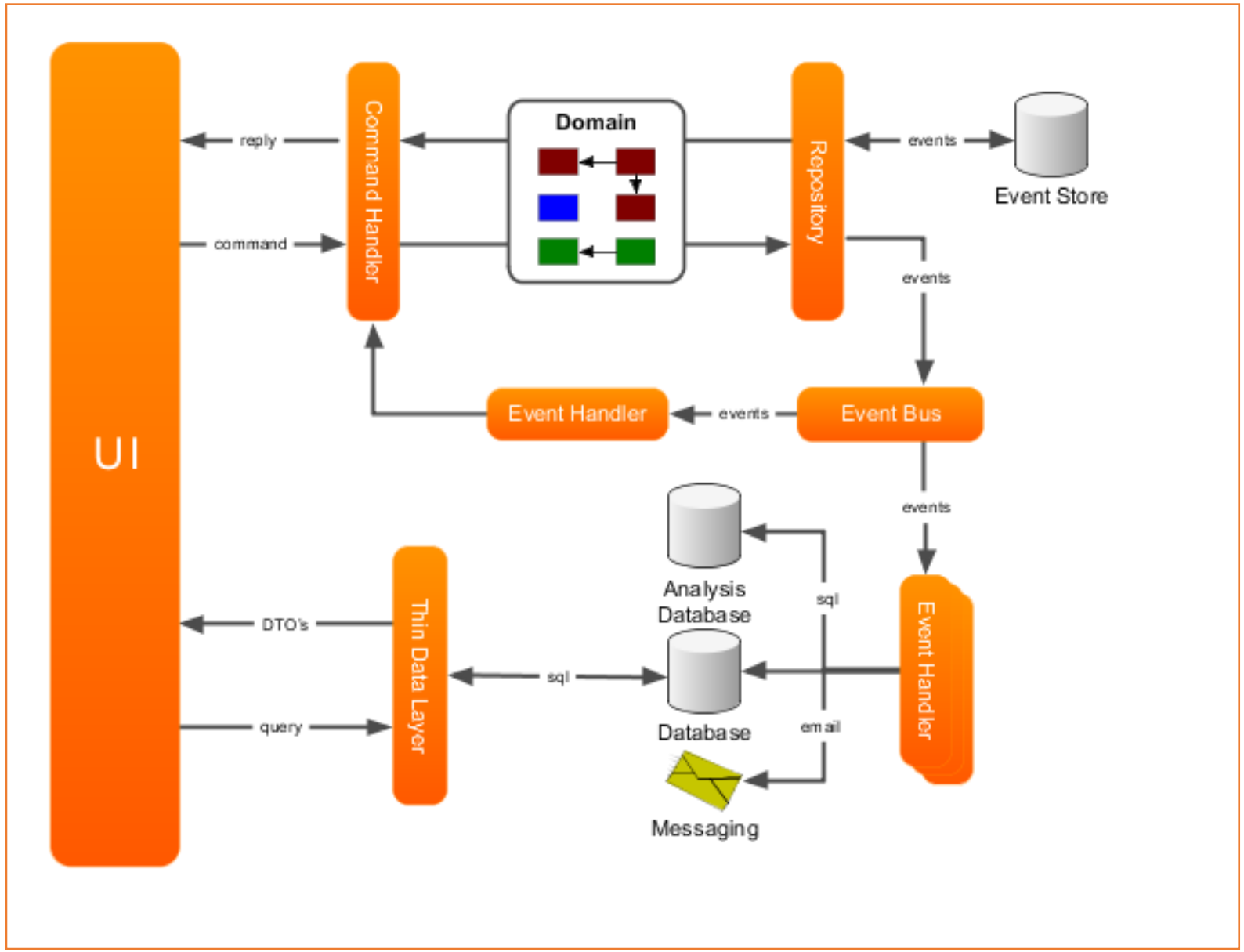
- interagit avec l'application de deux manières :
  - Emettre des commandes qui vont entrainer la modification de l'état de de l'application dans la base d'écriture
  - Emettre des requêtes pour consulter l'état de l'application à partir des bases de lectures



AXON Framework

Les commandes :

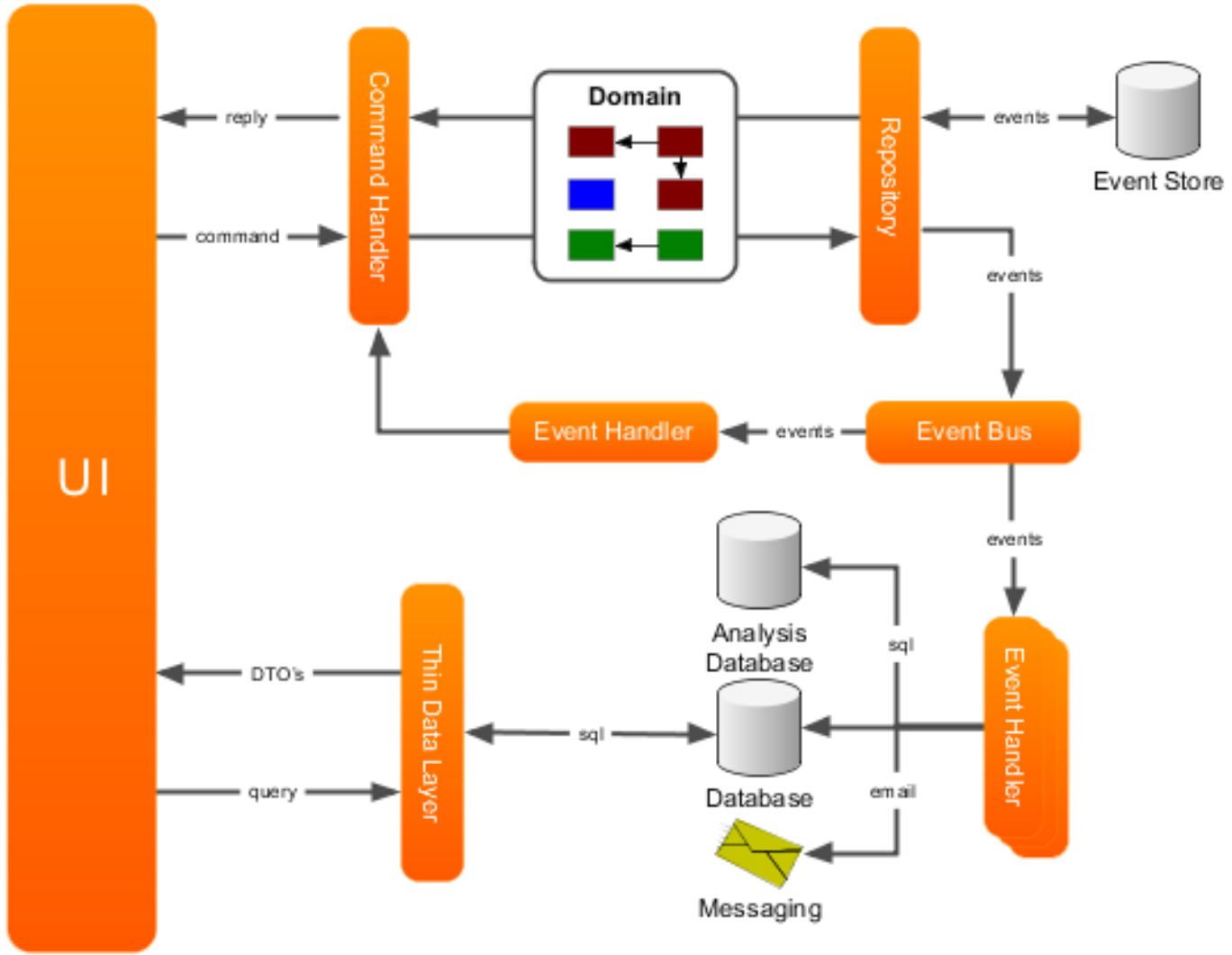
- Objets simples qui contiennent toutes les données nécessaires à un **Command Handler** pour les exécuter.
- Une commande exprime son intention par son nom : Le nom de la classe est utilisé pour déterminer ce qui doit être fait et que les champs de la commande fournissent les informations nécessaires pour le faire.



AXON Framework

Command Bus :

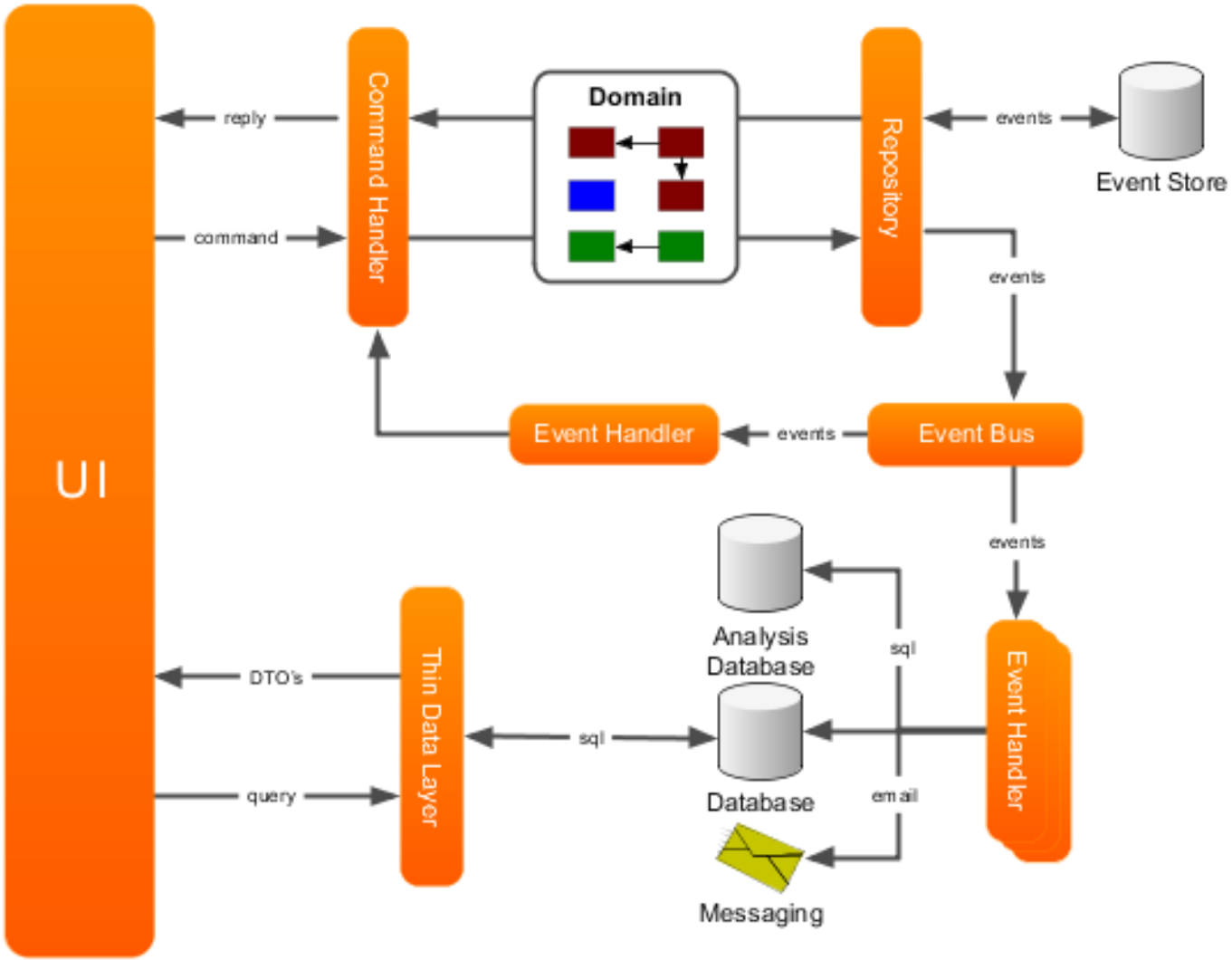
- Le bus de commande reçoit des commandes et les achemine vers les Command Handlers.
- Chaque Command Handler réagit à un type de commande spécifique et exécute une logique métier basée sur le contenu de la commande.
- Il est possible d'exécuter une logique quel que soit le type de commande, comme la validation, la journalisation ou l'autorisation.



AXON Framework

Command Handler :

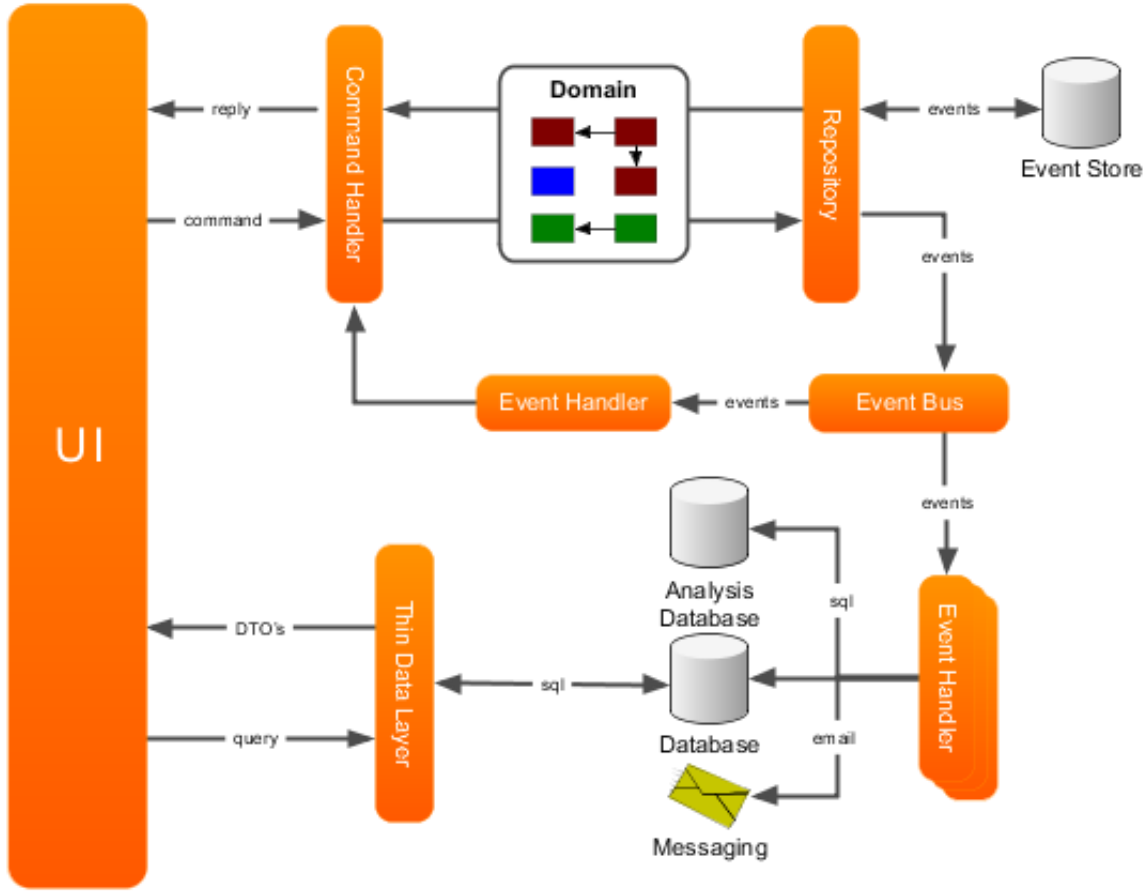
- Le gestionnaire de commandes récupère les objets de domaine (agrégats) d'un Repository et exécute des méthodes sur eux pour modifier leur état.
- Ces agrégats contiennent généralement la logique métier et sont donc chargés de protéger leurs propres invariants.
- Les changements d'état des agrégats entraînent la génération d'événements de domaine.
- Les événements de domaine et les agrégats forment le modèle de domaine.



AXON Framework

Repositoty:

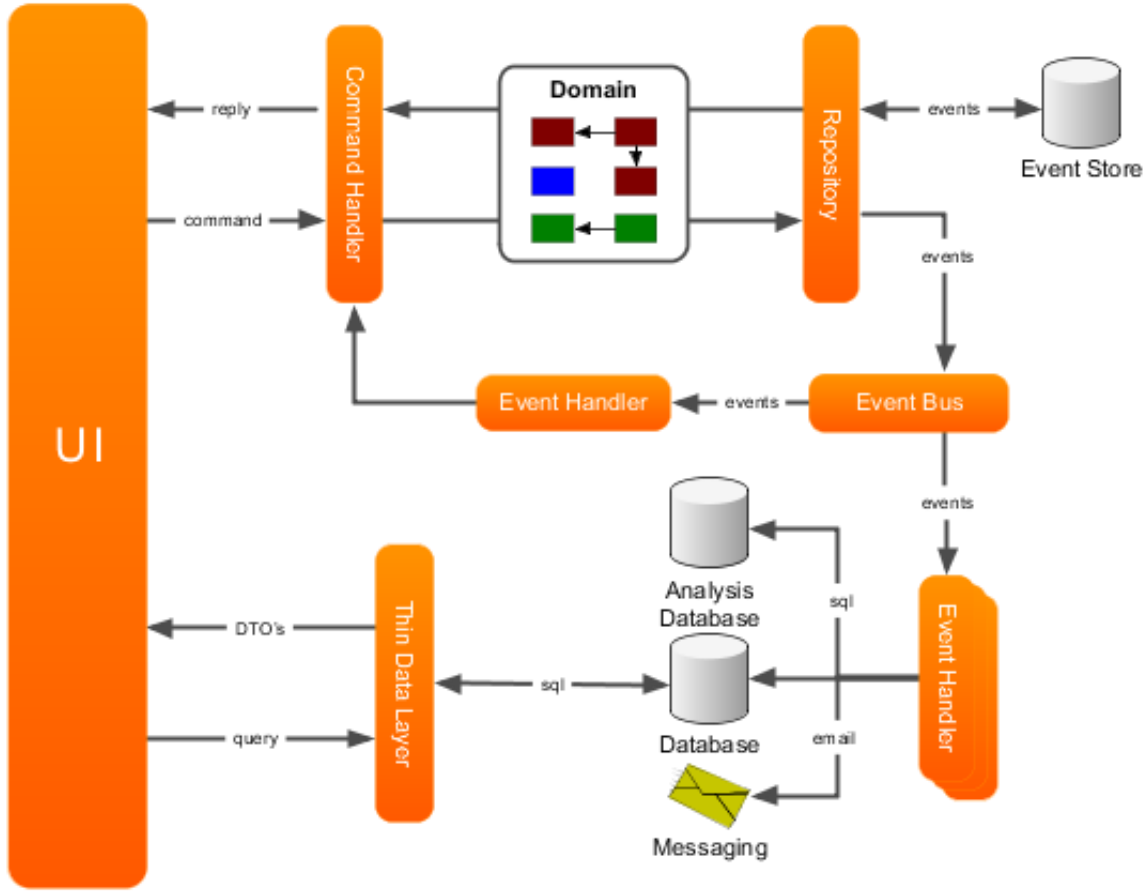
- Les Repositories sont chargés de fournir l'accès aux agrégats.
- En règle générale, ces Repositories sont optimisés pour la recherche d'un agrégat par son identifiant unique uniquement.
- Certains Repositories stockeront l'état de l'agrégat lui-même dans la base de ornées,
- D'autres Repositories stockeront les changements d'état que l'agrégat a subis dans un Event Store.
- Le Repository est également responsable de la persistance des modifications apportées aux agrégats dans son unité de sauvegarde.
- Axon prend en charge à la fois la méthode directe de persistance des agrégats et l'Event Sourcing.



AXON Framework

Event Bus:

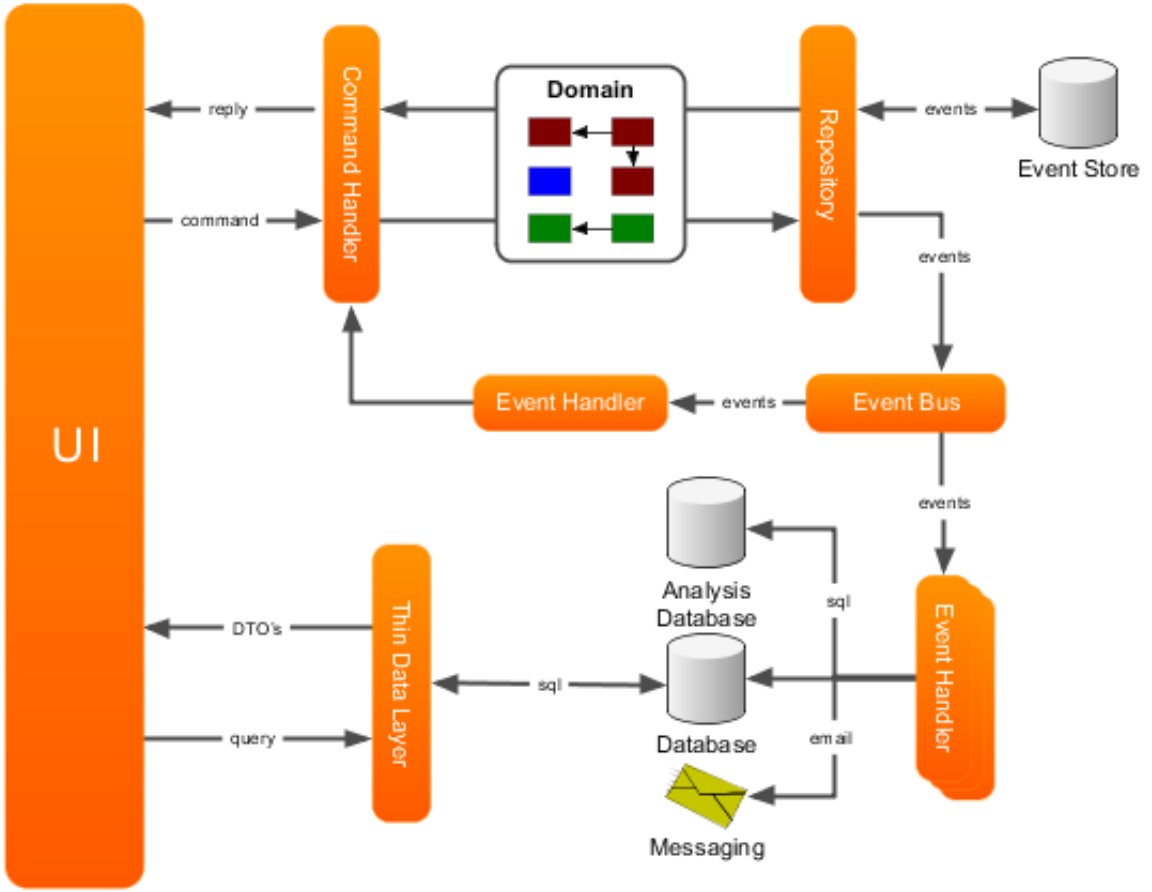
- Le bus d'événements distribue des événements à tous les **Event Handlers** intéressés.
- Cela peut être fait de manière synchrone ou asynchrone.
- La distribution **d'événements asynchrone** permet à l'exécution de la commande de céder le contrôle à l'utilisateur, tandis que les événements sont distribués et traités en arrière-plan.
- Ne pas avoir à attendre la fin du traitement des événements rend une application **plus réactive**.
- Le traitement synchrone des événements, en revanche, est plus simple et constitue le mode par défaut.
- Par défaut, **le traitement synchrone** traitera les écouteurs d'événements dans **la même transaction** qui a également traité la commande.



AXON Framework

Event Handlers:

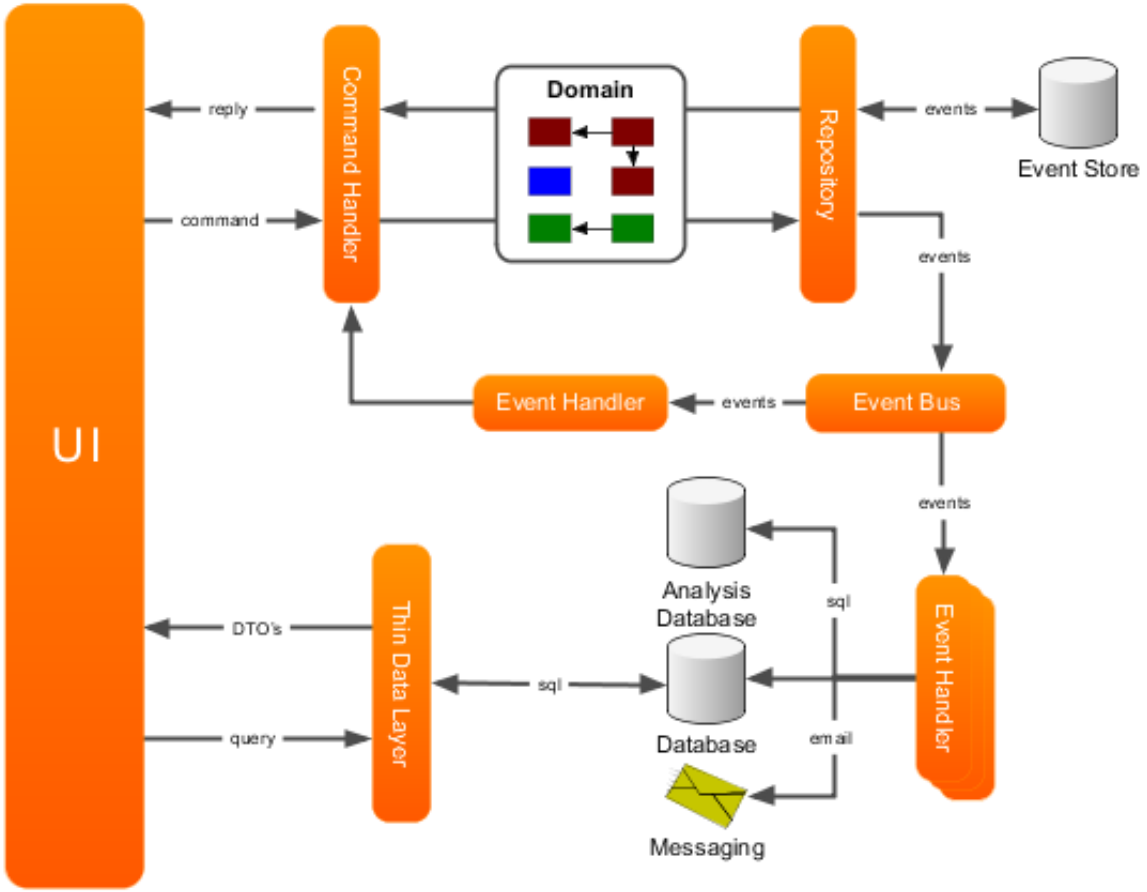
- Les écouteurs d'événements reçoivent des événements et les traitent.
- Certains gestionnaires mettront à jour la base de lecture
- D'autres enverront des messages à des systèmes externes.
- Comme vous pouvez le remarquer, les gestionnaires de commandes ignorent totalement les composants qui sont intéressés par les modifications qu'ils apportent.
- Pour étendre l'application avec de nouvelles fonctionnalités. tout ce que vous avez à faire est d'ajouter un autre Event Handlet.
- Les événements permettent de coupler faiblement les composants de votre application.



AXON Framework

Event Handlers:

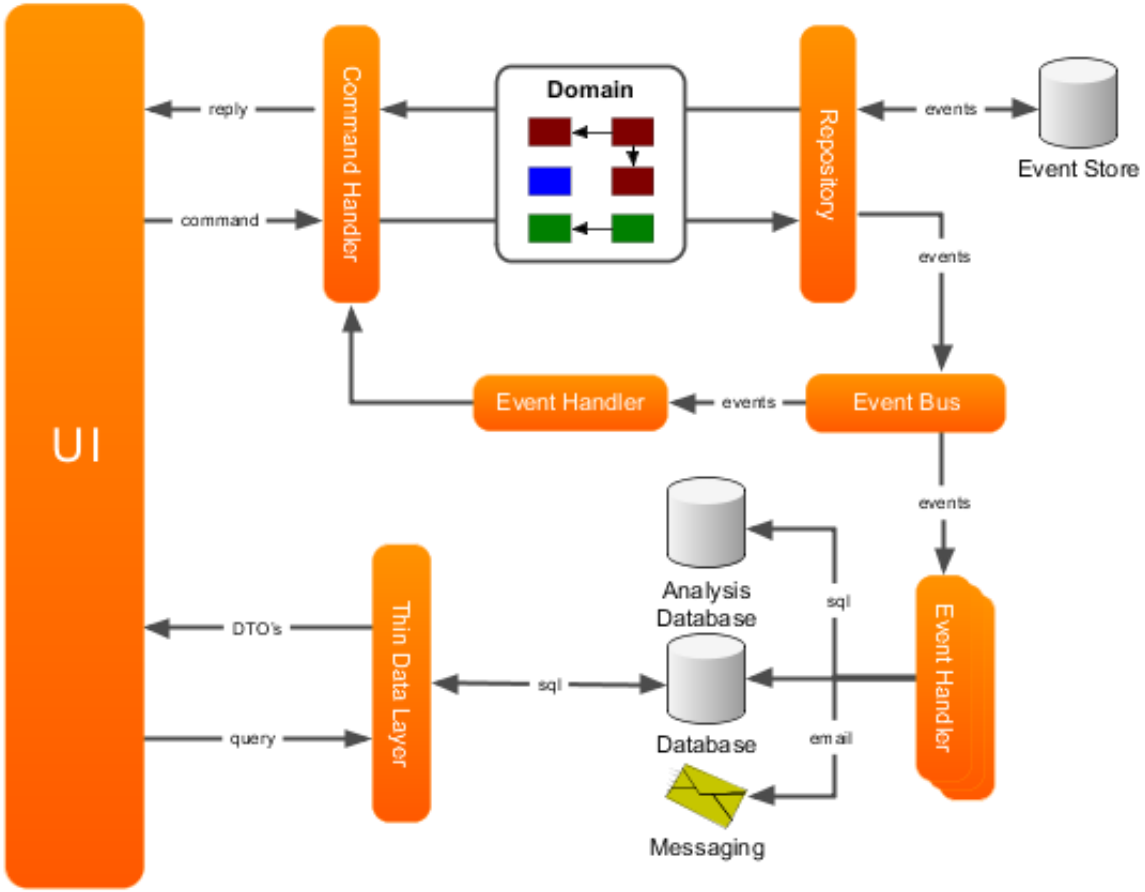
- Dans certains cas, le traitement des événements nécessite l'envoi de nouvelles commandes à l'application.
- Un exemple est lorsqu'une commande est reçue. Cela pourrait signifier que le compte du client doit être débité du montant de l'achat et qu'une commande d'expédition doit être dispatchée pour préparer l'expédition des marchandises achetées.
- Dans de nombreuses applications, la logique deviendra plus compliquée que cela : et si le client ne payait pas à temps ? Allez-vous expédier l'envoi immédiatement ou attendre le paiement en premier ? La saga est le concept du CQRS qui est chargé de gérer ces transactions commerciales complexes.



AXON Framework

Query Handler:

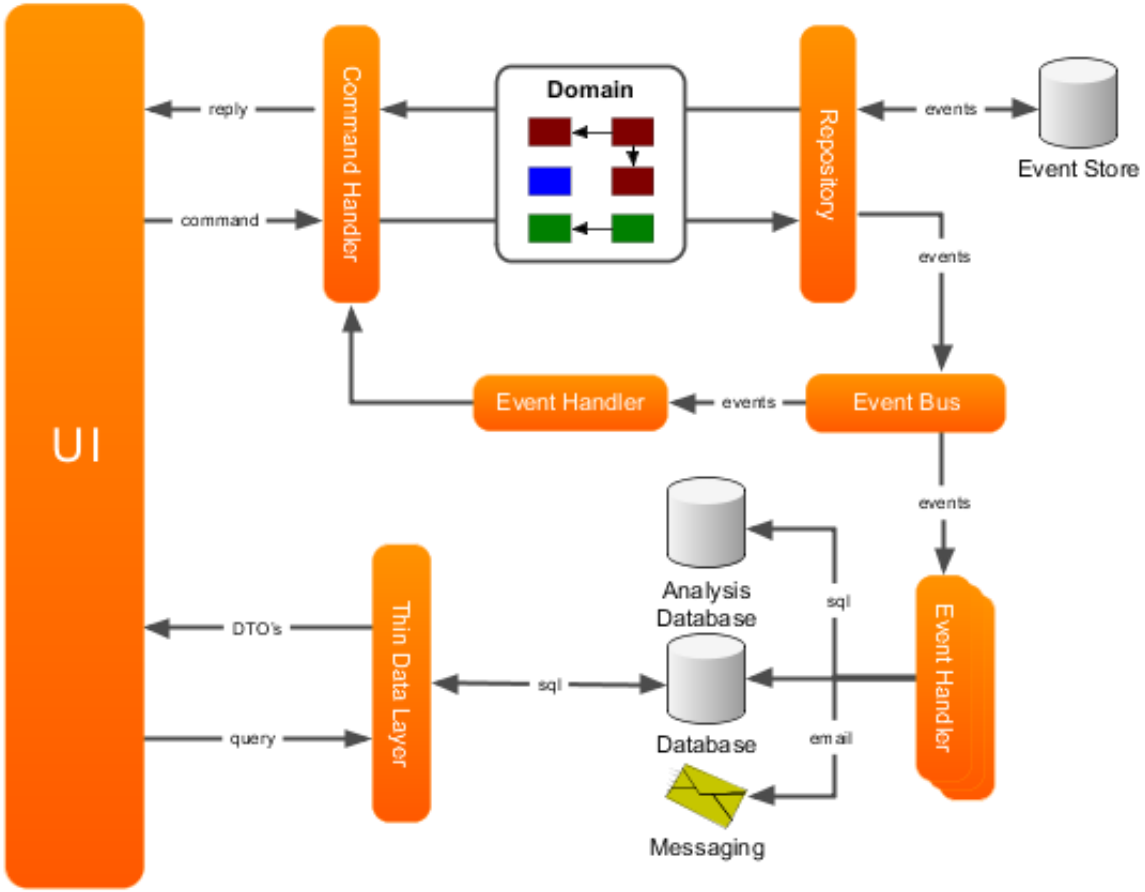
- Depuis Axon 3.1, le framework fournit des composants pour gérer les requêtes.
- Le **Query Bus** reçoit les requêtes et les achemine vers les Query Handlers.
- Un **Query Handler** est enregistré sur le bus de requêtes avec à la fois **le type de requête** qu'il gère et **le type de réponse** qu'il fournit.
- La requête et le type de résultat sont généralement des objets DTO simples en lecture seule.



AXON Framework

Query Handler:

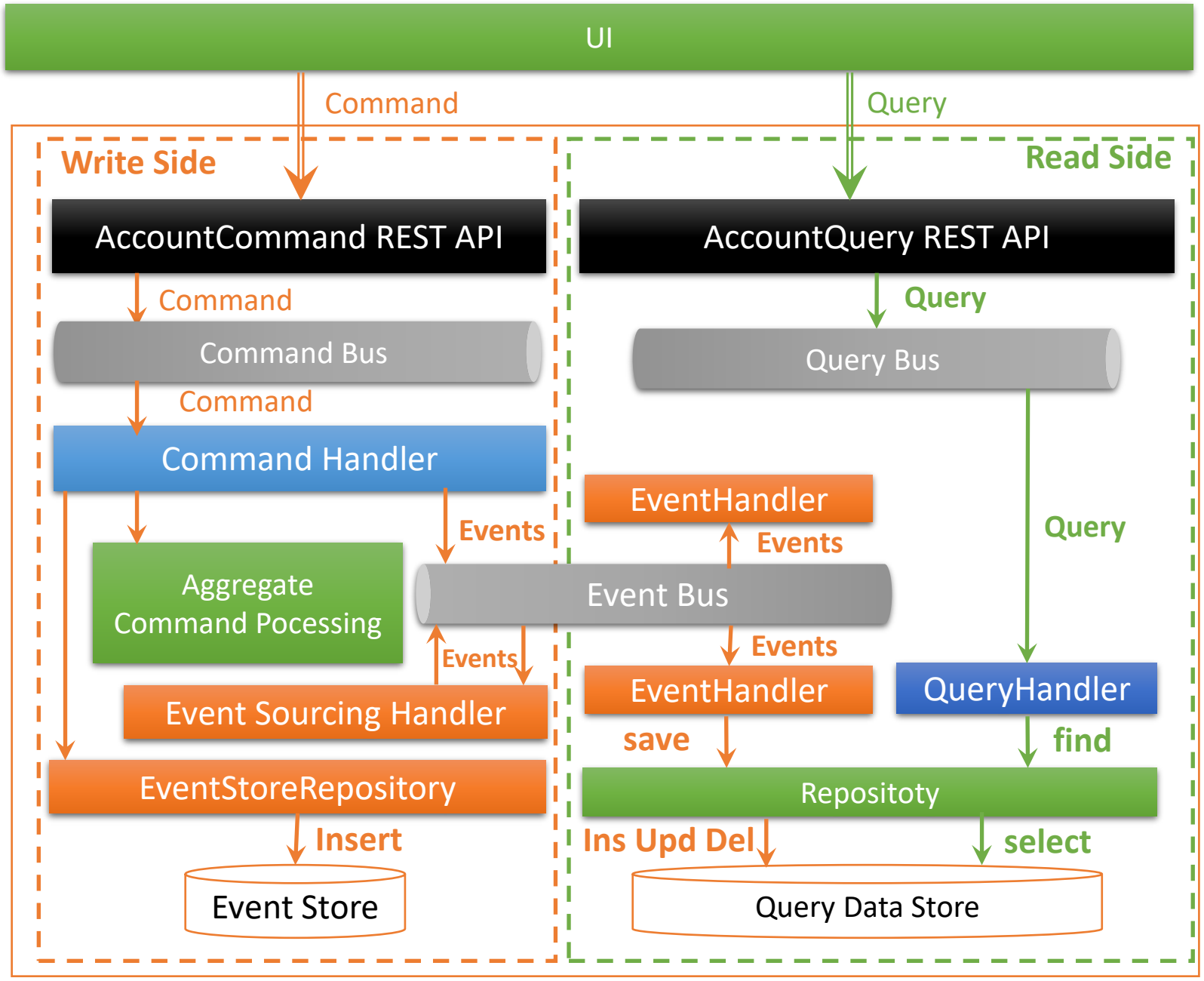
- Il est possible d'enregistrer plusieurs gestionnaires de requêtes pour le même type d’input et le même type de réponse.
- Lors de l'envoi des requêtes, le client peut indiquer s'il souhaite un résultat d'un ou de tous les gestionnaires de requêtes disponibles.

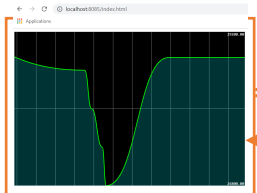


https://github.com/mohamedYoussfi/cqrs-eventsourcing-axon

Application

- Créer une application qui permet de gérer des comptes bancaires.
- Permet de :
  - Ajouter un Compte
  - Activer un compte après création
  - Créditer un compte
  - Débiter un compte
  - Consulter un compte
  - Consulter les comptes
  - Consulter les opérations d'un compte
  - Suivre en temps réel l'état d'un compte





UI

AccountCommandController

CommandGateway

send(cmd:Object) : void

CreateAccountCommand

- accountId : String
- balance : BigDecimal

Command BUS

AccountAggregate

- accountId : String
- balance : BigDecimal
- Status : String

CreateAccountCommand Handler

AccountQueryController

QueryGateway

query(query,resp) : void

query

reply

Query BUS

Event BUS

AccountCreatedEvent Sourcing Handler

AccountCreatedEvent

- accountId : String
- balance : BigDecimal

AccountService

CreateAccountQuery Handler

AccountCreatedEvent Handler

query

reply

Account Entity

AccountRepository

Spring Data

JPA | Hibernate| JDBC

Account DB

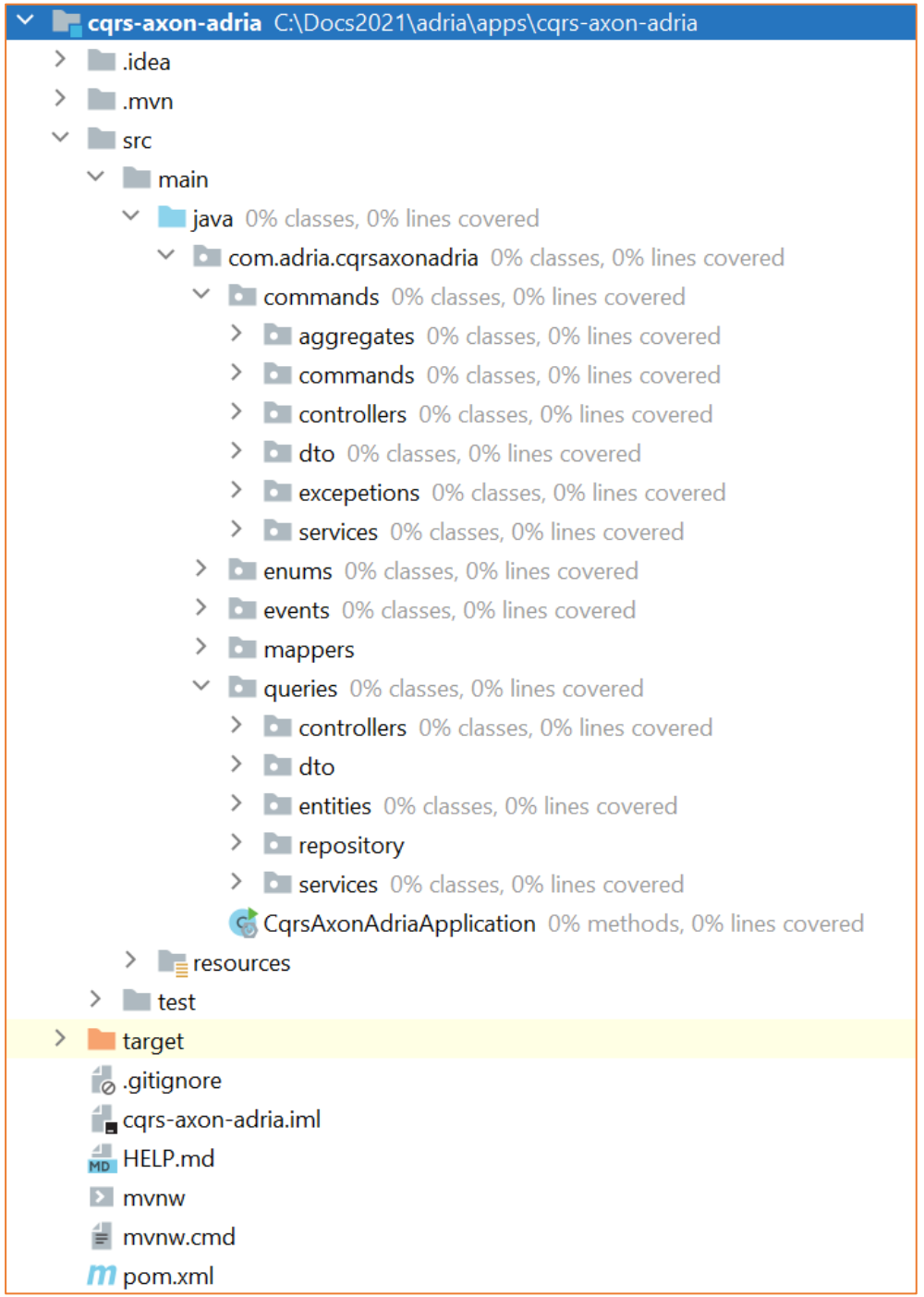
Account Created Event

AgregateLifeCycle

EventStoreRepository

Event Store

Structure du projet



Maven Dependencies

```
<parent>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-parent</artifactId>
   <version>2.5.2</version>
   <relativePath/>
</parent>
```

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
   <groupId>mysql</groupId>
   <artifactId>mysql-connector-java</artifactId>
</dependency>
```

AXON Framework


```
<dependency>
   <groupId>org.axonframework</groupId>
   <artifactId>axon-spring-boot-starter</artifactId>
   <version>4.4.3</version>
   <exclusions>
      <exclusion>
         <groupId>org.axonframework</groupId>
         <artifactId>axon-server-connector</artifactId>
      </exclusion>
   </exclusions>
</dependency>
```

Maven Dependencies

```
<dependency>
   <groupId>org.projectlombok</groupId>
   <artifactId>lombok</artifactId>
   <optional>true</optional>
</dependency>
```

Lombok

```
<dependency>
   <groupId>org.springdoc</groupId>
   <artifactId>springdoc-openapi-ui</artifactId>
   <version>1.5.2</version>
</dependency>
```

Open API

```
<dependency>
   <groupId>org.mapstruct</groupId>
   <artifactId>mapstruct</artifactId>
   <version>1.4.2.Final</version>
</dependency>
```

MapStruct

Maven Dependencies

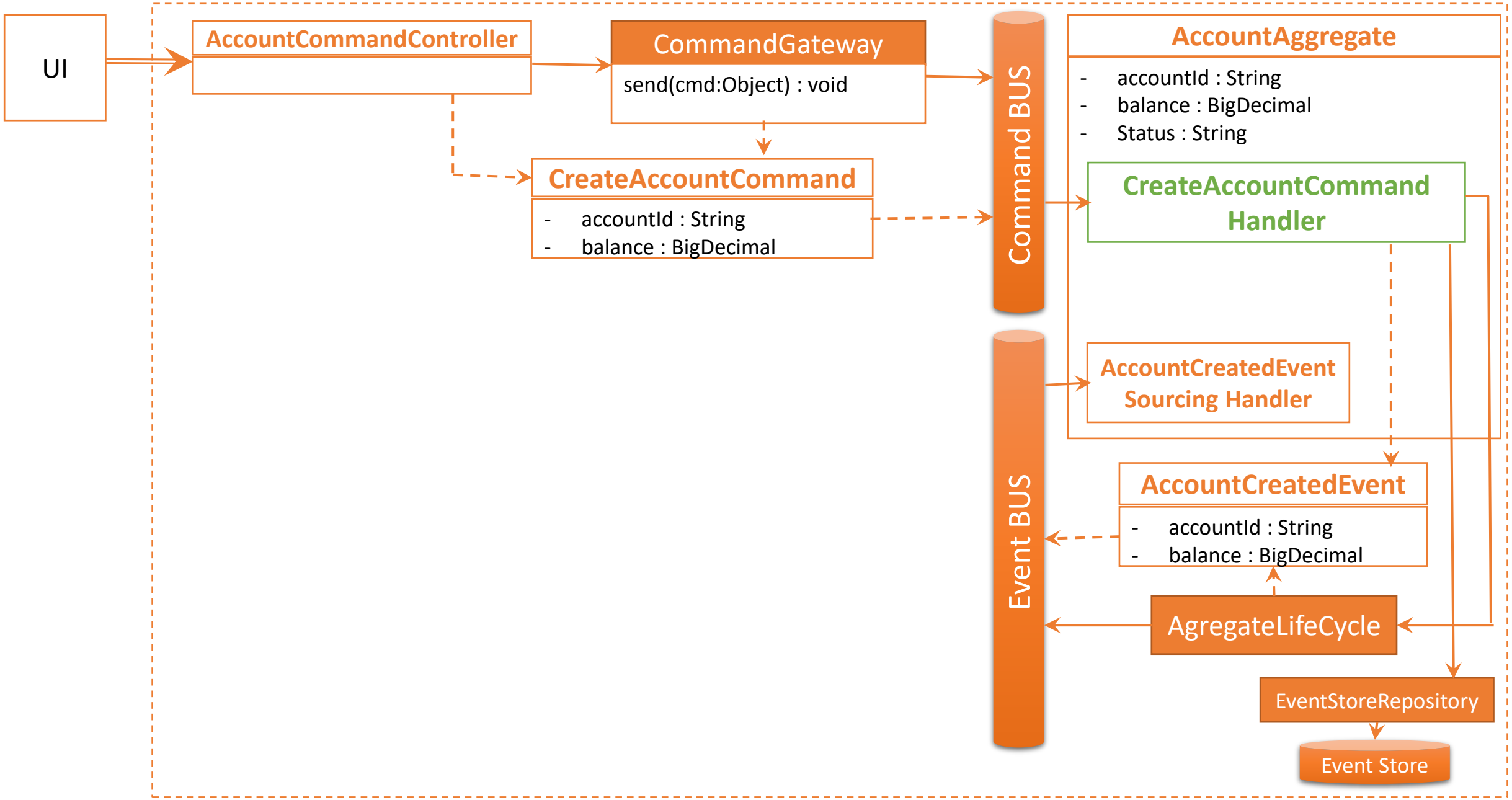
```
<dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-webflux</artifactId>
   <version>5.3.8</version>
</dependency>
```

Maven Dependencies

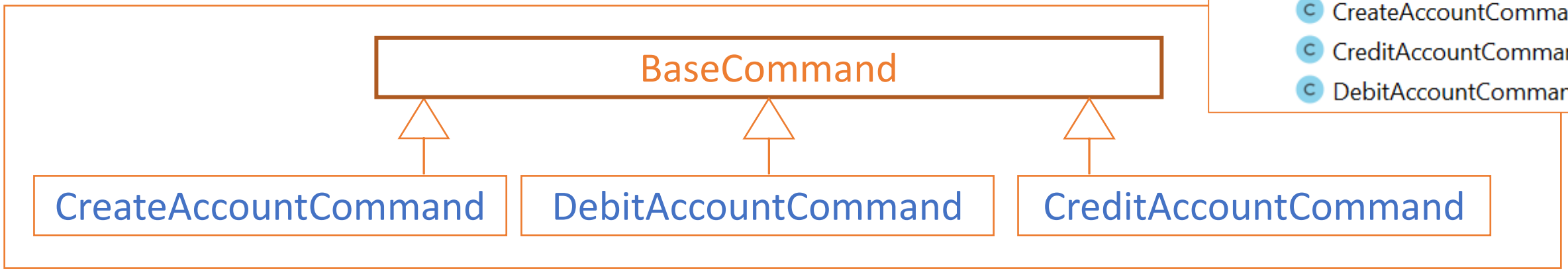
Plugin Java Compiler
MapStruct Processor
Configuration

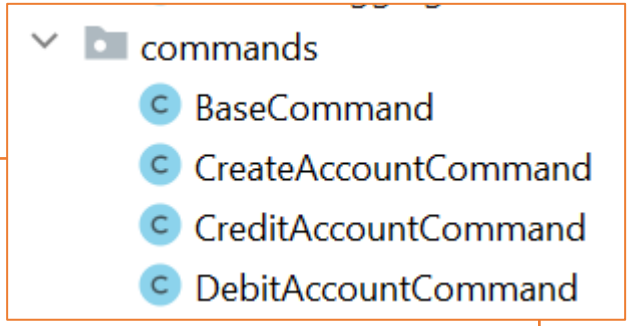
```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-compiler-plugin</artifactId>
  <version>3.8.1</version>
  <configuration>
    <source>1.8</source> <!-- depending on your project -->
    <target>1.8</target> <!-- depending on your project -->
    <annotationProcessorPaths>
      <path>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <version>1.18.16</version>
      </path>
      <path>
        <groupId>org.mapstruct</groupId>
        <artifactId>mapstruct-processor</artifactId>
        <version>1.4.2.Final</version>
      </path>
    </annotationProcessorPaths>
  </configuration>
</plugin>
```

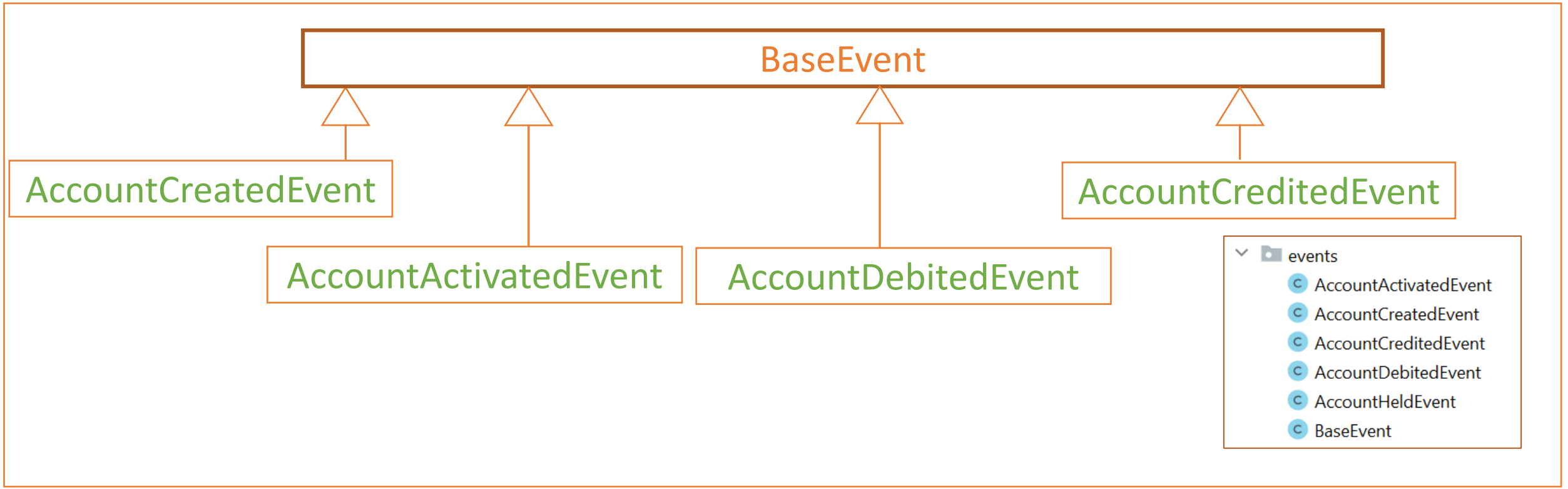
Commands Side



Commands and Events







Commands

```
public class BaseCommand<T> {
    @TargetAggregateIdentifier
    @Getter
    private T id;
    public BaseCommand(T id) {
        this.id = id;
    }
}
```

CreateAccountCommand

```
public class CreateAccountCommand extends BaseCommand<String> {
    @Getter private BigDecimal balance;
    @Getter private String currency;
    public CreateAccountCommand(String id, BigDecimal balance, String currency) {
        super(id);
        this.balance = balance;
        this.currency = currency;
    }
}
```

Commands

```
public class BaseCommand<T> {
    @TargetAggregateIdentifier
    @Getter
    private T id;
    public BaseCommand(T id) {
        this.id = id;
    }
}
```

CreditAccountCommand

```
public class CreditAccountCommand extends BaseCommand<String> {
    @Getter private BigDecimal amount;
    @Getter private String currency;
    public CreditAccountCommand(String id, BigDecimal amount, String currency) {
        super(id);
        this.amount = amount;
        this.currency = currency;
    }
}
```

Commands

```
public class BaseCommand<T> {
    @TargetAggregateIdentifier
    @Getter
    private T id;
    public BaseCommand(T id) {
        this.id = id;
    }
}
```

DebitAccountCommand

```
public class DebitAccountCommand extends BaseCommand<String> {
    @Getter private BigDecimal amount;
    @Getter private String currency;
    public DebitAccountCommand(String id, BigDecimal amount, String currency) {
        super(id);
        this.amount = amount;
        this.currency = currency;
    }
}
```

Events

```
public enum AccountStatus {
    CREATED,ACTIVATED
}
```

```
public class BaseEvent<T> {
    @Getter private T id;
    public BaseEvent(T id) {
        this.id = id;
    }
}
```

AccountCreatedEvent

```
public class AccountCreatedEvent extends BaseEvent<String> {
    @Getter private final BigDecimal balance;
    @Getter private final String currency;
    @Getter private final AccountStatus status;
    public AccountCreatedEvent(String id, BigDecimal balance, String currency, AccountStatus status) {
        super(id);
        this.balance = balance;
        this.currency = currency;
        this.status = status;
    }
}
```

Events

```
public class BaseEvent<T> {
    @Getter private T id;
    public BaseEvent(T id) {
        this.id = id;
    }
}
```

AccountActivatedEvent

```
public class AccountActivatedEvent extends BaseEvent<String> {
    @Getter private final AccountStatus status;
    public AccountActivatedEvent(String id, AccountStatus status) {
        super(id);
        this.status = status;
    }
}
```

Events

```java
public class BaseEvent<T> {
    @Getter private T id;
    public BaseEvent(T id) {
        this.id = id;
    }
}
```

AccountCreditedEvent

```java
public class AccountCreditedEvent extends BaseEvent<String> {
    @Getter private final BigDecimal amount;
    @Getter private final String currency;

    public AccountCreditedEvent(String id, BigDecimal amount, String currency) {
        super(id);
        this.amount = amount;
        this.currency = currency;
    }
}
```

Events

```
public class BaseEvent<T> {
    @Getter private T id;
    public BaseEvent(T id) {
        this.id = id;
    }
}
```

AccountDebitedEvent

```
public class AccountDebitedEvent extends BaseEvent<String> {
    @Getter private final BigDecimal amount;
    @Getter private final String currency;

    public AccountDebitedEvent(String id, BigDecimal amount, String currency) {
        super(id);
        this.amount = amount;
        this.currency = currency;

    }
}
```

Events

```
public class BaseEvent<T> {
    @Getter private T id;
    public BaseEvent(T id) {
        this.id = id;
    }
}
```

AccountDebitedEvent

```
public class AccountDebitedEvent extends BaseEvent<String> {
    @Getter private final BigDecimal amount;
    @Getter private final String currency;

    public AccountDebitedEvent(String id, BigDecimal amount, String currency) {
        super(id);
        this.amount = amount;
        this.currency = currency;

    }
}
```

Aggregate

AccountAggregate

```
//@Entity
@Aggregate
@Slf4j
public class AccountAggregate {
    @AggregateIdentifier
    //@Id
    private String accountId;
    private BigDecimal balance;
    private String currency;
    private String status;

    public AccountAggregate() {
    }
```

Commands: Aggregate

AccountAggregate

```
@CommandHandler
   public AccountAggregate(CreateAccountCommand createAccountCommand) {
      log.info("CreateAccountCommand Received");
      AggregateLifecycle.apply(
            new AccountCreatedEvent(
                  createAccountCommand.getId(),
                  createAccountCommand.getBalance(),
                  createAccountCommand.getCurrency(),
                  AccountStatus.CREATED
                  )
      );
   }
```

Commands: Aggregate

AccountAggregate

```
@EventSourcingHandler
  public void handle(AccountCreatedEvent accountCreatedEvent){
     log.info("AccountCreatedEvent Occured");
     this.accountId=accountCreatedEvent.getId();
     this.balance=accountCreatedEvent.getBalance();
     this.currency=accountCreatedEvent.getCurrency();
     this.status=String.valueOf(accountCreatedEvent.getStatus());
     AggregateLifecycle.apply(new
AccountActivatedEvent(this.accountId,AccountStatus.ACTIVATED));
  }
@EventSourcingHandler
  public void on(AccountActivatedEvent accountActivatedEvent){
     log.info("AccountActivatedEvent Occured");
     this.status=String.valueOf(accountActivatedEvent.getStatus());
  }
```

Commands: Aggregate

AccountAggregate

```
@CommandHandler
   public void handle(DebitAccountCommand debitAccountCommand){
if((this.balance.doubleValue()>0)&&(this.balance.subtract(debitAccountCommand.getAmount(
)).doubleValue()<0)){
         throw new InsufficientBalanceException("Insufficient
Balance=>"+this.balance.doubleValue());
      } else
      AggregateLifecycle.apply(
            new AccountDebitedEvent(
                  debitAccountCommand.getId(),   debitAccountCommand.getAmount(),
                  debitAccountCommand.getCurrency()
            )
      );
   }
```

Commands: Aggregate

AccountAggregate

```
@EventSourcingHandler
  public void on(AccountDebitedEvent accountDebitedEvent){
     log.info("AccountDebitedEvent Occured");
     this.balance=this.balance.subtract(accountDebitedEvent.getAmount());
     System.out.println(this.balance);
  }
```

Commands: Aggregate

AccountAggregate

```
@CommandHandler
   public void handle(CreditAccountCommand creditAccountCommand){
      AggregateLifecycle.apply(new AccountCreditedEvent(
        creditAccountCommand.getId(),
        creditAccountCommand.getAmount(),
        creditAccountCommand.getCurrency()
      ));
   }

@EventSourcingHandler
   public void on(AccountCreditedEvent accountCreditedEvent){
      this.balance=this.balance.add(accountCreditedEvent.getAmount());
   }
}
```

Commands DTO

CreateAccountRequestDTO

```
@Data
public class CreateAccountRequestDTO {
    private BigDecimal balance;
    private String currency;
}
```

CreateAccountRequestDTO

```
@Data
public class CreditAccountRequestDTO {
    private String accountId;
    private BigDecimal amount;
    private String currency;
}
```

```
@Data
public class DebitAccountRequestDTO {
    private String accountId;
    private BigDecimal amount;
    private String currency;
}
```

Command Exceptions

InsufficientBalanceException

```
public class InsufficientBalanceException extends RuntimeException {
    public InsufficientBalanceException(String message) {
        super(message);
    }
}
```

Commands : Services

AccountCommandService

```
public interface AccountCommandService {
   CompletableFuture<String> createAccount(CreateAccountRequestDTO accountRequestDTO);
   CompletableFuture<String> debitAccount(DebitAccountRequestDTO debitAccountRequestDTO);
   CompletableFuture<String> creditAccount(CreditAccountRequestDTO creditAccountRequestDTO);
}
```

AccountCommandServiceImpl

```
@Service
public class AccountCommandServiceImpl implements AccountCommandService {
   @Autowired
   private CommandGateway commandGateway;
```

Commands : Services

AccountCommandServiceImpl

```
@Override
   public CompletableFuture<String> createAccount(CreateAccountRequestDTO carDTO) {
      return commandGateway.send(
new CreateAccountCommand(
         UUID.randomUUID().toString(),
         carDTO.getBalance(),
         carDTO.getCurrency()));
   }
```

Commands : Services

AccountCommandServiceImpl

```
@Override
public CompletableFuture<String> debitAccount(DebitAccountRequestDTO
debitAccountRequestDTO) {
    return commandGateway.send(new DebitAccountCommand(
            debitAccountRequestDTO.getAccountId(),
            debitAccountRequestDTO.getAmount(),
            debitAccountRequestDTO.getCurrency()
    ));
}
```

Commands : Services

AccountCommandServiceImpl

```
@Override
   public CompletableFuture<String> creditAccount(CreditAccountRequestDTO
creditAccountRequestDTO) {
        return commandGateway.send(new CreditAccountCommand(
                creditAccountRequestDTO.getAccountId(),
                creditAccountRequestDTO.getAmount(),
                creditAccountRequestDTO.getCurrency()
        ));
   }
}
```

Commands : RestController

AccountCommandRestController

```
@RestController
@RequestMapping(path="/commands")
public class AccountCommandRestController {

    private final AccountCommandService accountCommandService;

    public AccountCommandRestController(AccountCommandService accountCommandService) {
        this.accountCommandService = accountCommandService;
    }
```

Commands : RestController

AccountCommandRestController

```
@PostMapping(path = "/accounts/create")
   public CompletableFuture<String> createAccount(@RequestBody CreateAccountRequestDTO
accountRequestDTO){
      return accountCommandService.createAccount(accountRequestDTO);
   }

   @PutMapping(path = "/accounts/debit")
   public CompletableFuture<String> debitAccount(@RequestBody DebitAccountRequestDTO
debitAccountRequestDTO){
      return accountCommandService.debitAccount(debitAccountRequestDTO);
   }
```

Commands : RestController

AccountCommandRestController

```
 @PutMapping(path = "/accounts/credit")
   public CompletableFuture<String> creditAccount(@RequestBody CreditAccountRequestDTO
creditAccountRequestDTO){
      return accountCommandService.creditAccount(creditAccountRequestDTO);
   }

   @ExceptionHandler(InsufficientBalanceException.class)
   public ResponseEntity<String> exceptionHandler(InsufficientBalanceException exception){
      return new ResponseEntity<String>(exception.getMessage(),HttpStatus.INTERNAL_SERVER_ERROR);
   }
}
```

```
#spring.datasource.url=jdbc:h2:mem:training-bank
spring.datasource.url=jdbc:mysql://localhost:3306/training-bank
server.port=8085
#spring.h2.console.enabled=true
spring.datasource.username = root
spring.datasource.password =
spring.jpa.hibernate.ddl-auto = update
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MariaDBDialect
```

Application

AccountCommandRestController

```
@SpringBootApplication
public class CqrsAxonAdriaApplication {

   public static void main(String[] args) {
      SpringApplication.run(CqrsAxonAdriaApplication.class, args);
   }
}
```

```
//@Bean
   public Repository<AccountAggregate>
accountAggregateRepository(EntityManagerProvider emp,
ParameterResolverFactory prf, EventBus eb){
      return GenericJpaRepository.builder(AccountAggregate.class)
            .entityManagerProvider(emp)
            .parameterResolverFactory(prf)
            .eventBus(eb)
            .build();
   }
```

```
//@Bean
   public EventSourcingRepository<AccountAggregate>
eventSourcingRepository(EventStore
eventStore,ParameterResolverFactory prf){
      return
EventSourcingRepository.builder(AccountAggregate.class)
            .eventStore(eventStore)
            .parameterResolverFactory(prf)
            .build();
```

EventSourcing Service

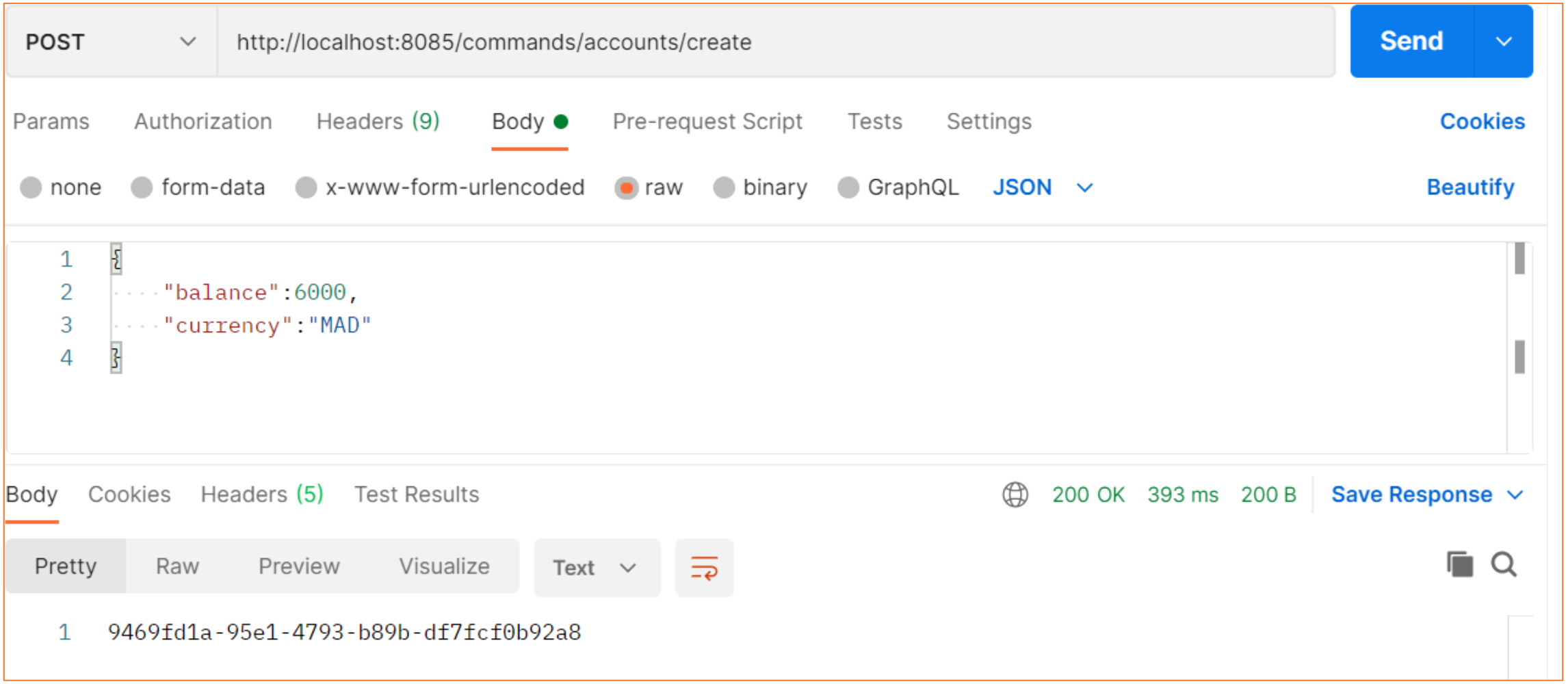
```
public interface EventSourcingService {
    DomainEventStream eventsByAccountId(String accountId);
}
```

```
@Service
public class EventSourcingServiceImpl implements EventSourcingService {
    private final EventStore eventStore;

    public EventSourcingServiceImpl(EventStore eventStore) {
        this.eventStore = eventStore;
    }

    @Override
    public DomainEventStream eventsByAccountId(String accountId) {
        DomainEventStream domainEventStream = eventStore.readEvents(accountId);
        return domainEventStream;
    }
}
```

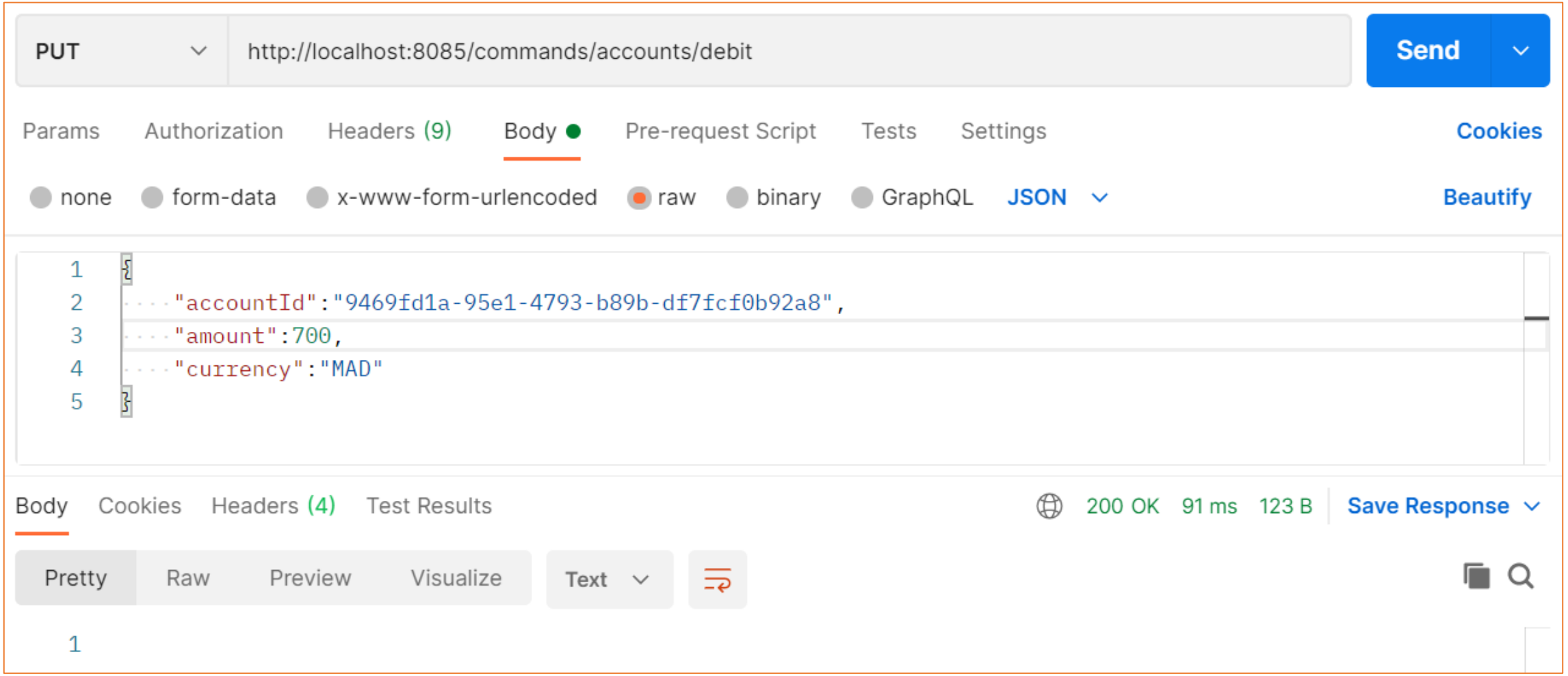
EventSourcing Rest Controller

```
@RestController
@RequestMapping(path = "/eventSourcing")
public class AccountEventSourcingRestController {
    private final EventSourcingService accountQueryService;

    public AccountEventSourcingRestController(EventSourcingService accountQueryService) {
        this.accountQueryService = accountQueryService;
    }

    @GetMapping(path = "/accountEvents/{id}")
    public Stream eventsByAccountId(@PathVariable String id){
        return accountQueryService.eventsByAccountId(id).asStream();
    }
}
```

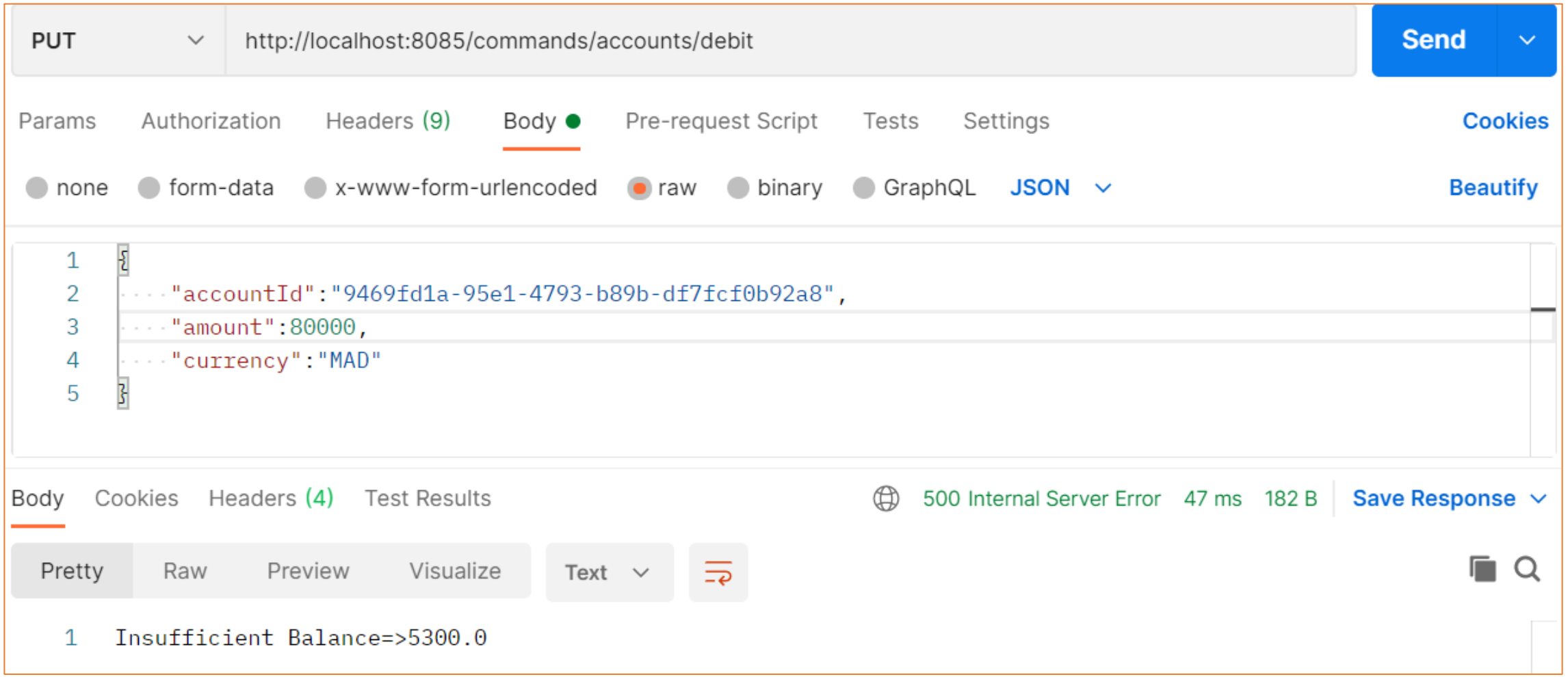
Test : Create Account Comand



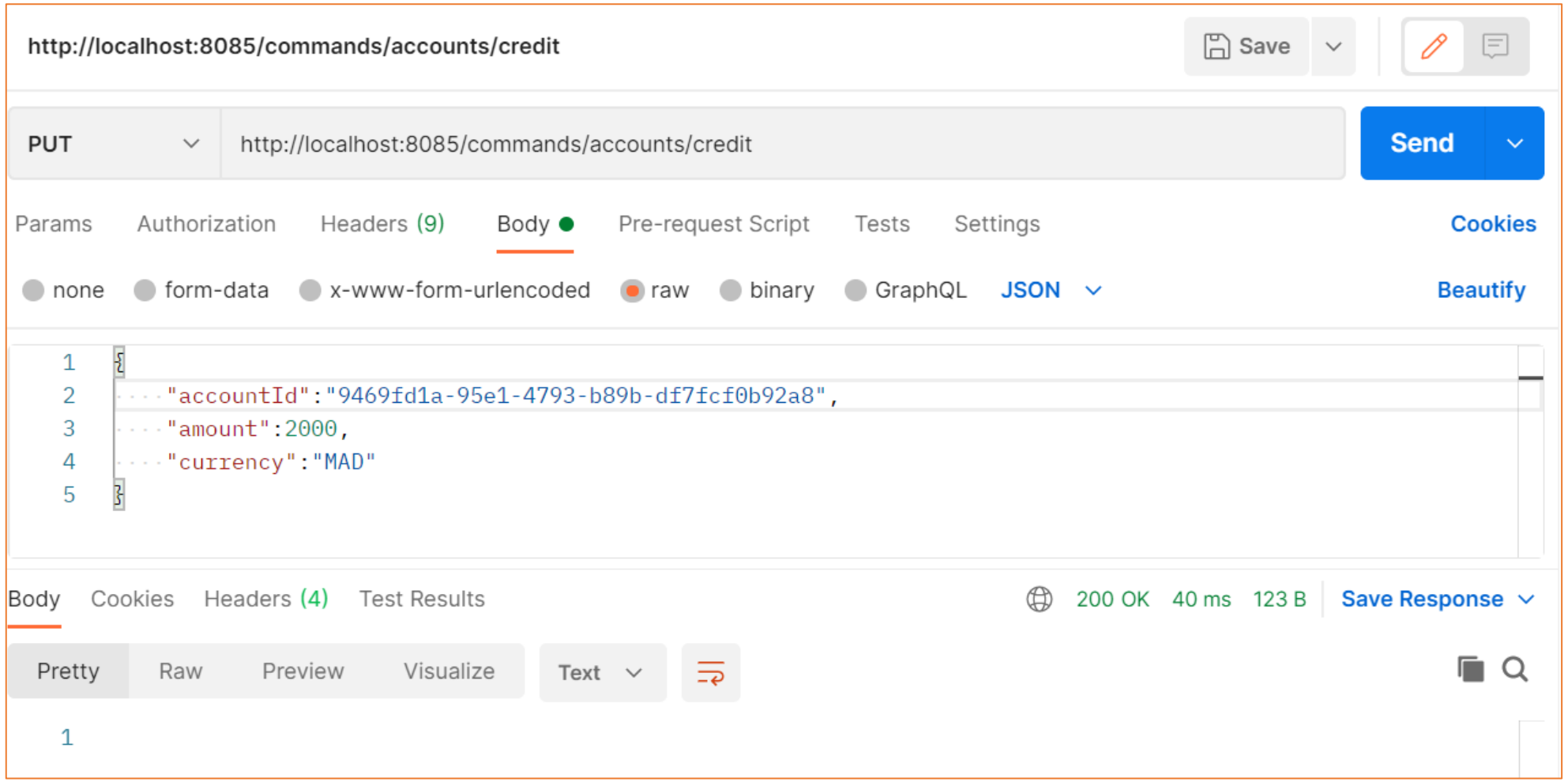
Test : Debit Account Comand



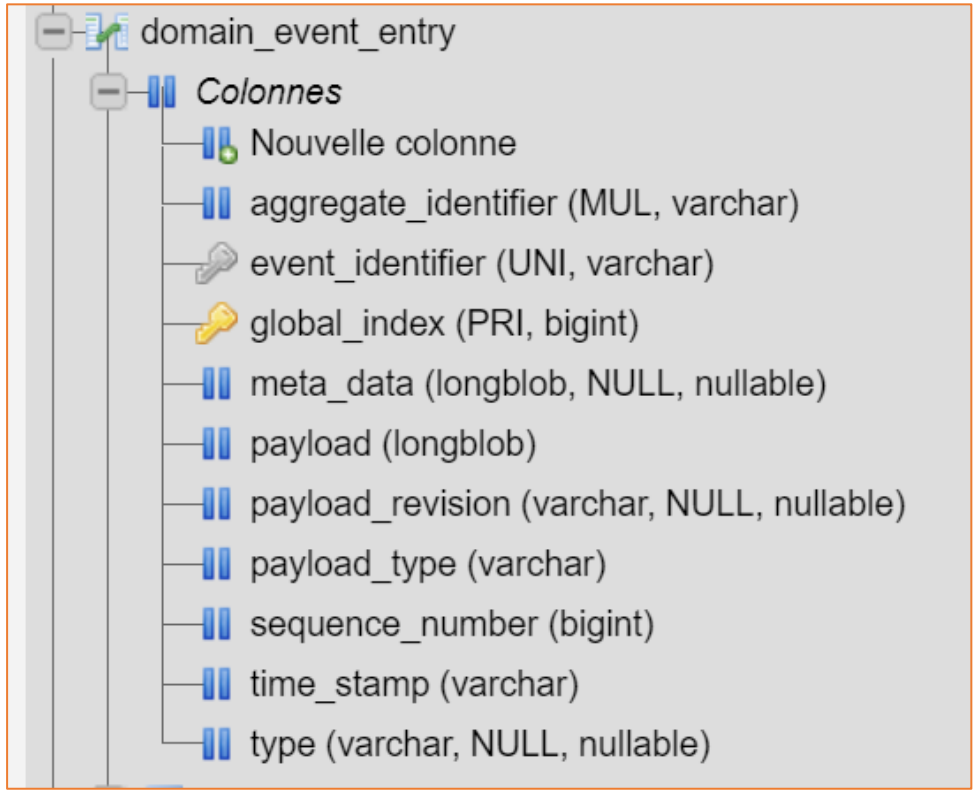
Test : Debit Account Comand



Test : Credit Account Comand

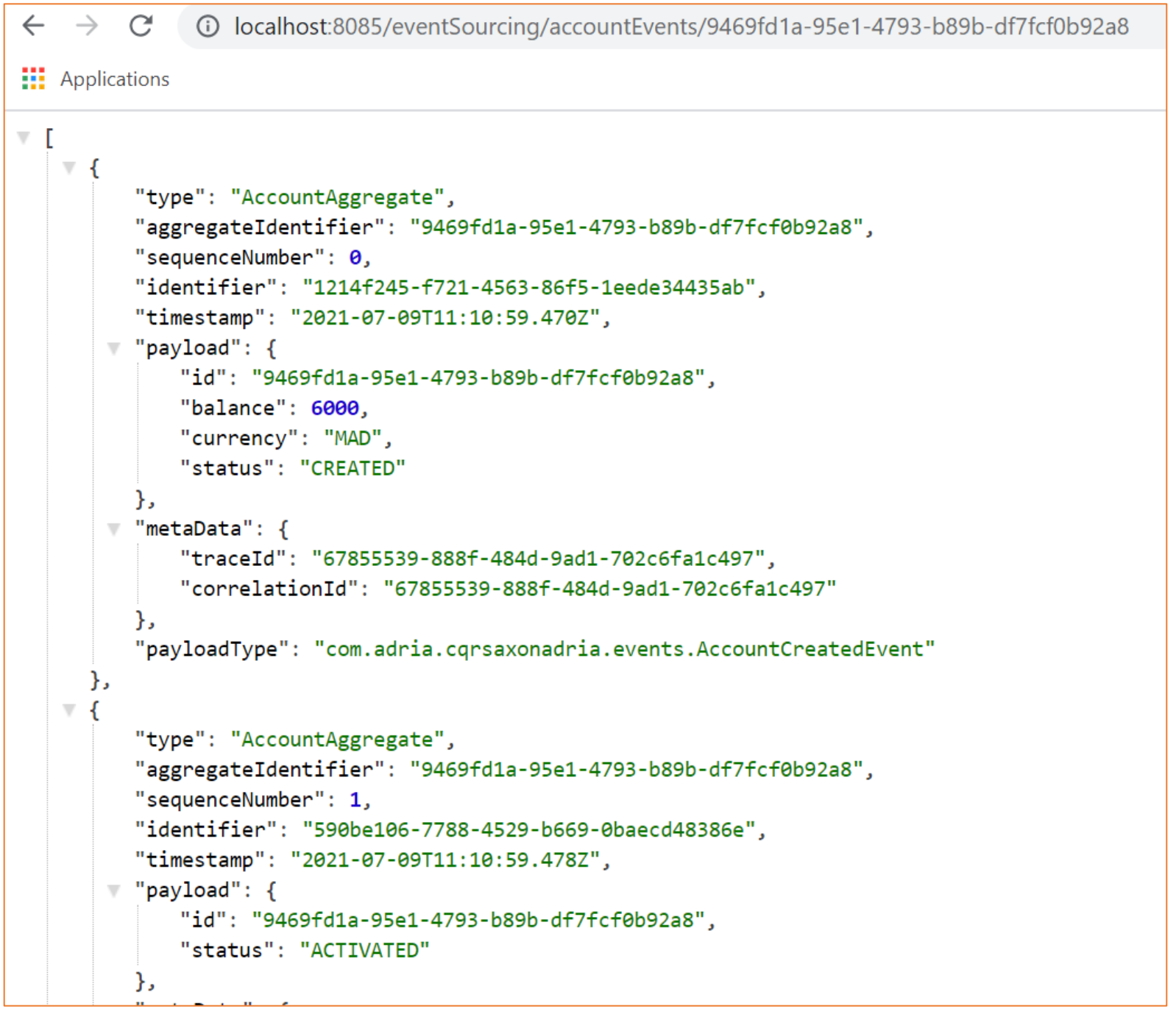


EventStore Data Base

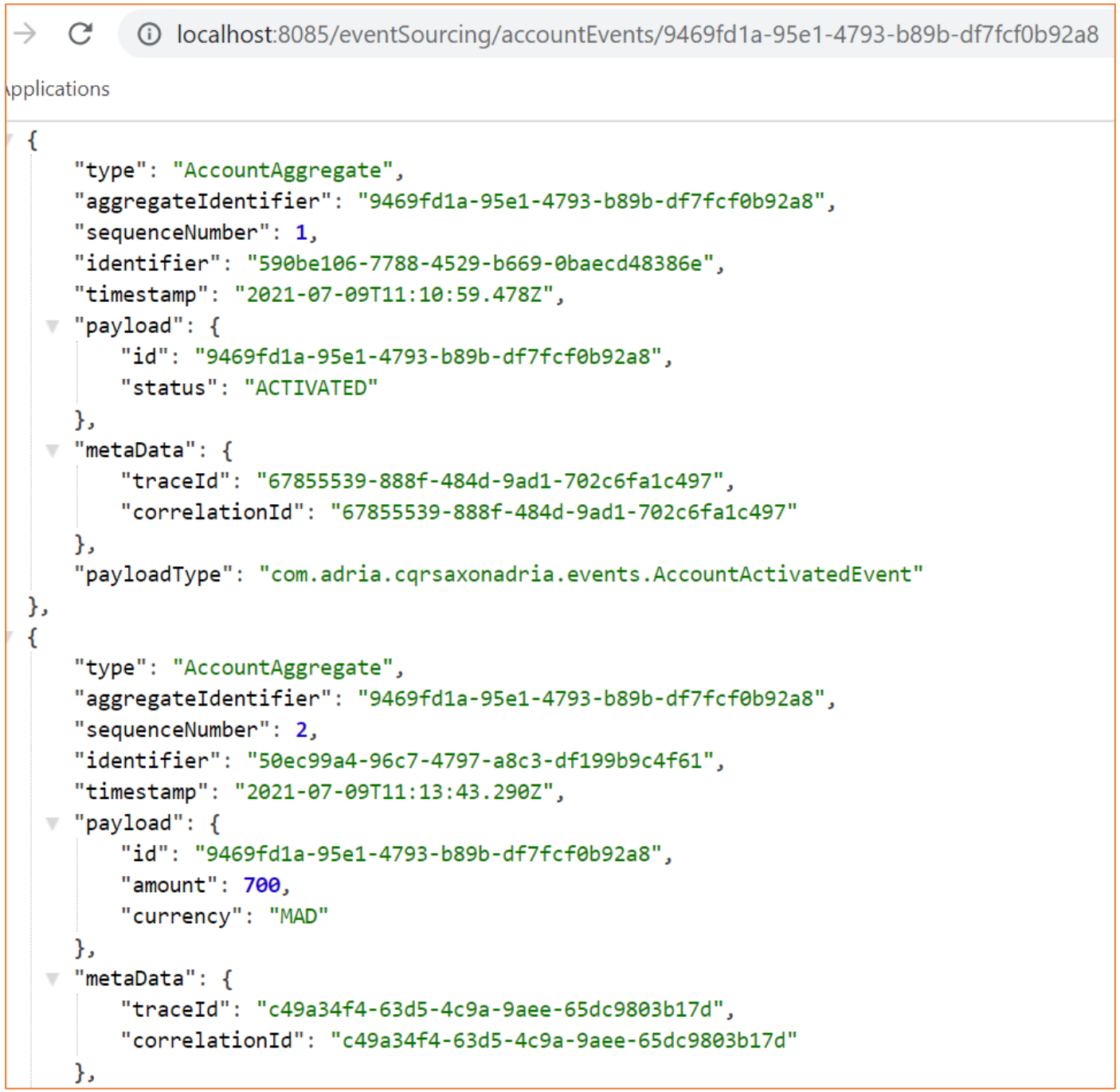


global_index	event_identifier	meta_data	payload	payload_revision	payload_type	time_stamp	aggregate_identifier	sequence_number	type
1	37f71974-bee2-4319-9cfd-e6b944369366	[BLOB - 213 o]	[BLOB - 236 o]	*NULL*	com.adria.cqrsaxonadria.events.AccountCreatedEvent	2021-07-06T11:45:11.947Z	f69ac3ab-f399-4328-a5ba-46fbdbb874c5	0	AccountAggregate
2	af9995ad-b33d-4dc3-b8cf-9d78953678a0	[BLOB - 213 o]	[BLOB - 195 o]	*NULL*	com.adria.cqrsaxonadria.events.AccountActivatedEve...	2021-07-06T11:45:11.951Z	f69ac3ab-f399-4328-a5ba-46fbdbb874c5	1	AccountAggregate
3	43f4607a-4f60-4cb8-9108-28905318fade	[BLOB - 213 o]	[BLOB - 236 o]	*NULL*	com.adria.cqrsaxonadria.events.AccountCreatedEvent	2021-07-06T11:45:47.374Z	4af49171-b8ce-4fce-814f-4529d34afc1d	0	AccountAggregate
4	1370744b-4e6b-4e5d-8ce5-9106b3048323	[BLOB - 213 o]	[BLOB - 195 o]	*NULL*	com.adria.cqrsaxonadria.events.AccountActivatedEve...	2021-07-06T11:45:47.380Z	4af49171-b8ce-4fce-814f-4529d34afc1d	1	AccountAggregate
5	021a2217-1a4d-4a69-a528-3fe266508f18	[BLOB - 213 o]	[BLOB - 209 o]	*NULL*	com.adria.cqrsaxonadria.events.AccountDebitedEvent	2021-07-06T11:48:06.456Z	4af49171-b8ce-4fce-814f-4529d34afc1d	2	AccountAggregate
6	c0b4d42b-1ab0-433b-9169-4d684923cd6c	[BLOB - 213 o]	[BLOB - 212 o]	*NULL*	com.adria.cqrsaxonadria.events.AccountCreditedEven...	2021-07-06T11:50:07.909Z	4af49171-b8ce-4fce-814f-4529d34afc1d	3	AccountAggregate
7	c9fbcce1-e2b3-4783-a2fb-304857de93c5	[BLOB - 213 o]	[BLOB - 212 o]	*NULL*	com.adria.cqrsaxonadria.events.AccountCreditedEven...	2021-07-06T12:03:41.530Z	4af49171-b8ce-4fce-814f-4529d34afc1d	4	AccountAggregate
8	8209b325-8059-4894-8dd9-9014a138a6cc	[BLOB - 213 o]	[BLOB - 212 o]	*NULL*	com.adria.cqrsaxonadria.events.AccountCreditedEven...	2021-07-06T12:03:54.810Z	4af49171-b8ce-4fce-814f-4529d34afc1d	5	AccountAggregate
9	2fe7001c-dc3d-4a4a-a815-788c0680c2cd	[BLOB - 213 o]	[BLOB - 212 o]	*NULL*	com.adria.cqrsaxonadria.events.AccountCreditedEven...	2021-07-06T19:33:20.855Z	4af49171-b8ce-4fce-814f-4529d34afc1d	6	AccountAggregate
10	0eba7fc0-f325-4463-8e0d-58b80659f94b	[BLOB - 213 o]	[BLOB - 209 o]	*NULL*	com.adria.cqrsaxonadria.events.AccountDebitedEvent	2021-07-06T19:33:33.835Z	4af49171-b8ce-4fce-814f-4529d34afc1d	7	AccountAggregate
11	8f455a83-dec5-4249-8d6a-ba6b89fcb6ea	[BLOB - 213 o]	[BLOB - 209 o]	*NULL*	com.adria.cqrsaxonadria.events.AccountDebitedEvent	2021-07-06T19:36:07.686Z	4af49171-b8ce-4fce-814f-4529d34afc1d	8	AccountAggregate
12	636702df-4b55-469d-9d47-ab7548dc1c92	[BLOB - 213 o]	[BLOB - 212 o]	*NULL*	com.adria.cqrsaxonadria.events.AccountCreditedEven...	2021-07-06T19:36:09.959Z	4af49171-b8ce-4fce-814f-4529d34afc1d	9	AccountAggregate
13	1214f245-f721-4563-86f5-1eede34435ab	[BLOB - 213 o]	[BLOB - 236 o]	*NULL*	com.adria.cqrsaxonadria.events.AccountCreatedEvent	2021-07-09T11:10:59.470Z	9469fd1a-95e1-4793-b89b-df7fcf0b92a8	0	AccountAggregate
14	590be106-7788-4529-b669-0baecd48386e	[BLOB - 213 o]	[BLOB - 195 o]	*NULL*	com.adria.cqrsaxonadria.events.AccountActivatedEve...	2021-07-09T11:10:59.478Z	9469fd1a-95e1-4793-b89b-df7fcf0b92a8	1	AccountAggregate
15	50ec99a4-96c7-4797-a8c3-df199b9c4f61	[BLOB - 213 o]	[BLOB - 209 o]	*NULL*	com.adria.cqrsaxonadria.events.AccountDebitedEvent	2021-07-09T11:13:43.290Z	9469fd1a-95e1-4793-b89b-df7fcf0b92a8	2	AccountAggregate
16	477e84c3-15b9-412c-9137-e339e812c607	[BLOB - 213 o]	[BLOB - 212 o]	*NULL*	com.adria.cqrsaxonadria.events.AccountCreditedEven...	2021-07-09T11:22:17.383Z	4af49171-b8ce-4fce-814f-4529d34afc1d	10	AccountAggregate
17	c0702fe7-d5f9-4050-8b99-8444ad714d50	[BLOB - 213 o]	[BLOB - 212 o]	*NULL*	com.adria.cqrsaxonadria.events.AccountCreditedEven...	2021-07-09T11:22:33.514Z	9469fd1a-95e1-4793-b89b-df7fcf0b92a8	3	AccountAggregate

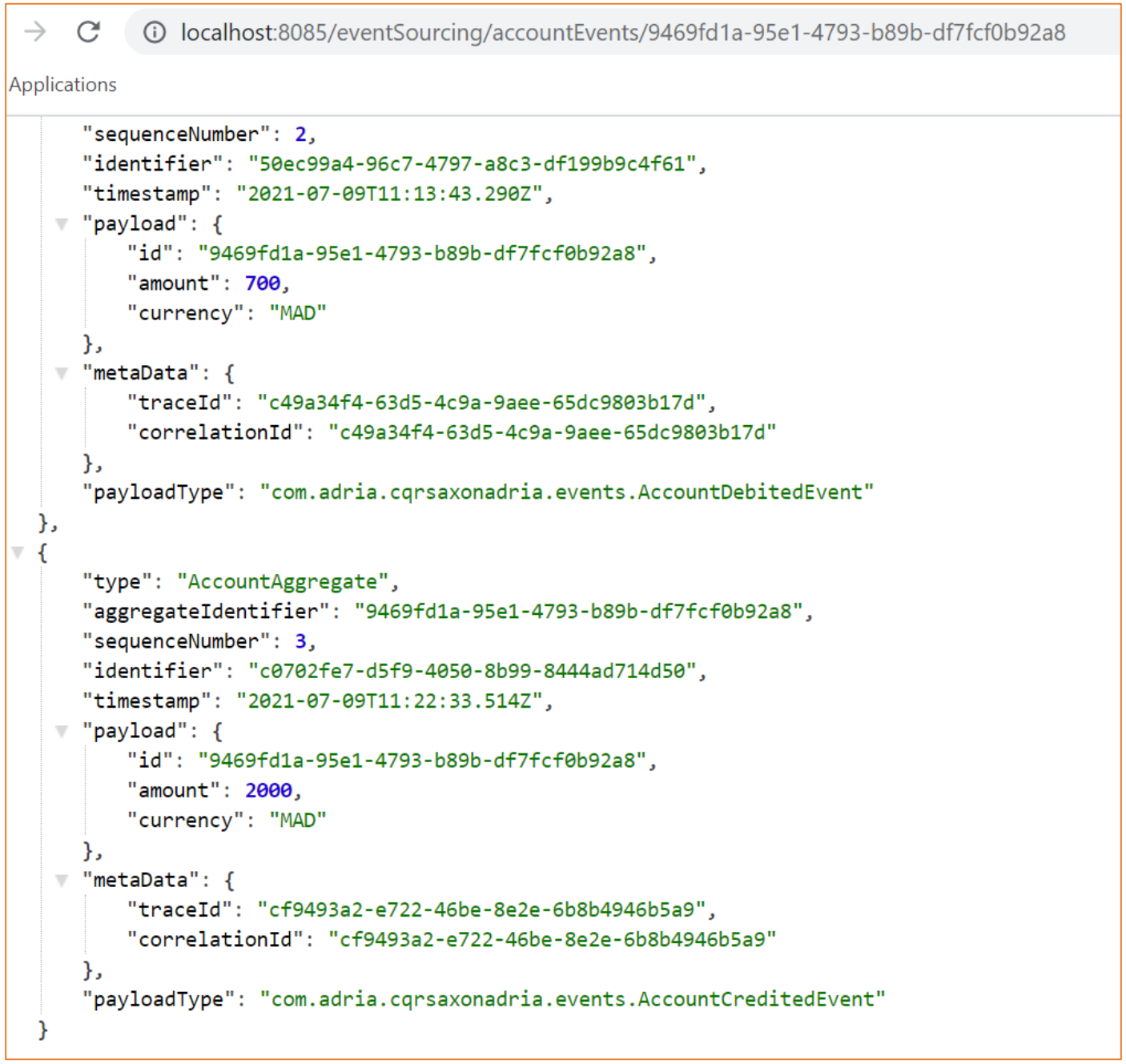
Events


localhost:8085/eventSourcing/accountEvents/9469fd1a-95e1-4793-b89b-df7fcf0b92a8
Applications
[
{
"type": "AccountAggregate",
"aggregateIdentifier": "9469fd1a-95e1-4793-b89b-df7fcf0b92a8",
"sequenceNumber": 0,
"identifier": "1214f245-f721-4563-86f5-1eede34435ab",
"timestamp": "2021-07-09T11:10:59.470Z",
"payload": {
"id": "9469fd1a-95e1-4793-b89b-df7fcf0b92a8",
"balance": 6000,
"currency": "MAD",
"status": "CREATED"
},
"metaData": {
"traceId": "67855539-888f-484d-9ad1-702c6fa1c497",
"correlationId": "67855539-888f-484d-9ad1-702c6fa1c497"
},
"payloadType": "com.adria.cqrsaxonadria.events.AccountCreatedEvent"
},
{
"type": "AccountAggregate",
"aggregateIdentifier": "9469fd1a-95e1-4793-b89b-df7fcf0b92a8",
"sequenceNumber": 1,
"identifier": "590be106-7788-4529-b669-0baecd48386e",
"timestamp": "2021-07-09T11:10:59.478Z",
"payload": {
"id": "9469fd1a-95e1-4793-b89b-df7fcf0b92a8",
"status": "ACTIVATED"
},

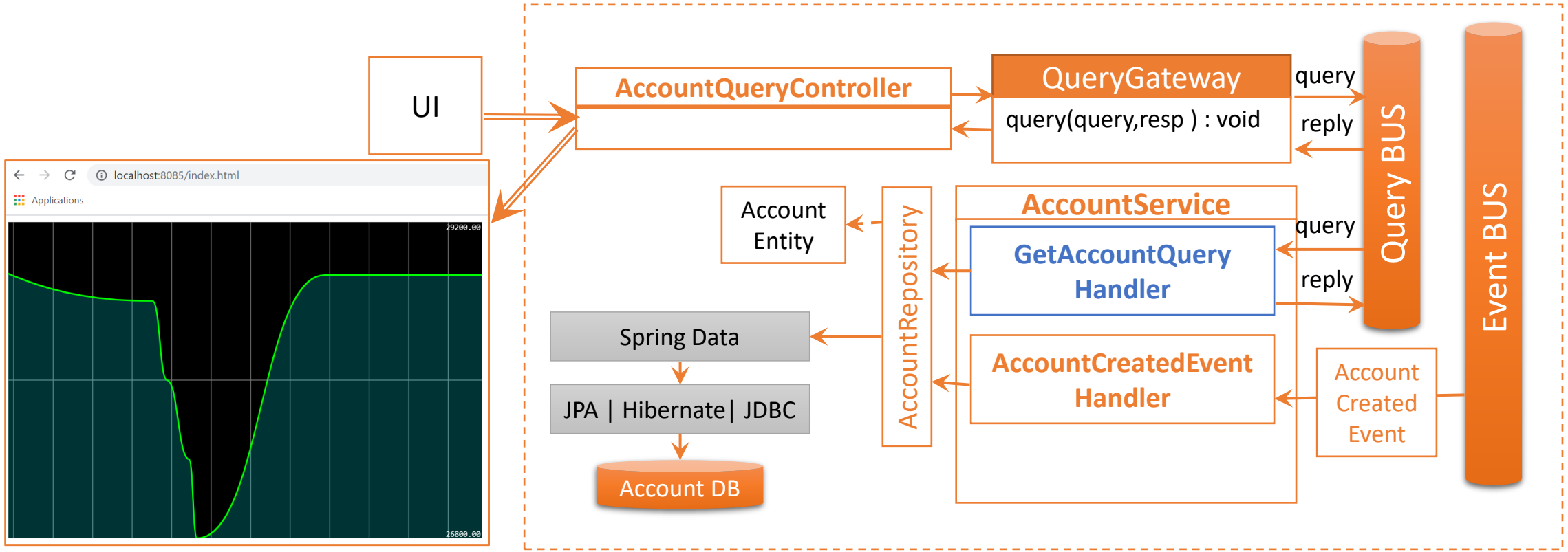
Events


localhost:8085/eventSourcing/accountEvents/9469fd1a-95e1-4793-b89b-df7fcf0b92a8
pplications
{
"type": "AccountAggregate",
"aggregateIdentifier": "9469fd1a-95e1-4793-b89b-df7fcf0b92a8",
"sequenceNumber": 1,
"identifier": "590be106-7788-4529-b669-0baecd48386e",
"timestamp": "2021-07-09T11:10:59.478Z",
"payload": {
"id": "9469fd1a-95e1-4793-b89b-df7fcf0b92a8",
"status": "ACTIVATED"
},
"metaData": {
"traceId": "67855539-888f-484d-9ad1-702c6fa1c497",
"correlationId": "67855539-888f-484d-9ad1-702c6fa1c497"
},
"payloadType": "com.adria.cqrsaxonadria.events.AccountActivatedEvent"
},
{
"type": "AccountAggregate",
"aggregateIdentifier": "9469fd1a-95e1-4793-b89b-df7fcf0b92a8",
"sequenceNumber": 2,
"identifier": "50ec99a4-96c7-4797-a8c3-df199b9c4f61",
"timestamp": "2021-07-09T11:13:43.290Z",
"payload": {
"id": "9469fd1a-95e1-4793-b89b-df7fcf0b92a8",
"amount": 700,
"currency": "MAD"
},
"metaData": {
"traceId": "c49a34f4-63d5-4c9a-9aee-65dc9803b17d",
"correlationId": "c49a34f4-63d5-4c9a-9aee-65dc9803b17d"
},

Events


localhost:8085/eventSourcing/accountEvents/9469fd1a-95e1-4793-b89b-df7fcf0b92a8
Applications
"sequenceNumber": 2,
"identifier": "50ec99a4-96c7-4797-a8c3-df199b9c4f61",
"timestamp": "2021-07-09T11:13:43.290Z",
"payload": {
"id": "9469fd1a-95e1-4793-b89b-df7fcf0b92a8",
"amount": 700,
"currency": "MAD"
},
"metaData": {
"traceId": "c49a34f4-63d5-4c9a-9aee-65dc9803b17d",
"correlationId": "c49a34f4-63d5-4c9a-9aee-65dc9803b17d"
},
"payloadType": "com.adria.cqrsaxonadria.events.AccountDebitedEvent"
},
{
"type": "AccountAggregate",
"aggregateIdentifier": "9469fd1a-95e1-4793-b89b-df7fcf0b92a8",
"sequenceNumber": 3,
"identifier": "c0702fe7-d5f9-4050-8b99-8444ad714d50",
"timestamp": "2021-07-09T11:22:33.514Z",
"payload": {
"id": "9469fd1a-95e1-4793-b89b-df7fcf0b92a8",
"amount": 2000,
"currency": "MAD"
},
"metaData": {
"traceId": "cf9493a2-e722-46be-8e2e-6b8b4946b5a9",
"correlationId": "cf9493a2-e722-46be-8e2e-6b8b4946b5a9"
},
"payloadType": "com.adria.cqrsaxonadria.events.AccountCreditedEvent"
}

QYERY Side


UI
localhost:8085/index.html
Applications
29200.00
26800.00
AccountQueryController
QueryGateway
query(query,resp) : void
query
reply
Query BUS
Event BUS
Account Entity
AccountRepository
AccountService
GetAccountQuery Handler
query
reply
Spring Data
JPA | Hibernate| JDBC
Account DB
AccountCreatedEvent Handler
Account Created Event

Query Side : JPA Entities

BankAccount

```
@Entity
@Data @AllArgsConstructor
@NoArgsConstructor
public class BankAccount {
   @Id
   private String id;
   private BigDecimal balance;
   @Enumerated(EnumType.STRING)
   private AccountStatus status;
}
```

OperationType

```
public enum OperationType {
   DEBIT, CREDIT
}
```

BankAccount

```
@Entity
@Data @AllArgsConstructor
@NoArgsConstructor
public class AccountOperation {
   @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;
   private Date operationDate;
   private BigDecimal amount;
   @Enumerated(EnumType.STRING)
   private OperationType type;
   @ManyToOne
   private BankAccount bankAccount;
}
```

Query Side : JPA Repositories

BankAccountRepository

```
public interface BankAccountRepository extends JpaRepository<BankAccount,String> {


}
```

AccountOperationRepository

```
public interface AccountOperationRepository extends JpaRepository<AccountOperation,Long> {
   List<AccountOperation> findByBankAccountId(String accountId);
}
```

Query Side : Queries DTO

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

```java
public class GetAllAccountsRequestDTO { }
```

```java
@Data @AllArgsConstructor @NoArgsConstructor
public class GetAccountOperationsQueryDTO {
    private String accountId;
}
```

```java
@Data @AllArgsConstructor @NoArgsConstructor
public class GetAccountOperationsQueryDTO {
    private String accountId;
}
```

```java
@Data @AllArgsConstructor @NoArgsConstructor
public class BankAccountResponseDTO {
    private String id;
    private BigDecimal balance;
    private AccountStatus status;
}
```

```java
@Data @AllArgsConstructor @NoArgsConstructor
public class AccountOperationResponseDTO {
    private Long id;
    private Date operationDate;
    private BigDecimal amount;
    private OperationType type;
}
```

Query Side : Mappers

BankAccountMapper

```
@Mapper(componentModel = "spring")
public interface BankAccountMapper {

    BankAccountResponseDTO bankAccountToBankAccountDTO(BankAccount bankAccount);

    AccountOperationResponseDTO accountOperationToAccountOperationDTO(AccountOperation accOp);

}
```

Query Side : QueryService

AccountQueryService

```
@Service
public class AccountQueryService {
    private final BankAccountRepository accountRepository;
    private final AccountOperationRepository accountOperationRepository;
    private final BankAccountMapper bankAccountMapper;

    public AccountQueryService(BankAccountRepository accountRepository, AccountOperationRepository
accountOperationRepository, BankAccountMapper bankAccountMapper) {
        this.accountRepository = accountRepository;
        this.accountOperationRepository = accountOperationRepository;
        this.bankAccountMapper = bankAccountMapper;
    }
```

Query Side : QueryService

AccountQueryService

```
@EventHandler
@Transactional
public void on(AccountCreatedEvent accountCreatedEvent){
    BankAccount bankAccount=new BankAccount();
    bankAccount.setId(accountCreatedEvent.getId());
    bankAccount.setBalance(accountCreatedEvent.getBalance());
    bankAccount.setStatus(accountCreatedEvent.getStatus());
    accountRepository.save(bankAccount);
}
@EventHandler
@Transactional
public void on(AccountActivatedEvent accountActivatedEvent){
    BankAccount bankAccount=accountRepository.findById(accountActivatedEvent.getId()).get();
    bankAccount.setStatus(accountActivatedEvent.getStatus());
    accountRepository.save(bankAccount);
}
```

Query Side : QueryService

AccountQueryService

```
    @EventHandler
    @Transactional
    public void on(AccountDebitedEvent accountDebitedEvent){
        BankAccount bankAccount=accountRepository.findById(accountDebitedEvent.getId()).get();
        bankAccount.setBalance(bankAccount.getBalance().subtract(accountDebitedEvent.getAmount()));
        BankAccount savedAccount =accountRepository.save(bankAccount);
        AccountOperation accountOperation=new AccountOperation();
        accountOperation.setOperationDate(new Date());
        accountOperation.setAmount(accountDebitedEvent.getAmount());
        accountOperation.setBankAccount(savedAccount);
        accountOperation.setType(OperationType.DEBIT);
        accountOperationRepository.save(accountOperation);
        queryUpdateEmitter.emit(m->(
                (GetAccountQueryDTO)m.getPayload()).getId().equals(accountDebitedEvent.getId()),
        bankAccountMapper.bankAccountToBankAccountDTO(bankAccount)
);
    }
```

Query Side : QueryService

AccountQueryService

```
@EventHandler
 @Transactional
 public void on(AccountCreditedEvent accountCreditedEvent){
    BankAccount bankAccount=accountRepository.findById(accountCreditedEvent.getId()).get();
    bankAccount.setBalance(bankAccount.getBalance().add(accountCreditedEvent.getAmount()));
    BankAccount savedAccount = accountRepository.save(bankAccount);
    AccountOperation accountOperation=new AccountOperation();
    accountOperation.setOperationDate(new Date());
    accountOperation.setAmount(accountCreditedEvent.getAmount());
    accountOperation.setBankAccount(savedAccount);
    accountOperation.setType(OperationType.CREDIT);
    accountOperationRepository.save(accountOperation);
    queryUpdateEmitter.emit(m->(
    (GetAccountQueryDTO)m.getPayload()).getId().equals(accountCreditedEvent.getId()),
    bankAccountMapper.bankAccountToBankAccountDTO(bankAccount) );
 }
```

Query Side : QueryService

AccountQueryService

```
 @QueryHandler
public BankAccountResponseDTO on(GetAccountQueryDTO accountQuery) {
   BankAccount bankAccount = accountRepository.findById(accountQuery.getId()).get();
   return bankAccountMapper.bankAccountToBankAccountDTO(bankAccount);
}

@QueryHandler
public List<BankAccountResponseDTO> on(GetAllAccountsRequestDTO accountsRequest) {
   List<BankAccount> bankAccountList = accountRepository.findAll();
   return bankAccountList.stream().map((acc->
     bankAccountMapper.bankAccountToBankAccountDTO(acc))).collect(Collectors.toList());
}
```

Query Side : QueryService

AccountQueryService

```
@QueryHandler
public List<AccountOperationResponseDTO> on(GetAccountOperationsQueryDTO
getAccountOperationsQueryDTO)  {
   List<AccountOperation> accountOperations =
accountOperationRepository.findByBankAccountId(getAccountOperationsQueryDTO.getAccountId());
   return accountOperations.stream().map(op->
        bankAccountMapper.accountOperationToAccountOperationDTO(op)).collect(Collectors.toList());
}
```

Query Side : Query Rest Controller

AccountQueryRestController

```
@RestController
@RequestMapping(path = "/query")
public class AccountQueryRestController {

   private QueryGateway queryGateway;

   public AccountQueryRestController(QueryGateway queryGateway) {
      this.queryGateway = queryGateway;
   }

@GetMapping(path = "/accounts/{id}")
public BankAccountResponseDTO getBanAccount(@PathVariable String id){
   GetAccountQueryDTO queryDTO=new GetAccountQueryDTO();
   queryDTO.setId(id);
   return queryGateway.query(queryDTO, BankAccountResponseDTO.class).join();
}
```

Query Side : QueryService

AccountQueryRestController

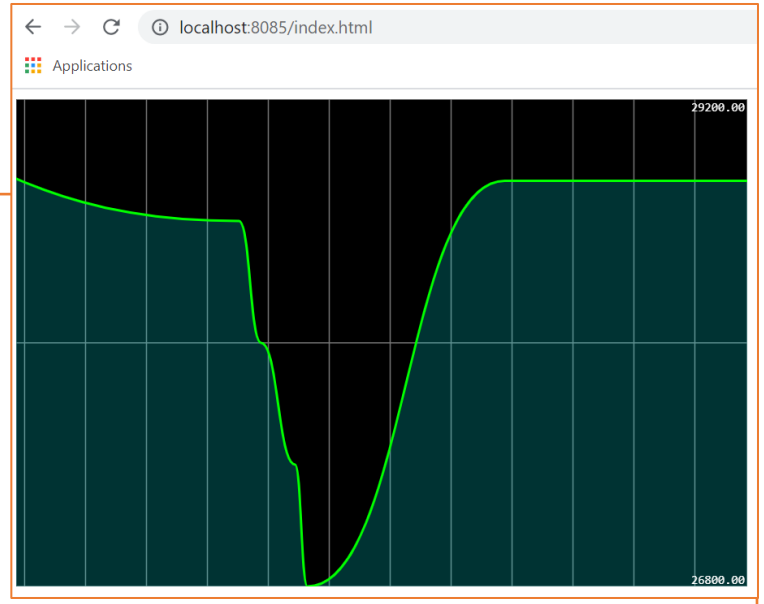
```
@GetMapping(path = "/accounts")
public List<BankAccountResponseDTO> getAll(){
   return queryGateway.query(new GetAllAccountsRequestDTO(),
         ResponseTypes.multipleInstancesOf(BankAccountResponseDTO.class)).join();
}

@GetMapping(path = "/accountOperations/{accountId}")
public List<AccountOperationResponseDTO> accountOperationList(@PathVariable String accountId){
   return queryGateway.query(new
GetAccountOperationsQueryDTO(accountId),ResponseTypes.multipleInstancesOf(AccountOperationResponseDT
O.class)).join();
}
```

Query Side : QueryService

AccountQueryRestController

```
@GetMapping(value = "/{accountId}/watch",produces = MediaType.TEXT_EVENT_STREAM_VALUE)
public Flux<AccountOperationResponseDTO> watch(@PathVariable String accountId){

@GetMapping(value = "/{accountId}/watch",produces = MediaType.TEXT_EVENT_STREAM_VALUE)
public Flux<BankAccountResponseDTO> watch(@PathVariable String accountId){

   SubscriptionQueryResult<BankAccountResponseDTO,BankAccountResponseDTO> result=
         queryGateway.subscriptionQuery(
               new GetAccountQueryDTO(accountId),
               ResponseTypes.instanceOf(BankAccountResponseDTO.class),
               ResponseTypes.instanceOf(BankAccountResponseDTO.class)
               );
   return result.initialResult().concatWith(result.updates());
  // return result.initialResult().flatMapMany(Flux::fromIterable).concatWith(result.updates());
}
}
```

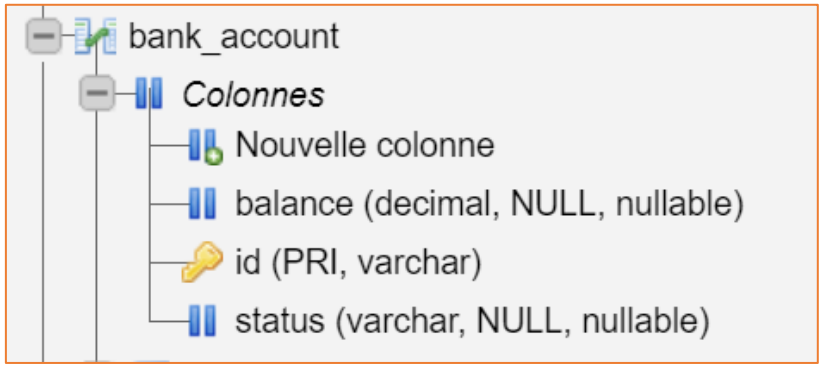
Query Side : QueryService



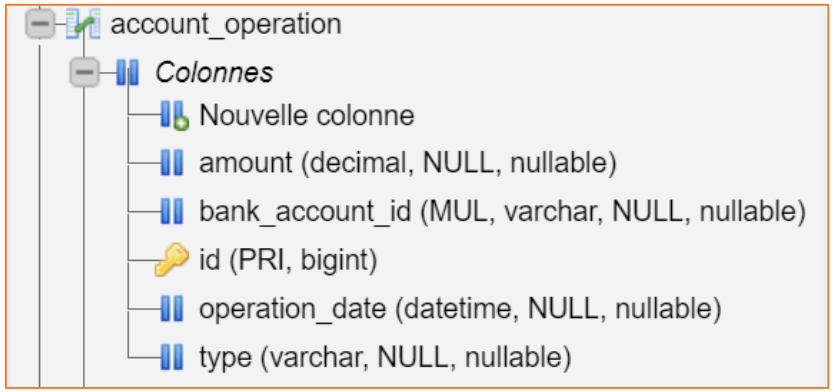
AccountQueryRestController

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">   <title>Analytics</title>
   <script src="https://cdnjs.cloudflare.com/ajax/libs/smoothie/1.34.0/smoothie.min.js"></script>
</head>
<body>
<canvas id="chart2" width="600" height="400"></canvas>
<script>
   courbe=new TimeSeries();
   var smoothieChart = new SmoothieChart({tooltip: true});
   smoothieChart.streamTo(document.getElementById("chart2"), 500);
   smoothieChart.addTimeSeries(courbe, {strokeStyle : 'rgba(0, 255, 0, 1)', fillStyle : 'rgba(0, 255, 255, 0.2)', lineWidth : 2});
   var stockEventSource= new EventSource("query/4af49171-b8ce-4fce-814f-4529d34afc1d/watch");
   stockEventSource.addEventListener("message", function (event) {
      var val=JSON.parse(event.data);
      courbe.append(new Date().getTime(),val.balance);
   }, false);
</script>
</body></html></body></html>
```

Query Side : Query Data base



id	balance	status
4af49171-b8ce-4fce-814f-4529d34afc1d	16300.00	ACTIVATED
9469fd1a-95e1-4793-b89b-df7fcf0b92a8	7300.00	ACTIVATED
f69ac3ab-f399-4328-a5ba-46fbdbb874c5	9000.00	CREATED



id	amount	operation_date	type	bank_account_id
3	600.00	2021-07-06 20:36:07	DEBIT	4af49171-b8ce-4fce-814f-4529d34afc1d
4	2000.00	2021-07-06 20:36:09	CREDIT	4af49171-b8ce-4fce-814f-4529d34afc1d
5	700.00	2021-07-09 12:13:43	DEBIT	9469fd1a-95e1-4793-b89b-df7fcf0b92a8
6	2000.00	2021-07-09 12:22:17	CREDIT	4af49171-b8ce-4fce-814f-4529d34afc1d
7	2000.00	2021-07-09 12:22:33	CREDIT	9469fd1a-95e1-4793-b89b-df7fcf0b92a8

Query Side : Test

localhost:8085/query/accounts

Applications


```
[
  {
    "id": "4af49171-b8ce-4fce-814f-4529d34afc1d",
    "balance": 16300,
    "status": "ACTIVATED"
  },
  {
    "id": "9469fd1a-95e1-4793-b89b-df7fcf0b92a8",
    "balance": 7300,
    "status": "ACTIVATED"
  },
  {
    "id": "f69ac3ab-f399-4328-a5ba-46fbdbb874c5",
    "balance": 9000,
    "status": "CREATED"
  }
]
```


localhost:8085/query/accounts/9469fd1a-95e1-4793-b89b-df7fcf0b92a8

Applications


```
{
  "id": "9469fd1a-95e1-4793-b89b-df7fcf0b92a8",
  "balance": 7300,
  "status": "ACTIVATED"
}
```


localhost:8085/query/accountOperations/9469fd1a-95e1-4793-b89b-df7fcf0b92a8

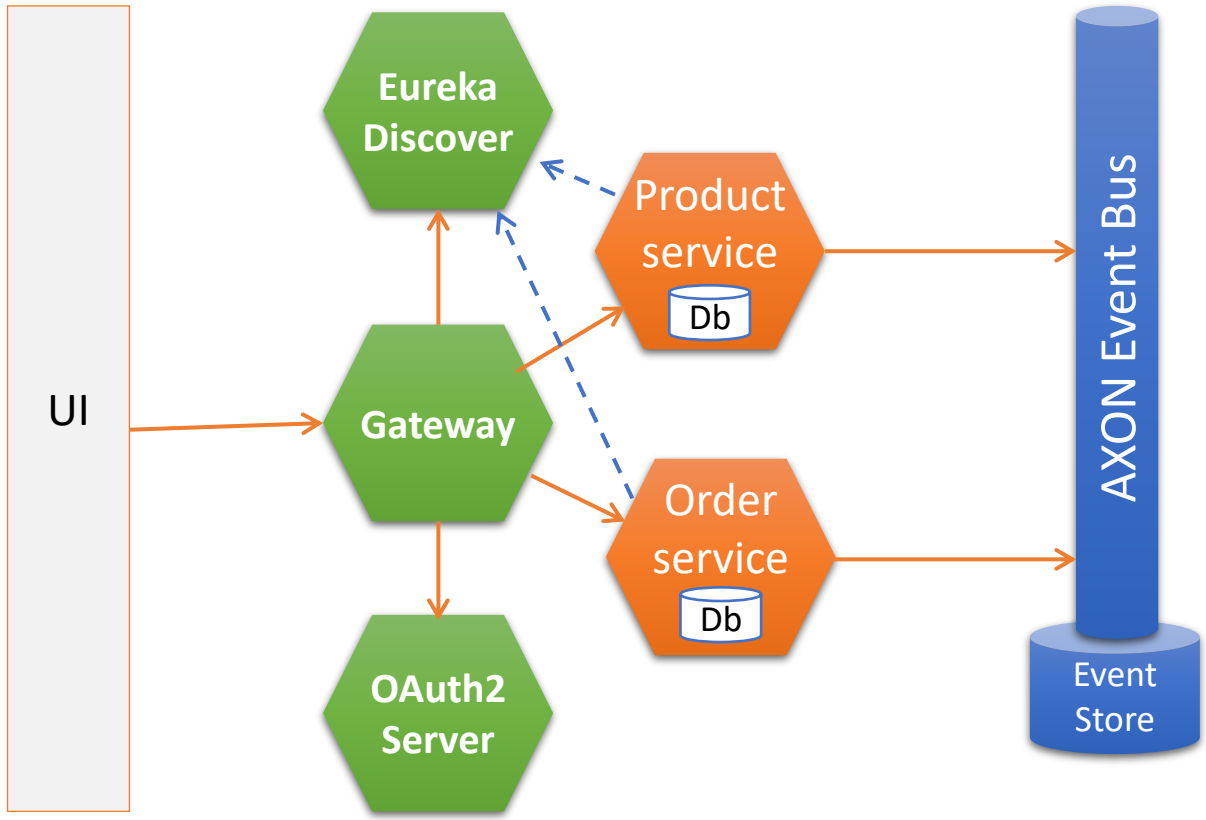
Applications


```
[
  {
    "id": 5,
    "operationDate": "2021-07-09T11:13:43.000+00:00",
    "amount": 700,
    "type": "DEBIT"
  },
  {
    "id": 7,
    "operationDate": "2021-07-09T11:22:33.000+00:00",
    "amount": 2000,
    "type": "CREDIT"
  }
]
```

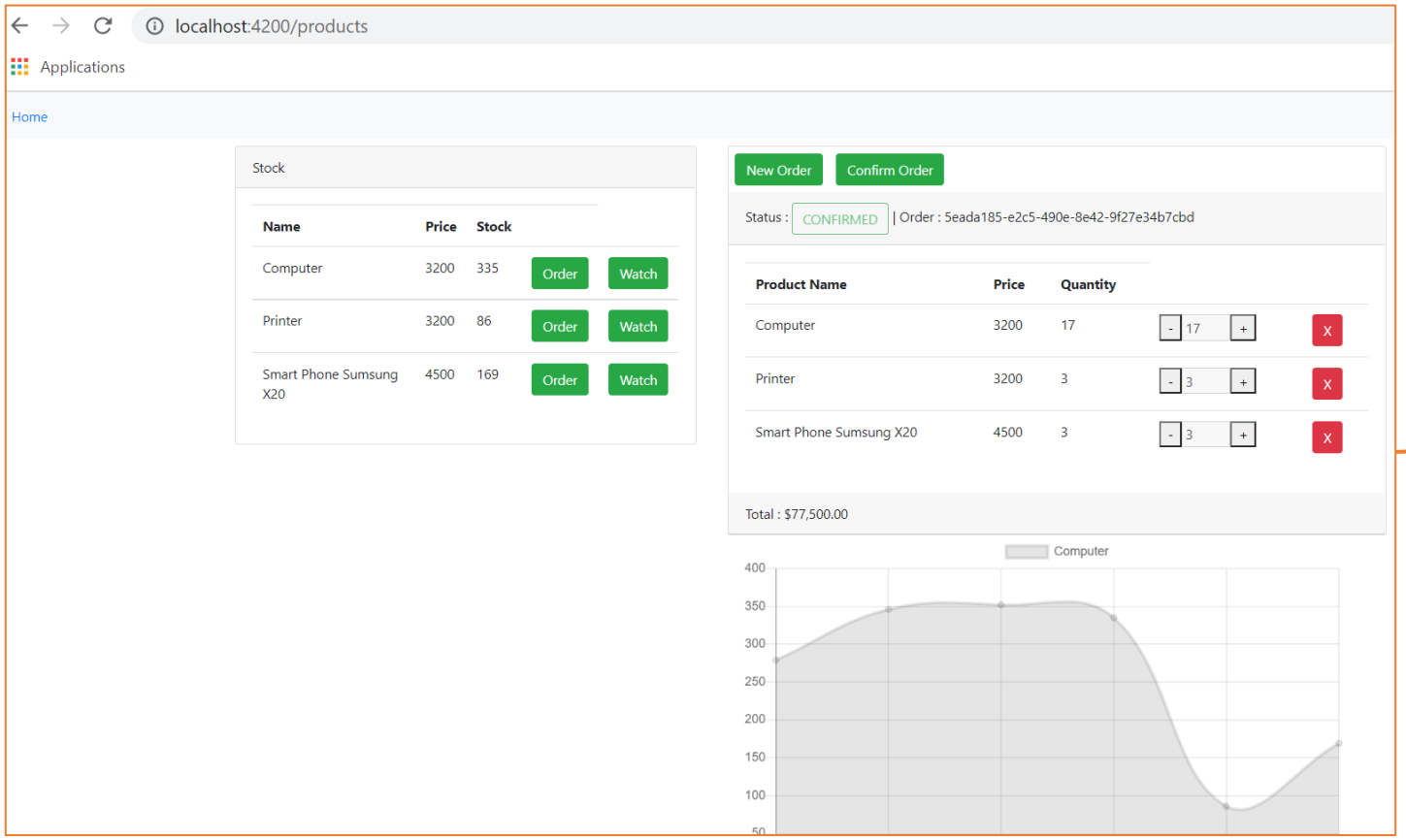
Query Side : Test

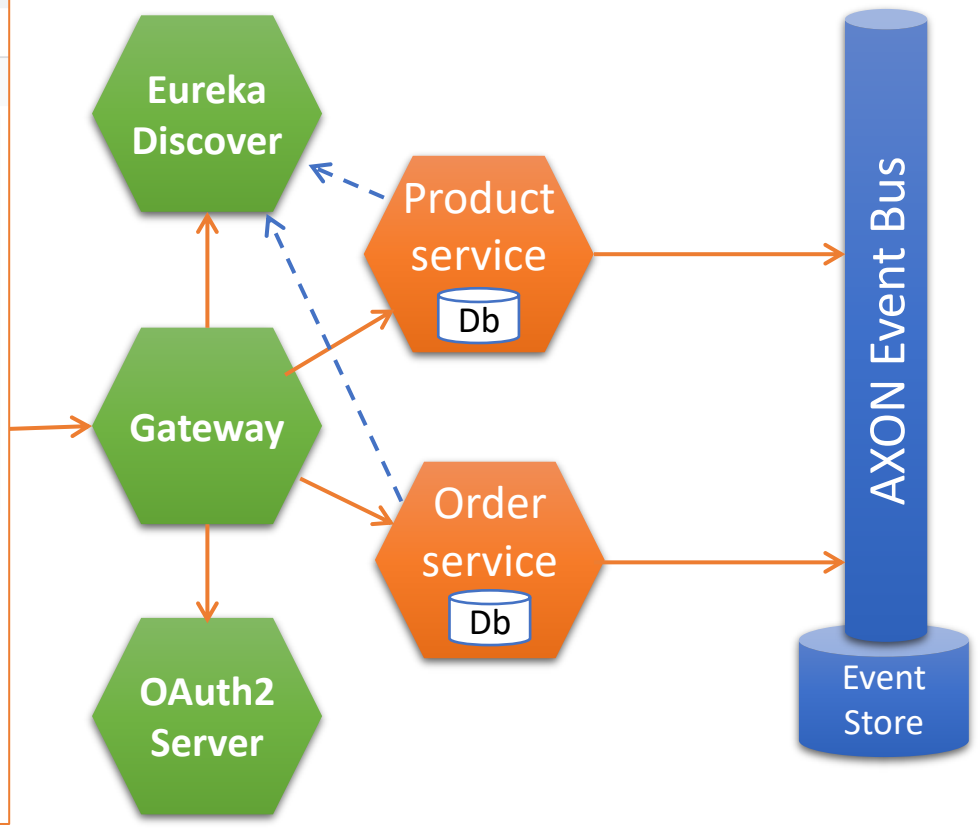
Application

- Créer une application basée sur les micro-services qui permet de gérer des commandes de produits et qui respecte les patterns CQRS et Event Sourcing
- L'application est décomposée en deux micro-services fonctionnels:
  - Product-Serrvice : Permet de gérer des produits
  - Order-Service : Permet de gérer des commandes
- Les services techniques de l'architecture sont :
  - Spring Cloud Gateway Service
  - Eureka Discovery Service
  - Oauth2 JWT Authentication Service
  - L'Event Bus et l'Event Store sont assurée par AXON Server
- La partie Frontend Web est basée sur Angular

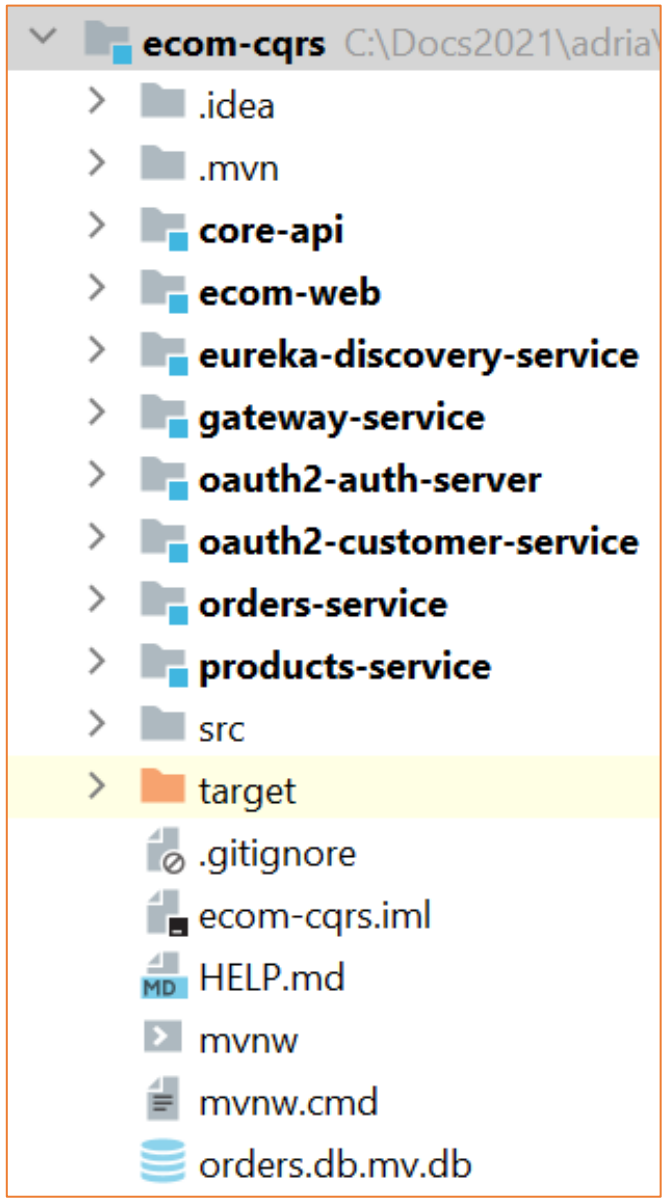


Application





Structure du Projet




```
<modules>
  <module>products-service</module>
  <module>orders-service</module>
  <module>core-api</module>
</modules>
```

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.5.3</version>
  <relativePath/>
</parent>
```

```
<properties>
  <java.version>1.8</java.version>
  <kotlin.version>1.5.30-M1</kotlin.version>
</properties>
```

Maven Dependencies

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>

  <dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <scope>runtime</scope>
  </dependency>

  <dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
  </dependency>

  <dependency>
    <groupId>org.axonframework</groupId>
    <artifactId>axon-spring-boot-starter</artifactId>
    <version>4.5.2</version>
  </dependency>

  <dependency>
    <groupId>io.projectreactor</groupId>
    <artifactId>reactor-core</artifactId>
    <version>3.4.8</version>
  </dependency>

  <dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-ui</artifactId>
    <version>1.5.2</version>
  </dependency>

<dependency>
    <groupId>org.mapstruct</groupId>
    <artifactId>mapstruct</artifactId>
    <version>1.4.2.Final</version>
  </dependency>
```

Maven Dependencies

```
  <dependency>
    <groupId>org.jetbrains.kotlin</groupId>
    <artifactId>kotlin-stdlib-jdk8</artifactId>
    <version>${kotlin.version}</version>
  </dependency>
  <dependency>
    <groupId>org.jetbrains.kotlin</groupId>
    <artifactId>kotlin-test</artifactId>
    <version>${kotlin.version}</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.jetbrains.kotlin</groupId>
    <artifactId>kotlin-stdlib-jdk8</artifactId>
    <version>${kotlin.version}</version>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies >
```

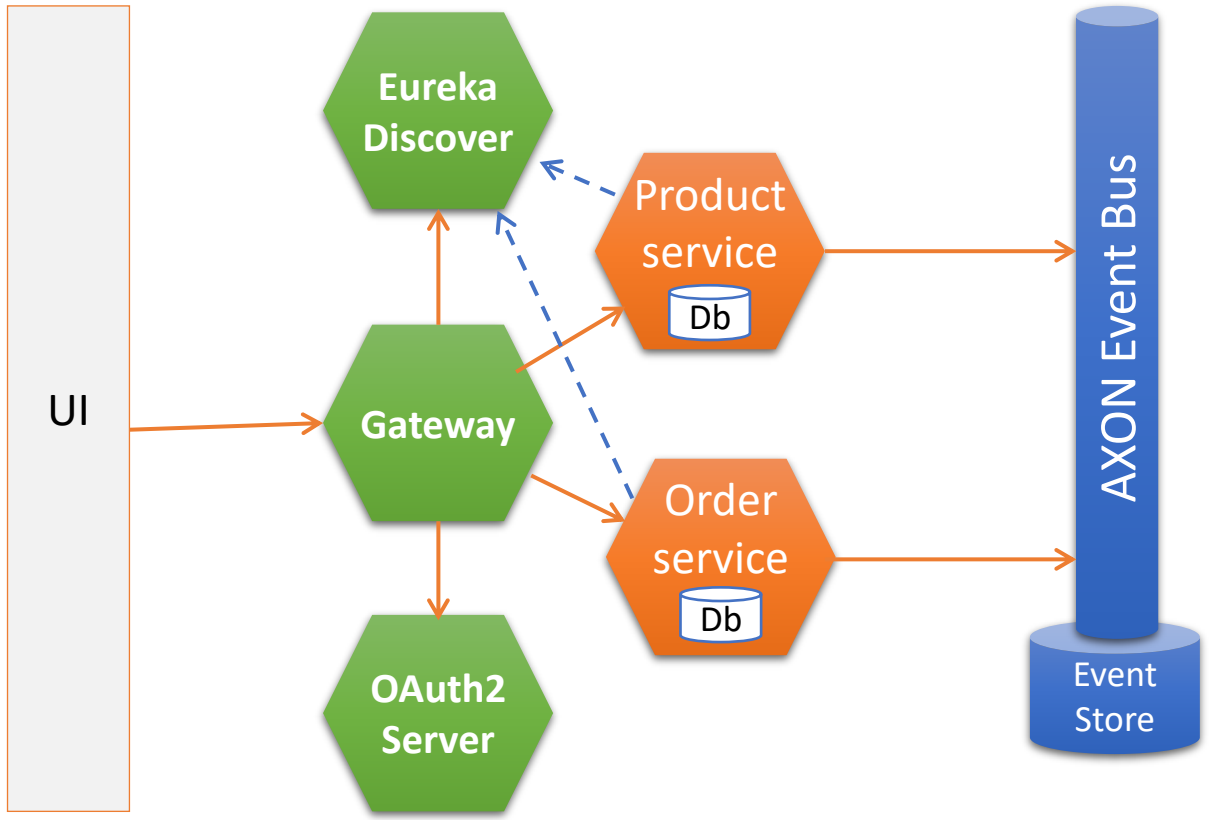
Maven Plugins

```xml
<plugins>
  <plugin>
    <groupId>org.jetbrains.kotlin</groupId>
    <artifactId>kotlin-maven-plugin</artifactId>
    <version>${kotlin.version}</version>
    <executions>
      <execution>
        <id>compile</id>
        <phase>compile</phase>
        <goals>
          <goal>compile</goal>
        </goals>
      </execution>
      <execution>
        <id>test-compile</id>
        <phase>test-compile</phase>
        <goals>
          <goal>test-compile</goal>
        </goals>
      </execution>
    </executions>
    <configuration>
      <jvmTarget>1.8</jvmTarget>
    </configuration>
  </plugin>
```

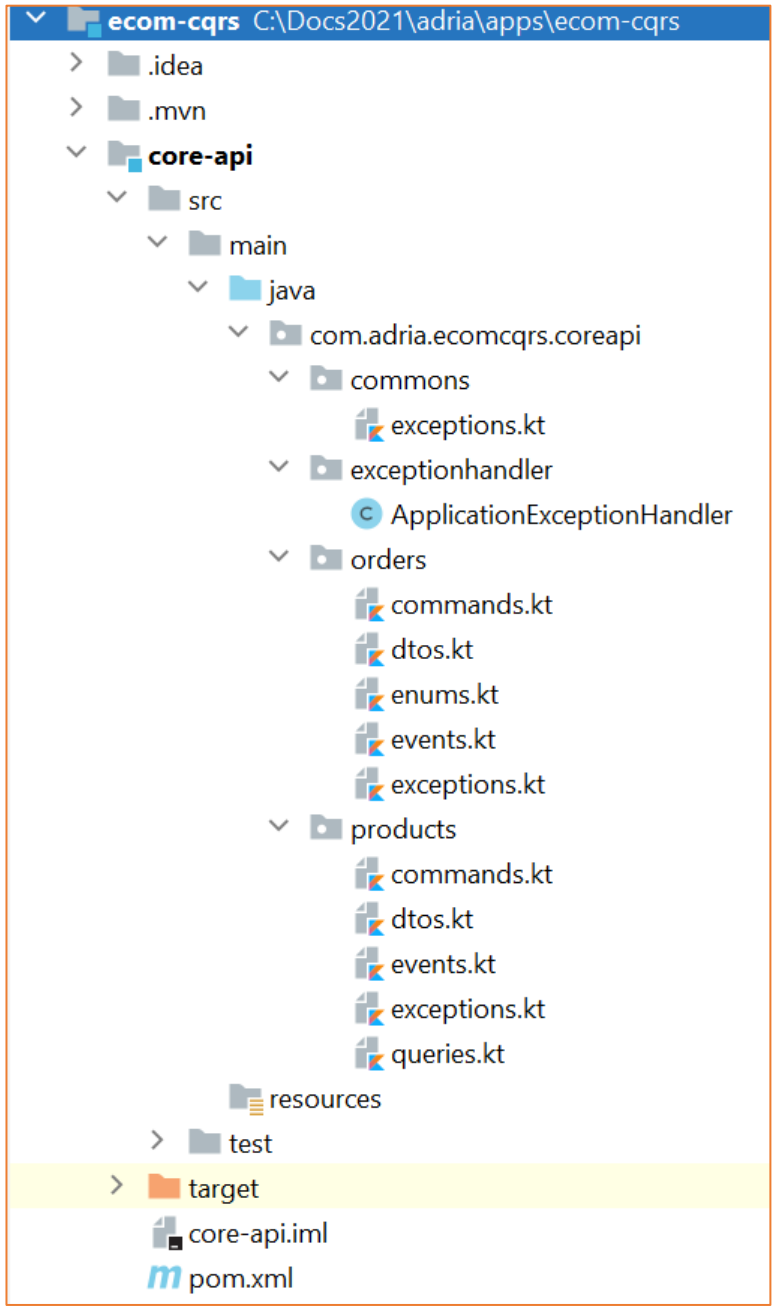
```xml
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>3.8.1</version>
    <configuration>
      <source>1.8</source>
      <target>1.8</target>
      <annotationProcessorPaths>
        <path>
          <groupId>org.projectlombok</groupId>
          <artifactId>lombok</artifactId>
          <version>1.18.16</version>
        </path>
        <path>
          <groupId>org.mapstruct</groupId>
          <artifactId>mapstruct-processor</artifactId>
          <version>1.4.2.Final</version>
        </path>
        <!-- other annotation processors -->
      </annotationProcessorPaths>
    </configuration>
  </plugin>
</plugins>
```

```xml
<plugin>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-maven-plugin</artifactId>
    <configuration>
      <excludes>
        <exclude>
          <groupId>org.projectlombok</groupId>
          <artifactId>lombok</artifactId>
        </exclude>
      </excludes>
    </configuration>
  </plugin>
```

CORE-API


UI
Eureka Discover
Gateway
OAuth2 Server
Product service
Db
Order service
Db
AXON Event Bus
Event Store

Module core-api




```
<parent>
   <artifactId>ecom-cqrs</artifactId>
   <groupId>com.adria</groupId>
   <version>0.0.1-SNAPSHOT</version>
</parent>
<modelVersion>4.0.0</modelVersion>

<artifactId>core-api</artifactId>

<properties>
   <maven.compiler.source>8</maven.compiler.source>
   <maven.compiler.target>8</maven.compiler.target>
</properties>
```

Module core-api / Orders : Commands

Command.kt

```
package com.adria.ecomcqrs.coreapi.orders

import
org.axonframework.modelling.command.Targ
etAggregateIdentifier

data class CreateOrderCommand(
        val status : OrderStatus
)

data class AddOrderLineCommand(
        @TargetAggregateIdentifier
        val orderId : String,
        val productId : String,
        val quantity : Double,
        val price : Double
)
```

```
data class RemoveOrderLineCommand(
        @TargetAggregateIdentifier
        val orderId : String,
        val productId : String,
)

data class UpdateOrderLineCommand(
        @TargetAggregateIdentifier
        val orderId : String,
        val productId : String,
        val quantity : Double,
        val price : Double
)

data class ConfirmOrderCommand(
        @TargetAggregateIdentifier
        val orderId : String,
        val status : OrderStatus
)
```

enums.kt

```
package
com.adria.ecomcqrs.corea
pi.orders

enum class OrderStatus{
    CREATED, CONFIRMED
}
```

Module core-api / Orders : Events

events.kt

```
package com.adria.ecomcqrs.coreapi.orders
import java.util.*

data class OrderCreatedEvent(
        val orderId:String,
        val date:Date,
        val status : OrderStatus
)

data class OrderLineAddedEvent(
        val orderId:String,
        val productId:String,
        val quantity:Double,
        val price : Double
)
```

```
data class OrderLineRemovedEvent(
        val orderId:String,
        val productId:String,
)

data class OrderLineUpdatedEvent(
        val orderId : String,
        val productId : String,
        val quantity : Double,
        val price : Double
)
data class OrderConfirmedEvent(
        val orderId:String,
        val status : OrderStatus
)
```

Module core-api / Orders : Queries ans DTOs

dtos.kt

```
package com.adria.ecomcqrs.coreapi.orders

import java.util.*

data class AddOrderLineRequestDTO(
        val orderId:String,
        val productId:String,
        val quantity:Int,
        val price : Double
)

data class RemoveOrderLineRequestDTO(
        val orderId:String,
        val productId:String,
)
data class UpdateOrderLineRequestDTO(
        val orderId:String,
        val productId:String,
        val quantity:Int,
        val price : Double
)
```

```
data class ConfirmOrderRequestDTO(
        val orderId:String,
        val value:String
)

data class ProductDTO(
        var productId : String?,
        var name : String?
)

data class ProductResponseDTO(
        var productId : String?,
        var name : String?,
        var price : Double?,
        var stock: Double?,
        var eventType : String?
)
```

```
data class OrderLineDTO(
        var productId: String?,
        var productName : String?,
        var price : Double?,
        var quantity : Double?
)
data class OrderDTO(
        var orderId : String?,
        var date : Date?,
        var status : OrderStatus?,
        var
orderItems:List<OrderLineDTO>?
)

data class GetOrderById(var orderId
: String?)
```

Module core-api / Orders : Exceptions

Exceptions.kt

```
package com.adria.ecomcqrs.coreapi.commons

import java.lang.RuntimeException
abstract class BusinessEcommerceException(message:String) : RuntimeException(message)
```

exceptions

```
package com.adria.ecomcqrs.coreapi.orders

import com.adria.ecomcqrs.coreapi.commons.BusinessEcommerceException

class OrderLineNotExistException(message:String) : BusinessEcommerceException(message)

class QuantityException(message:String) : BusinessEcommerceException(message)

class OrderLineAlreadyExistException(message: String) : BusinessEcommerceException(message)

class OrderAlreadyConfirmedException(message: String) : BusinessEcommerceException(message)

class OrderEmptyException(message: String) : BusinessEcommerceException(message)

class ProductQuantityNotAvailableException(message: String) : BusinessEcommerceException(message)
```

Module core-api / Products : Commands

Command.kt

```
package
com.adria.ecomcqrs.coreapi.products;

import
org.axonframework.modelling.command.Targ
etAggregateIdentifier

data class CreateProductCommand(
        val name : String,
        val price : Double,
        val stock : Double
)
data class UpdateProductPriceCommand(
        @TargetAggregateIdentifier
        val productId : String,
        val price : Double
)
```

```
data class ProvisionProductStockCommand(
        @TargetAggregateIdentifier
        val productId : String,
        val quantity : Double
)
```

Module core-api / Products : Events

events.kt

```
package
com.adria.ecomcqrs.coreapi.products;

data class ProductCreatedEvent(
        val productId : String,
        val name : String,
        val price : Double,
        val stock : Double
)
data class ProductPriceUpdatedEvent(
        val productId : String,
        val price : Double
)
data class ProductStockProvisionedEvent(
        val productId : String,
        val quantity : Double
)
```

```
data class ProductOrderedEvent(
        val orderId : String,
        val productId : String,
        val quantity : Double,
        val price : Double
)
```

Module core-api / Products : Queries ans DTOs

dtos.kt

```
package
com.adria.ecomcqrs.coreapi.products;

data class CreateProductRequestDTO(
        val name : String,
        val price : Double,
        val stock : Double
)

data class UpdateProductPriceRequestDTO(
        val productId : String,
        val price : Double
)

data class UpdateProductStockRequestDTO(
        val productId : String,
        val quantity : Double
)
```

```
data class ProvisionProductStockRequestDTO(
        val productId : String,
        val quantity : Double
)
```

queries.kt

```
package com.adria.ecomcqrs.coreapi.products;

class GetAllProductsQuery ()
data class GetProductByIdQuery( val productId : String)
```

Module core-api / Products : Exceptions

Exceptions.kt

```
package com.adria.ecomcqrs.coreapi.commons

import java.lang.RuntimeException
abstract class BusinessEcommerceException(message:String) : RuntimeException(message)
```

exception.kt

```
package com.adria.ecomcqrs.coreapi.products;

import com.adria.ecomcqrs.coreapi.commons.BusinessEcommerceException

class StockInsufficientException(message:String) : BusinessEcommerceException(message)
class ProductNotFoundException(message:String) : BusinessEcommerceException(message)
```

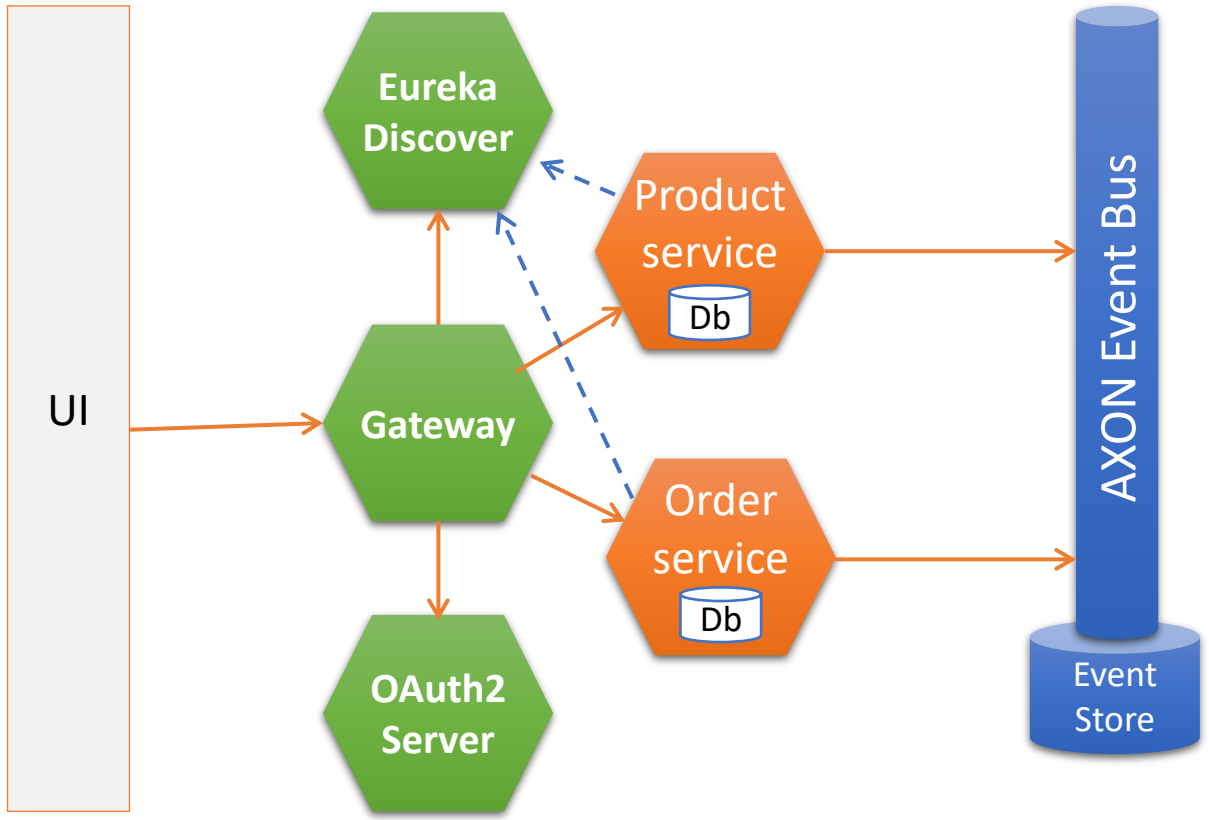
Module core-api / Application Exception Handler

ApplicationExceptionHandler.java

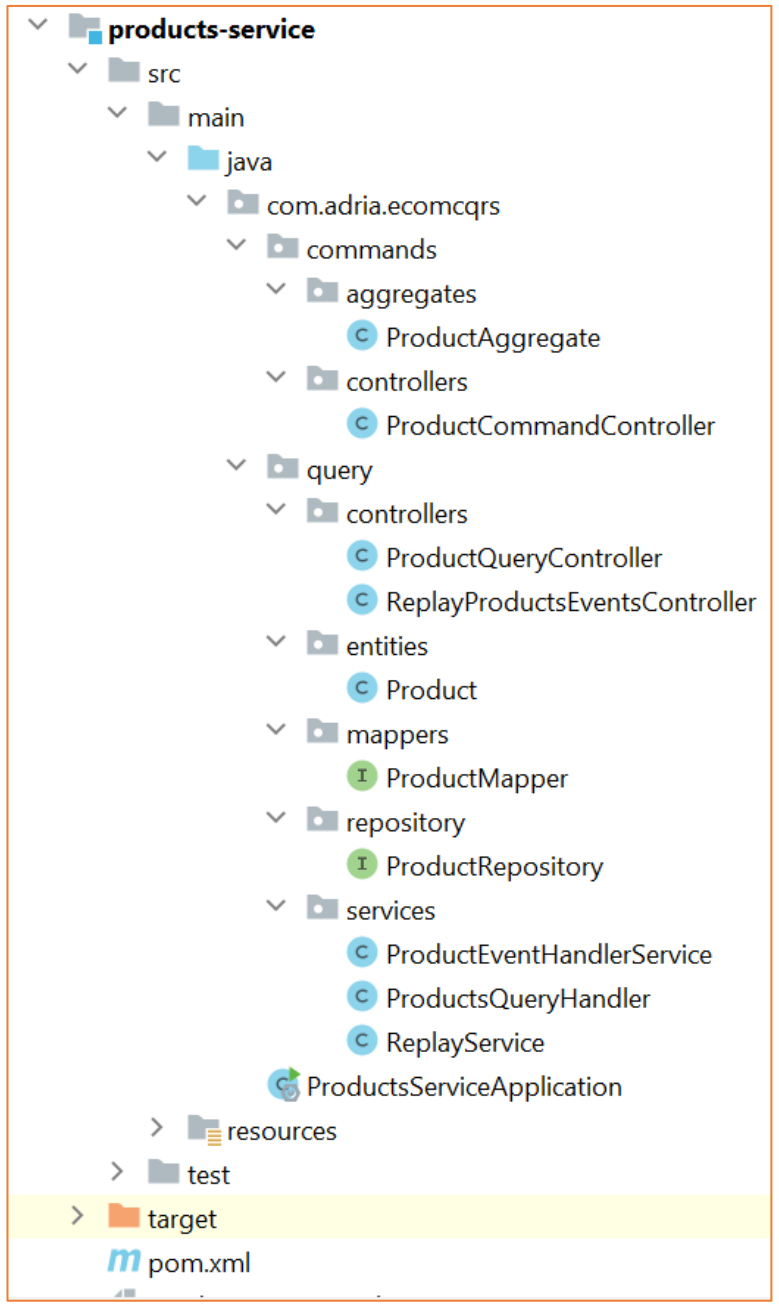
```
package com.adria.ecomcqrs.coreapi.exceptionhandler;
import com.adria.ecomcqrs.coreapi.commons.BusinessEcommerceException;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;

import java.util.HashMap;
import java.util.Map;

@ControllerAdvice
public class ApplicationExceptionHandler {
    @ExceptionHandler({BusinessEcommerceException.class})
    public ResponseEntity<Map<String,String>> exceptionHandler(BusinessEcommerceException exception){
        exception.printStackTrace();
        Map<String,String> errorMessage=new HashMap<>();
        errorMessage.put("message",exception.getMessage());
        errorMessage.put("status", HttpStatus.INTERNAL_SERVER_ERROR.toString());
        return new ResponseEntity<>(errorMessage,HttpStatus.INTERNAL_SERVER_ERROR);
    }
}
```

Product-service

Commands Side



Module Products-service




```
<parent>
   <artifactId>ecom-cqrs</artifactId>
   <groupId>com.adria</groupId>
   <version>0.0.1-SNAPSHOT</version>
</parent>
<modelVersion>4.0.0</modelVersion>

<artifactId>products-service</artifactId>

<properties>
   <maven.compiler.source>8</maven.compiler.source>
   <maven.compiler.target>8</maven.compiler.target>
   <spring-cloud.version>2020.0.3</spring-cloud.version>
</properties>
<packaging>jar</packaging>
```

Module Products-service : Maven Dependencies

```
<dependency>
   <groupId>com.adria</groupId>
   <artifactId>core-api</artifactId>
   <version>0.0.1-SNAPSHOT</version>
</dependency>

<dependency>
   <groupId>org.springframework.cloud</groupId>
   <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
</dependency>

<dependency>
   <groupId>com.google.guava</groupId>
   <artifactId>guava</artifactId>
   <version>24.0-jre</version>
</dependency>
```

Module Products-service : Maven Dependency Management (Spring Cloud)

```
<dependencyManagement>
   <dependencies>
      <dependency>
         <groupId>org.springframework.cloud</groupId>
         <artifactId>spring-cloud-dependencies</artifactId>
         <version>${spring-cloud.version}</version>
         <type>pom</type>
         <scope>import</scope>
      </dependency>
   </dependencies>
</dependencyManagement>
```

Module Products-service : application.properties

```
server.port=8094
spring.application.name=product-service
spring.datasource.url=jdbc:h2:mem:product-db
axon.axonserver.enabled=true
spring.h2.console.enabled=true

eureka.instance.prefer-ip-address=true
spring.cloud.discovery.enabled=false
```

Module Products-service : ProductAggregate

```
@Aggregate
@Slf4j
public class ProductAggregate {
    @AggregateIdentifier
    private String productId;
    private String name;
    private Double price;
    private Double stock;
    public ProductAggregate() {
    }
    @CommandHandler
    public ProductAggregate(CreateProductCommand command) {
        log.info("CreateProductCommand Received");
        AggregateLifecycle.apply(new ProductCreatedEvent(
                UUID.randomUUID().toString(),
                command.getName(),
                command.getPrice(),
                command.getStock()
        ));
    }
```

Module Products-service : ProductAggregate

```
@EventSourcingHandler
public void on(ProductCreatedEvent event){
    log.info("ProductCreatedEvent Occured");
    this.productId=event.getProductId();
    this.name=event.getName();
    this.price=event.getPrice();
    this.stock=event.getStock();
}

@CommandHandler
public void handle(UpdateProductPriceCommand command) {
    log.info("UpdateProductPriceCommand Received");
    AggregateLifecycle.apply(new ProductPriceUpdatedEvent(
            command.getProductId(),
            command.getPrice()
    ));
}
```

Module Products-service : ProductAggregate

```
  @EventSourcingHandler
   public void on(ProductPriceUpdatedEvent event){
      log.info("ProductPriceUpdatedEvent Occured");
      this.price=event.getPrice();
   }
   @CommandHandler
   public void handle(ProvisionProductStockCommand command) {
      log.info("ProvisionProductStockCommand Received");
      AggregateLifecycle.apply(new ProductStockProvisionedEvent(
            command.getProductId(),
            command.getQuantity()
      ));
   }
   @EventSourcingHandler
   public void on(ProductStockProvisionedEvent event){
      log.info("ProductStockUpdatedEvent Occured");
      this.stock+=event.getQuantity();
   }
}
```

Module Products-service : ProductCommandController

```
//@CrossOrigin("*")
@RestController
@AllArgsConstructor
public class ProductCommandController {
    private CommandGateway commandGateway;
    private EventStore eventStore;
    @PostMapping("/products/commands/create")
    public CompletableFuture<String> save(@RequestBody CreateProductRequestDTO request){
        return commandGateway.send(new CreateProductCommand(
                request.getName(),request.getPrice(), request.getStock()
        ));
    }
    @PutMapping("/products/commands/updatePrice")
    public CompletableFuture<String> updatePrice(@RequestBody UpdateProductPriceRequestDTO request){
        return commandGateway.send(new UpdateProductPriceCommand(
                request.getProductId(),request.getPrice()
        ));
    }
```

Module Products-service : ProductCommandController

```
    @PutMapping("/products/commands/provisionStock")
    public CompletableFuture<String> provisionStock(@RequestBody ProvisionProductStockRequestDTO request){
        return commandGateway.send(new ProvisionProductStockCommand(
                request.getProductId(),request.getQuantity()
        ));
    }

    @GetMapping("products/{productId}/events")
    public Stream productEvents(@PathVariable String productId){
        return eventStore.readEvents(productId).asStream();
    }
}
```

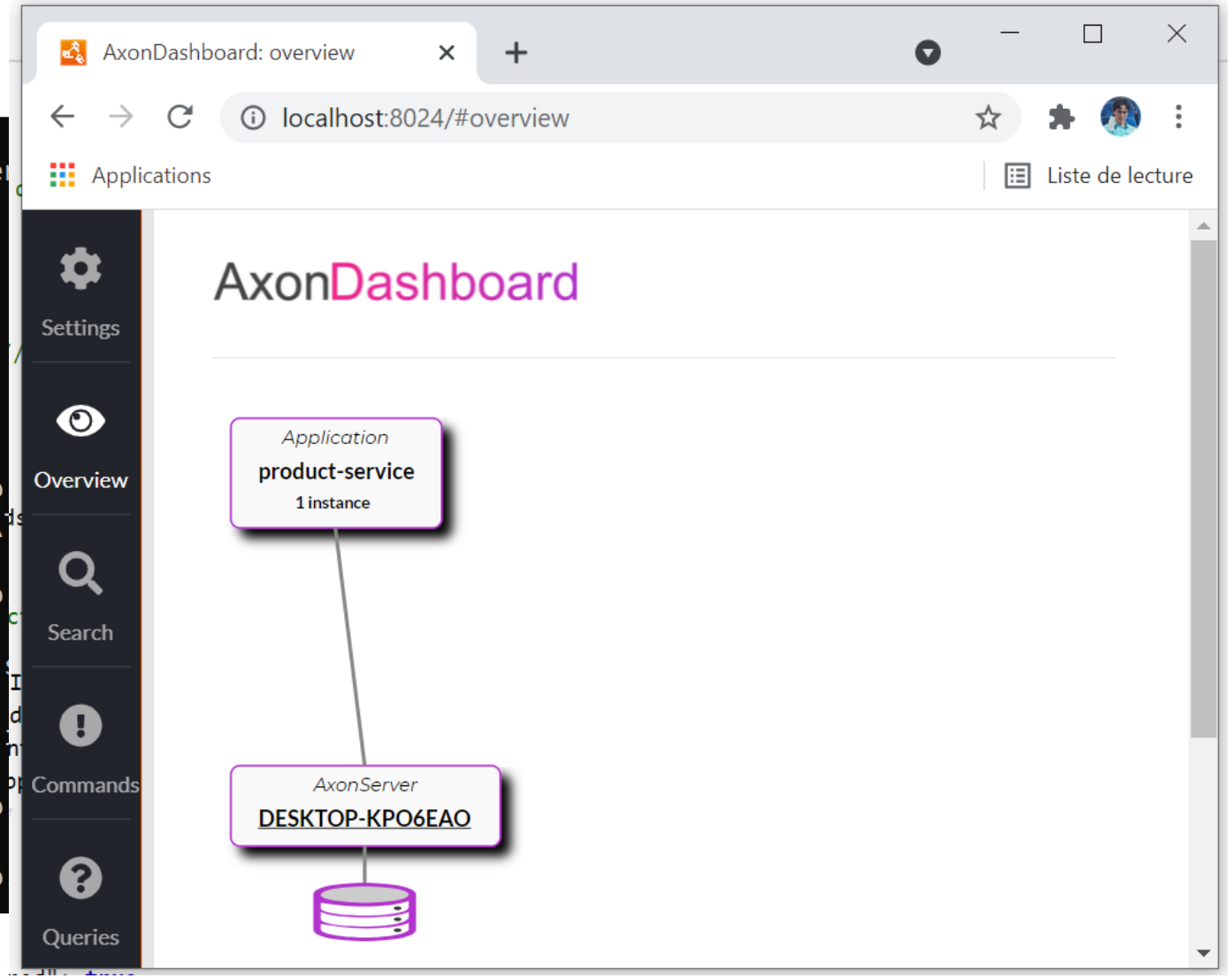
Module Products-service : ProductsServiceApplication

```
package com.adria.ecomcqrs;

@SpringBootApplication
public class ProductsServiceApplication {
    public static void main(String[] args) {
        SpringApplication.run(ProductsServiceApplication.class);
    }
    @Bean
    public CommandBus commandBus(TransactionManager txManager, AxonConfiguration axonConfiguration){
        return SimpleCommandBus.builder().build();
    }

}
```

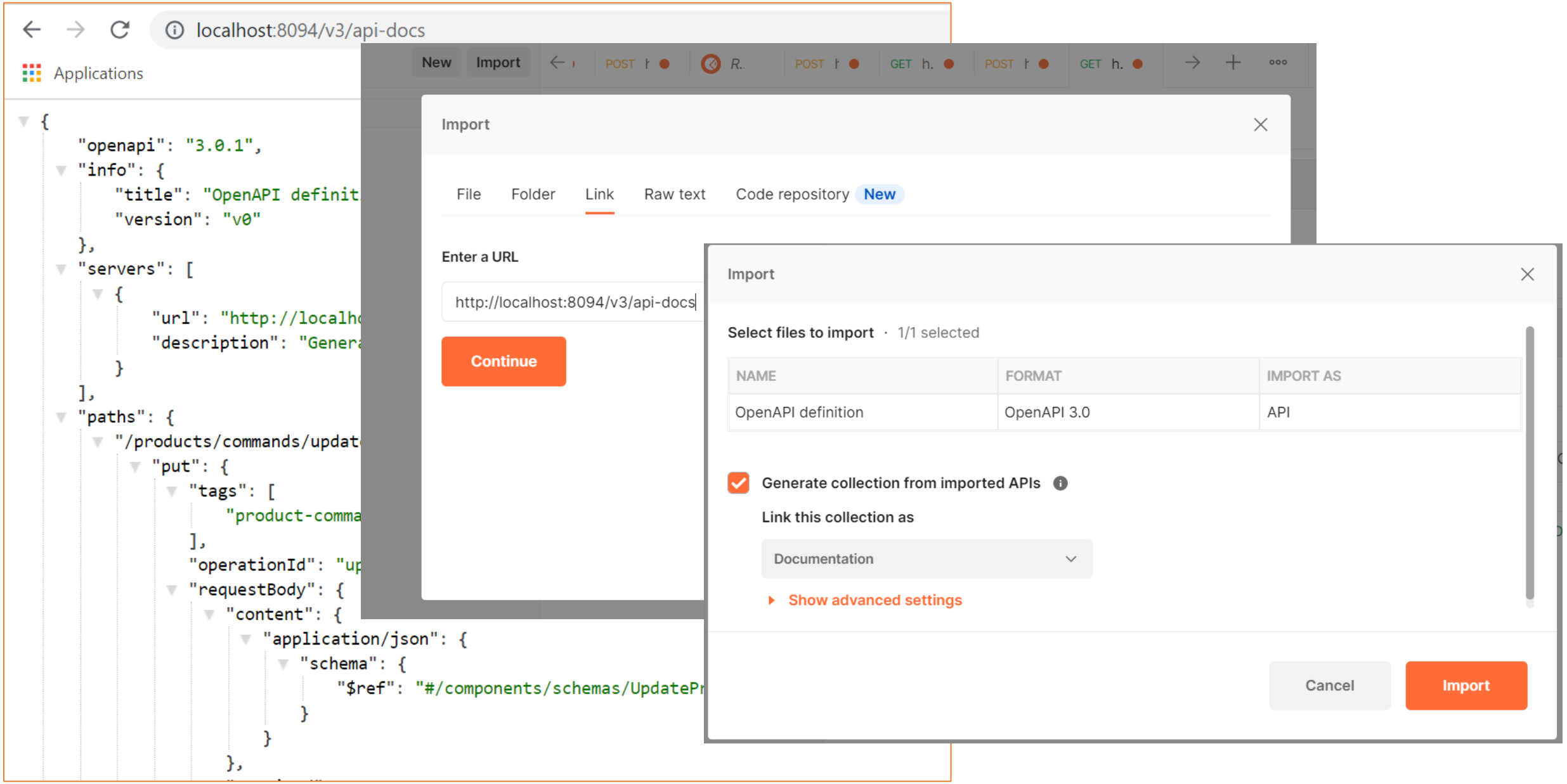
Starting Axon Server

Invite de commandes - java -jar axonserver-4.5.5.jar

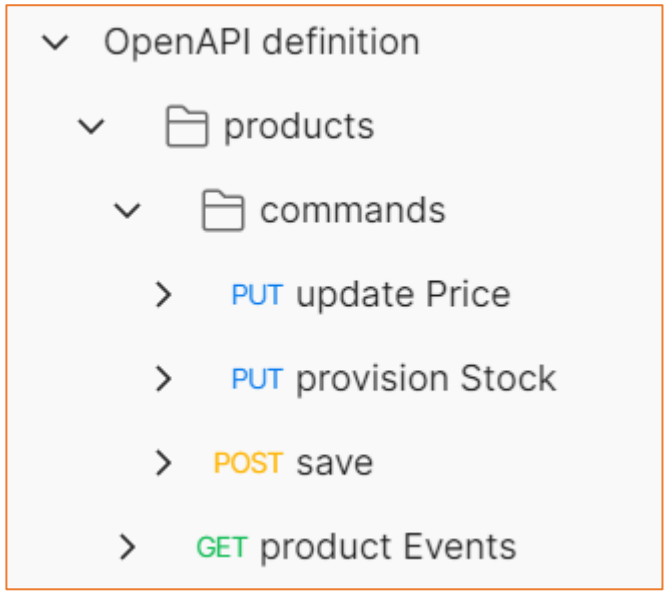
```
C:\Tools\axonquickstart-4.5.5\AxonServer>java -jar axonserver
 Standard Edition                                Powered by AxonIQ

version: 4.5.5
2021-08-22 11:16:07.460  INFO 16732 --- [           main] io
 using Java 1.8.0_291 on DESKTOP-KPO6EAO with PID 16732 (C:\
tarted by med in C:\Tools\axonquickstart-4.5.5\AxonServer)
2021-08-22 11:16:07.464  INFO 16732 --- [           main] io
et, falling back to default profiles: default
2021-08-22 11:16:10.779  INFO 16732 --- [           main] o.
with port(s): 8024 (http)
2021-08-22 11:16:12.410  INFO 16732 --- [           main] A.
alized with SSL DISABLED and access control DISABLED.
2021-08-22 11:16:14.882  INFO 16732 --- [           main] io
 4.5.5
2021-08-22 11:16:20.156  INFO 16732 --- [           main] io
```



Module Products-service : Commands Side Test



Module Products-service : Commands Side Tests : Save New product



POST {{baseUrl}}/products/commands/create

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

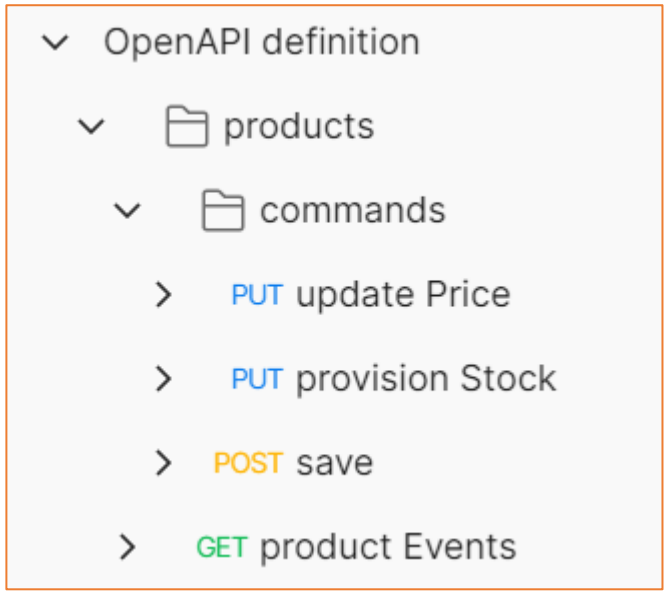
```
{
    "name": "Smart Phone Sumsung X20",
    "price": 4500,
    "stock": 12
}
```

Body Cookies Headers (5) Test Results 200 OK 233 ms 200 B

Pretty Raw Preview Visualize Text

```
577d20a3-5038-4b42-bd59-84a7adac69d1
```

Module Products-service : Commands Side Test : Update Product Price



PUT {{baseUrl}}/products/commands/updatePrice

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

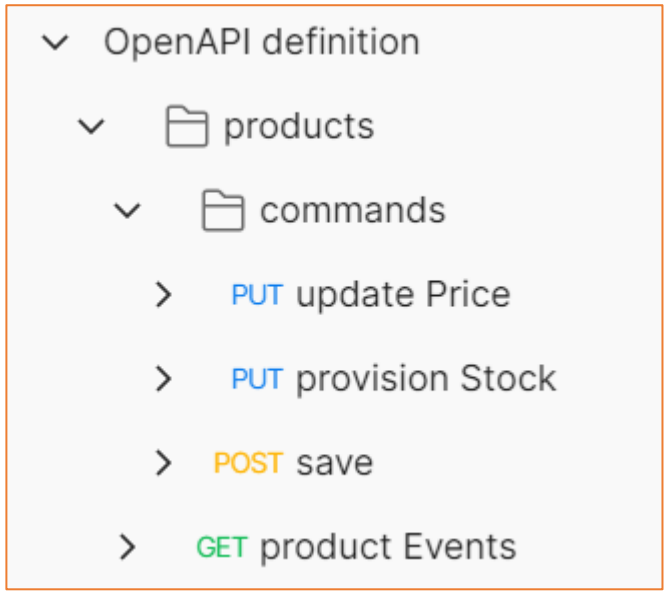
```
{
    "productId": "577d20a3-5038-4b42-bd59-84a7adac69d1",
    "price": 4500
}
```

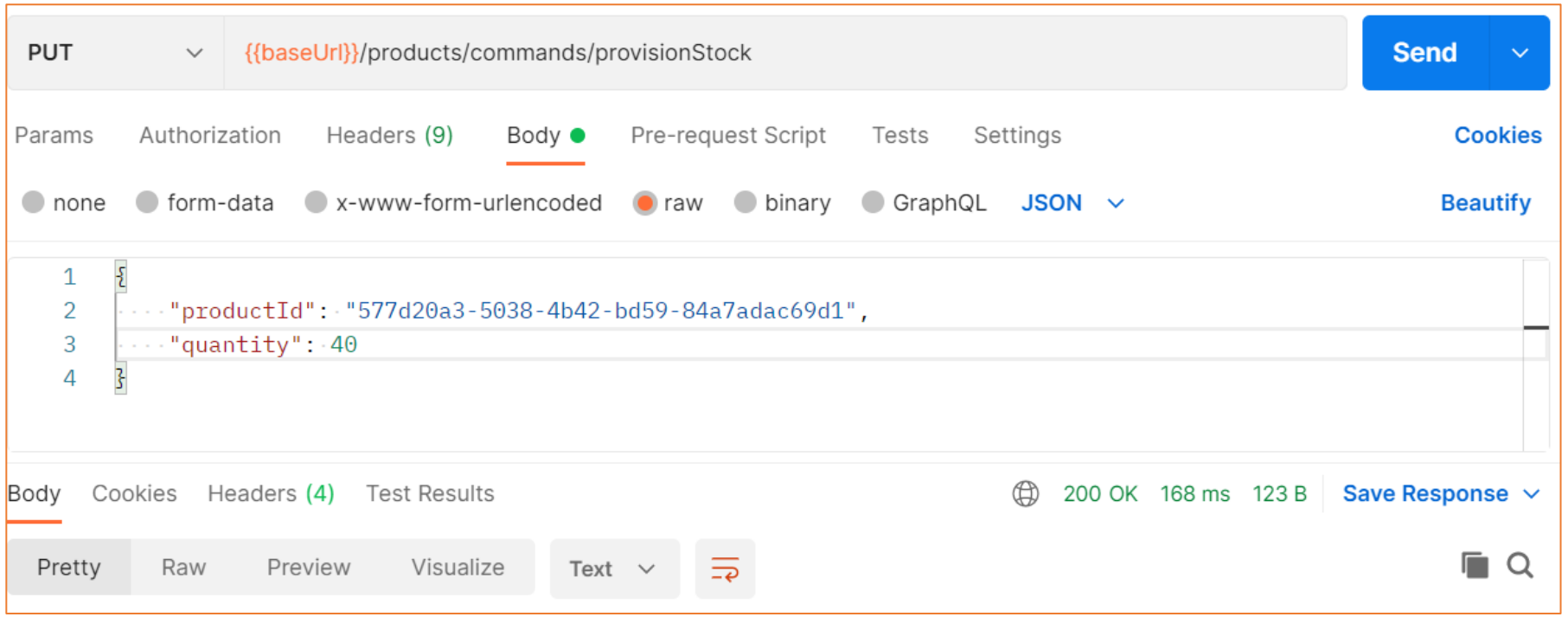
Body Cookies Headers (4) Test Results 200 OK 369 ms 123 B

Pretty Raw Preview Visualize Text

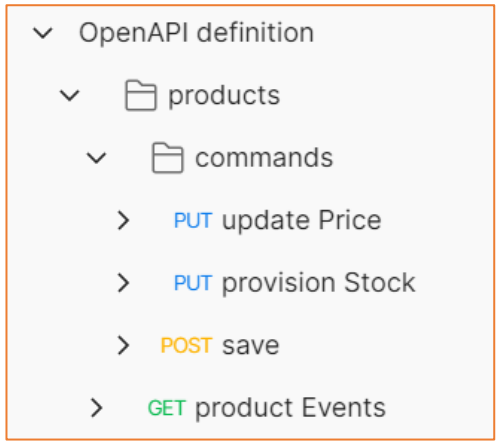
```
1
```

Module Products-service : Commands Side Test : Update Product Stock





Module Products-service : Commands Side Test : Get Product Events



GET {{baseUrl}}/products/577d20a3-5038-4b42-bd59-84a7adac69d1/events Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	ooo	Bulk Edit

Body Cookies Headers (5) Test Results 200 OK 55 ms 1.28 KB Save Response

Pretty Raw Preview Visualize JSON


```
[
    {
        "type": "ProductAggregate",
        "aggregateIdentifier": "577d20a3-5038-4b42-bd59-84a7adac69d1",
        "sequenceNumber": 0,
        "identifier": "e2263bfe-71e7-43a2-a62d-3b96782574be",
        "timestamp": "2021-08-26T11:00:29.444Z",
        "payload": {
            "productId": "577d20a3-5038-4b42-bd59-84a7adac69d1",
            "name": "Smart Phone Sumsung X20",
            "price": 4500.0,
            "stock": 12.0
```

Module Products-service : Commands Side Test : Get Product Events


```
[
  {
    "type": "ProductAggregate",
    "aggregateIdentifier": "577d20a3-5038-4b42-bd59-84a7adac69d1",
    "sequenceNumber": 0,
    "identifier": "e2263bfe-71e7-43a2-a62d-3b96782574be",
    "timestamp": "2021-08-26T11:00:29.444Z",
    "payload": {
      "productId": "577d20a3-5038-4b42-bd59-84a7adac69d1",
      "name": "Smart Phone Sumsung X20",
      "price": 4500.0,
      "stock": 12.0
    },
    "metaData": {},
    "payloadType": "com.adria.ecomcqrs.coreapi.products.ProductCreatedEvent"
  },
```

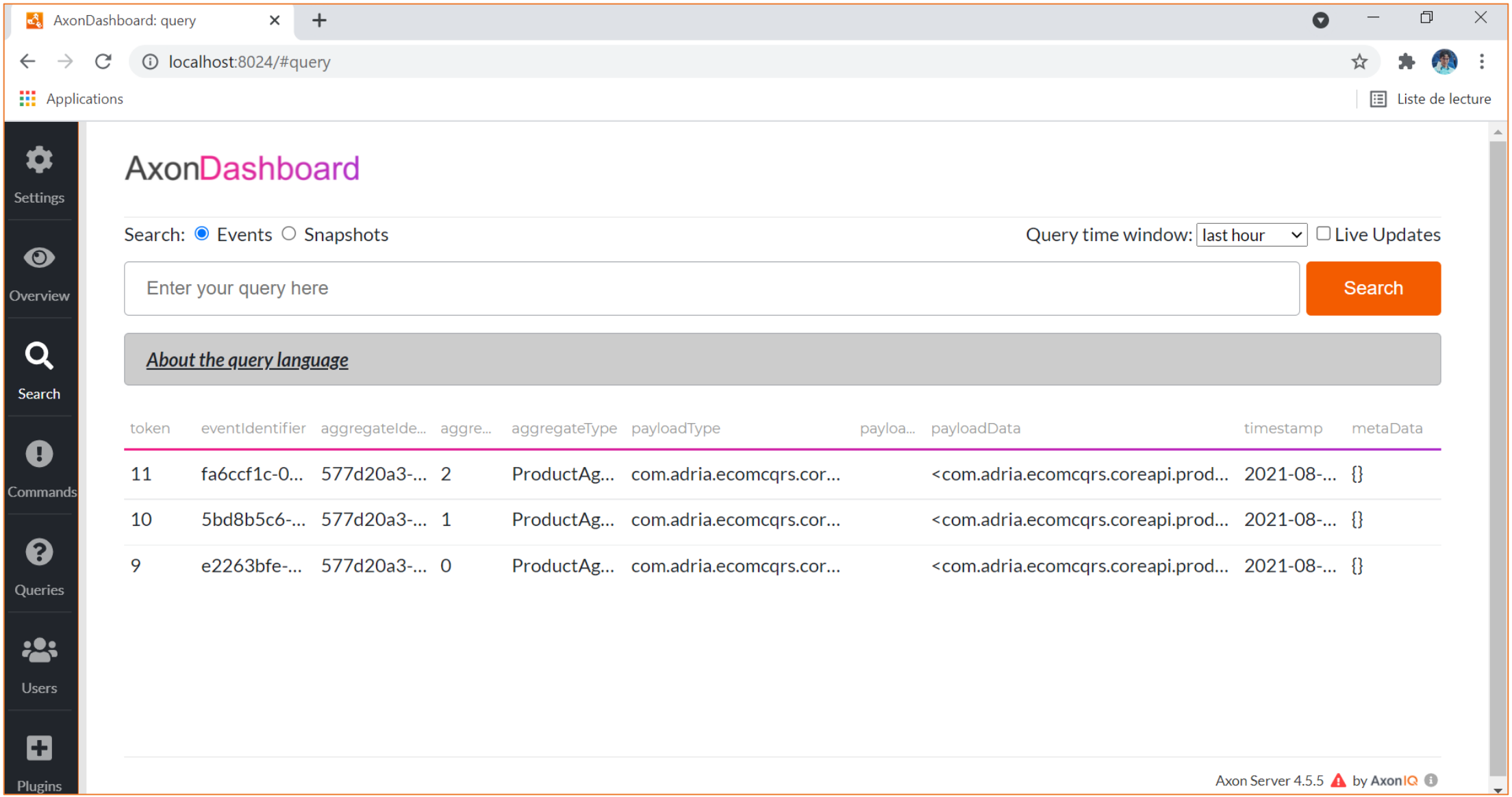
Module Products-service : Commands Side Test : Get Product Events


```
{
    "type": "ProductAggregate",
    "aggregateIdentifier": "577d20a3-5038-4b42-bd59-84a7adac69d1",
    "sequenceNumber": 1,
    "identifier": "5bd8b5c6-279b-4018-a068-57be678ce5e6",
    "timestamp": "2021-08-26T11:02:14.192Z",
    "payload": {
      "productId": "577d20a3-5038-4b42-bd59-84a7adac69d1",
      "price": 4500.0
    },
    "metaData": {},
    "payloadType": "com.adria.ecomcqrs.coreapi.products.ProductPriceUpdatedEvent"
  },
```

Module Products-service : Commands Side Test : Get Product Events

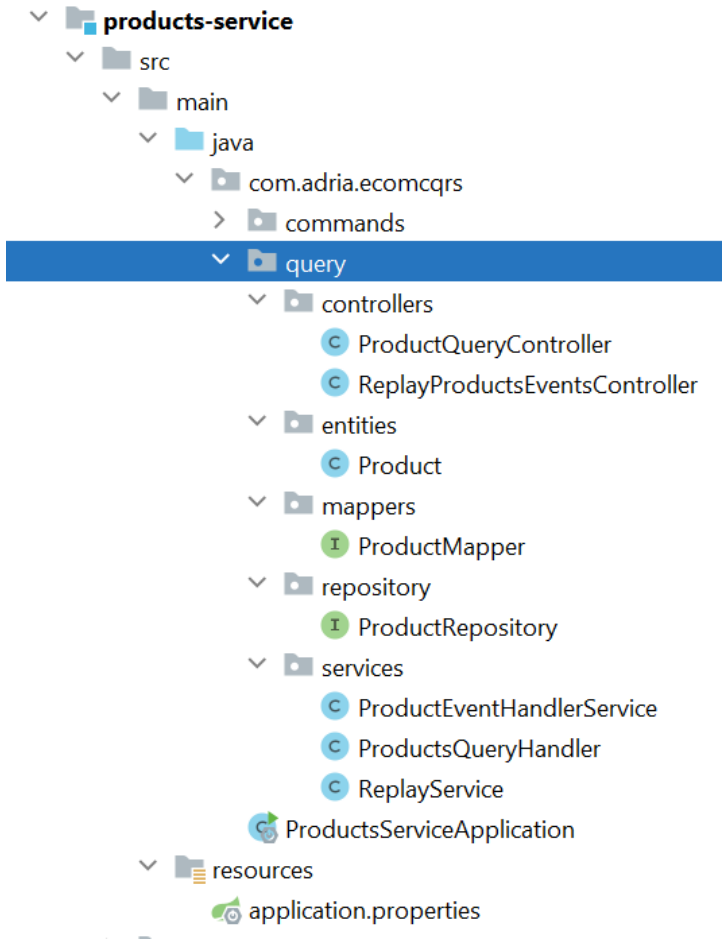

```
{
      "type": "ProductAggregate",
      "aggregateIdentifier": "577d20a3-5038-4b42-bd59-84a7adac69d1",
      "sequenceNumber": 2,
      "identifier": "fa6ccf1c-0ca7-4d44-97e6-78fca1ef9108",
      "timestamp": "2021-08-26T11:04:21.822Z",
      "payload": {
         "productId": "577d20a3-5038-4b42-bd59-84a7adac69d1",
         "quantity": 40.0
      },
      "metaData": {},
      "payloadType": "com.adria.ecomcqrs.coreapi.products.ProductStockProvisionedEvent"
   }
]
```

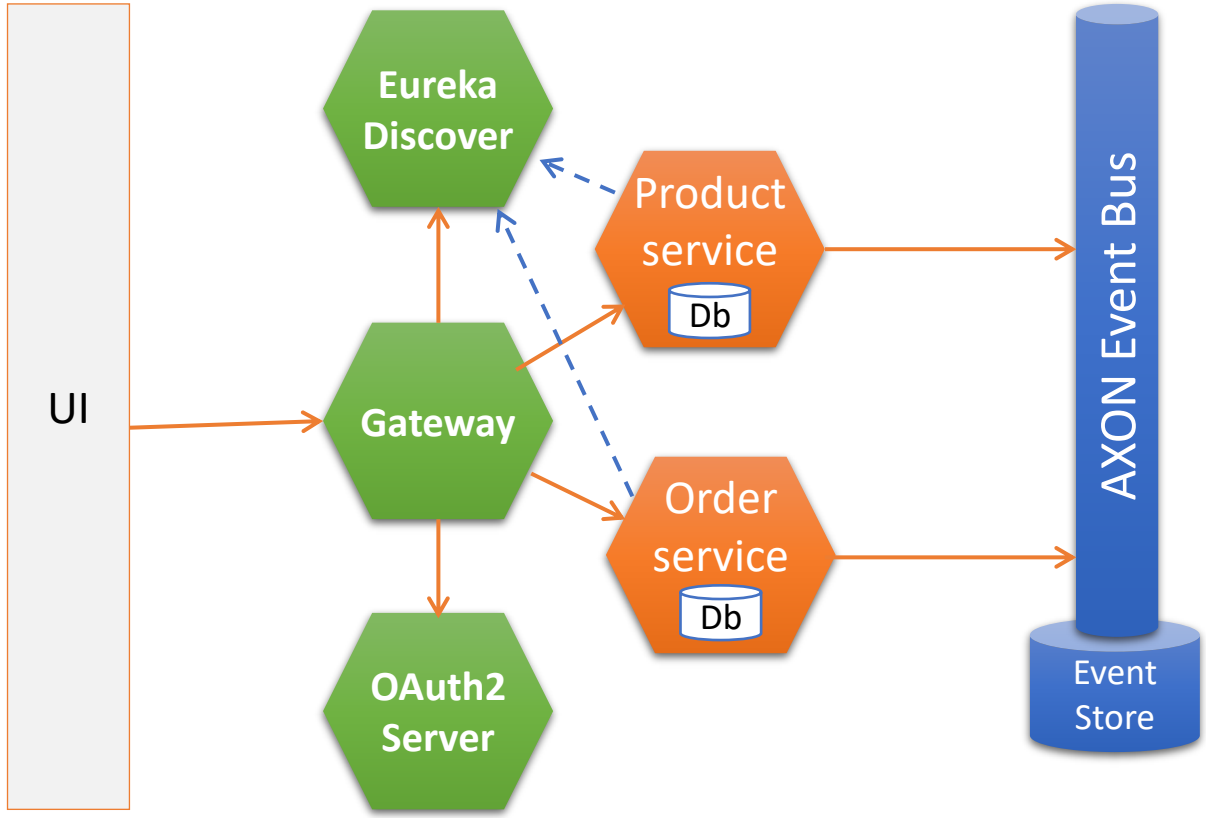
Starting Axon Server : Event Store



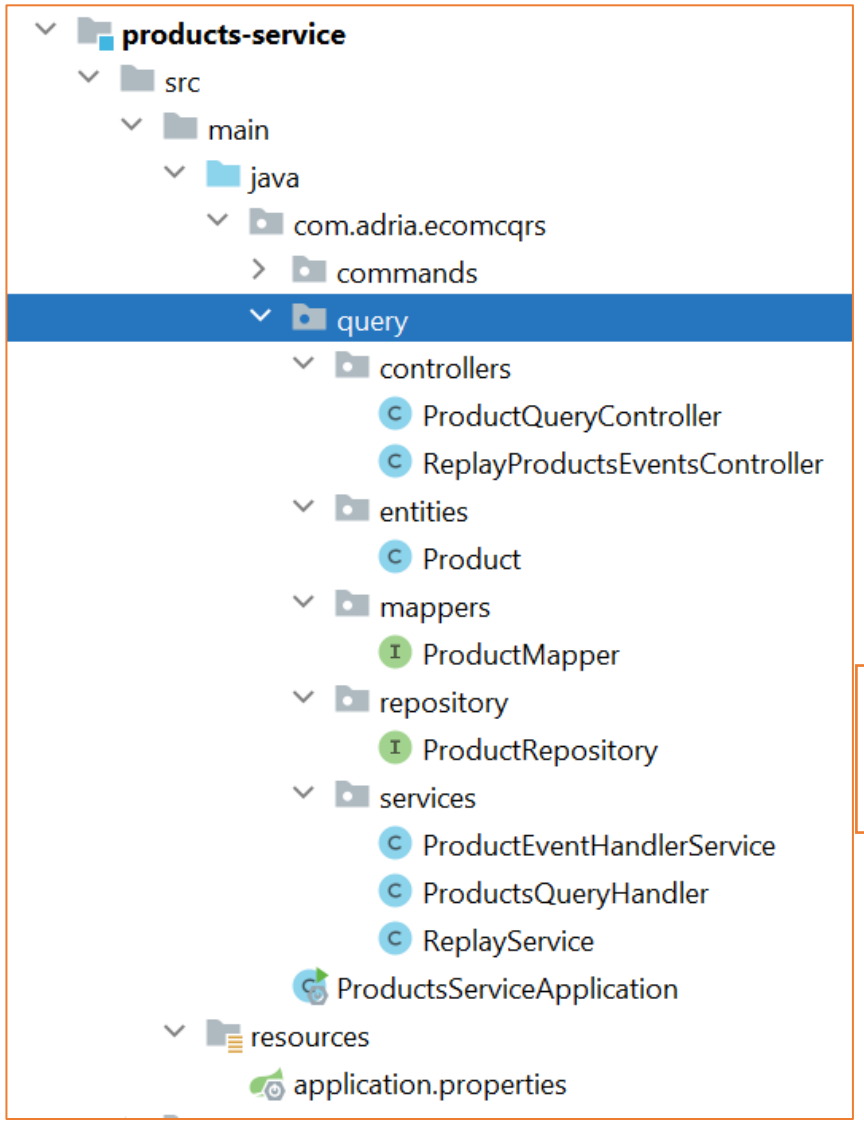
Product-service

Query Side





Module Products-service : Query Model




```java
@Entity
@Data @NoArgsConstructor @AllArgsConstructor
public class Product {
   @Id
   private String productId;
   private String name;
   private Double price;
   private Double stock;
}
```

```java
public interface ProductRepository extends JpaRepository<Product,String> {
}
```

```java
@Mapper(componentModel = "spring")
public interface ProductMapper {
   ProductResponseDTO from(Product product);
}
```

Module Products-service : Event Handlers

```
@Service
@AllArgsConstructor
@Slf4j
public class ProductEventHandlerService {
    private ProductRepository productRepository;
    private QueryUpdateEmitter queryUpdateEmitter;
    private ProductMapper productMapper;

    @ResetHandler
    public void resetData(){
        log.info("resetData .Delating All Products");
        productRepository.deleteAll();
    }
```

Module Products-service : Event Handlers

```
@EventHandler
  public void on (ProductCreatedEvent event){
     log.info("********* Query Side **********");
     log.info("ProductCreatedEvent occured");
     Product product=new Product(
           event.getProductId(),
           event.getName(),
           event.getPrice(),
           event.getStock()
     );
     productRepository.save(product);
     ProductResponseDTO productResponseDTO=productMapper.from(product);
     productResponseDTO.setEventType("ProductCreatedEvent");
     queryUpdateEmitter.emit(m->true,productResponseDTO);
  }
```

Module Products-service : Event Handlers

```
@EventHandler
  public void on (ProductStockProvisionedEvent event){
     log.info("********* Query Side **********");
     log.info("ProductStockProvisionedEvent occured");
     Product product=productRepository.findById(event.getProductId()).get();
     product.setStock(product.getStock()+event.getQuantity());
     productRepository.save(product);

     ProductResponseDTO productResponseDTO=productMapper.from(product);
     productResponseDTO.setEventType("ProductStockProvisionedEvent");
     queryUpdateEmitter.emit(m->true,productResponseDTO);
  }
```

Module Products-service : Event Handlers

```
@EventHandler
  public void on (ProductPriceUpdatedEvent event){
     log.info("********* Query Side **********");
     log.info("ProductPriceUpdatedEvent occured");
     Product product=productRepository.findById(event.getProductId()).get();
     product.setPrice(event.getPrice());
     productRepository.save(product);

     ProductResponseDTO productResponseDTO=productMapper.from(product);
     productResponseDTO.setEventType("ProductPriceUpdatedEvent");
     queryUpdateEmitter.emit(m->true,productResponseDTO);
  }
```

Module Products-service : Event Handlers

```
@EventHandler
  public void on (ProductOrderedEvent event){
     log.info("********* Query Side **********");
     log.info("ProductOrderedEvent occured");
     Product product=productRepository.findById(event.getProductId()).get();
     product.setStock(product.getStock()-event.getQuantity());
     productRepository.save(product);

     ProductResponseDTO productResponseDTO=productMapper.from(product);
     productResponseDTO.setEventType("ProductOrderedEvent");
     queryUpdateEmitter.emit(m->true,productResponseDTO);
  }
}
```

Module Products-service : Query Handlers

```
@Service
@Slf4j
@AllArgsConstructor
public class ProductsQueryHandler {
    private ProductRepository productRepository;
    private ProductMapper productMapper;

    @QueryHandler
    public List<ProductResponseDTO> handle(GetAllProductsQuery query){
        List<Product> products = productRepository.findAll();
        return products.stream().map(p->productMapper.from(p)).collect(Collectors.toList());
    }
    @QueryHandler
    public ProductResponseDTO handle(GetProductByIdQuery query){
        Product product = productRepository.findById(query.getProductId()).get();
        return productMapper.from(product);
    }}
```

Module Products-service : Replay Events Service

```
@Service
@AllArgsConstructor
@Slf4j
public class ReplayService {
    private EventProcessingConfiguration eventProcessingConfiguration;

    public void reply(){
        String name="com.adria.ecomcqrs.query.services";
        eventProcessingConfiguration.eventProcessor(name, TrackingEventProcessor.class)
                .ifPresent(processor->{
                    processor.shutDown();
                    processor.resetTokens();
                    processor.start();
                });
    }
}
```

Module Products-service : Query Controller

```
//@CrossOrigin("*")
@RestController
@AllArgsConstructor
public class ProductQueryController {

    private QueryGateway queryGateway;

    @GetMapping("/query/products/all")
    public CompletableFuture<List<ProductResponseDTO>> productList(){
        return queryGateway.query(
                new GetAllProductsQuery(),
                ResponseTypes.multipleInstancesOf(ProductResponseDTO.class)
        );
    }
```

Module Products-service : Query Controller

```
    @GetMapping("/query/products/{productId}")
    public CompletableFuture<ProductResponseDTO> handleRequest(@PathVariable String
productId){
        return queryGateway.query(
                new GetProductByIdQuery(productId),
                ResponseTypes.instanceOf(ProductResponseDTO.class)
        );
    }
```

Module Products-service : Query Controller

```
@GetMapping(path = "/query/products/watch",produces =
MediaType.TEXT_EVENT_STREAM_VALUE)
   public Flux<ProductResponseDTO> watch(){
      SubscriptionQueryResult<List<ProductResponseDTO>, ProductResponseDTO> result =
queryGateway.subscriptionQuery(new GetAllProductsQuery(),
            ResponseTypes.multipleInstancesOf(ProductResponseDTO.class),
            ResponseTypes.instanceOf(ProductResponseDTO.class)
      );
      return result.updates();
   }
```

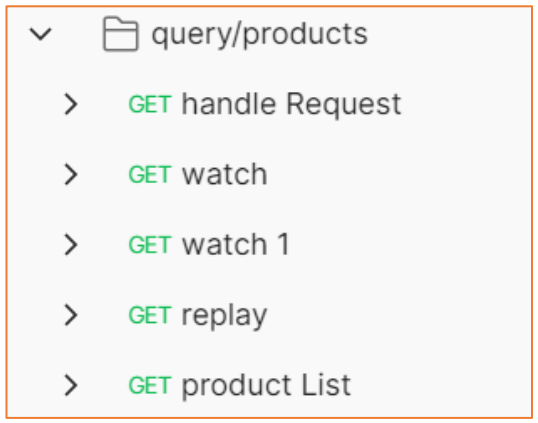
Module Products-service : Query Controller

```
@GetMapping(path = "/query/products/watchProduct/{productId}",produces =
MediaType.TEXT_EVENT_STREAM_VALUE)
   public Flux<ProductResponseDTO> watch(@PathVariable String productId){
      SubscriptionQueryResult<ProductResponseDTO, ProductResponseDTO> result =
queryGateway.subscriptionQuery(new GetProductByIdQuery(productId),
            ResponseTypes.instanceOf(ProductResponseDTO.class),
            ResponseTypes.instanceOf(ProductResponseDTO.class)
      );
      return result.initialResult().concatWith(result.updates());
   }
}
```

Module Products-service : Replay Events Controller

```
@RestController
@AllArgsConstructor
public class ReplayProductsEventsController {
    private final ReplayService replayService;

    @GetMapping("/query/products/replayEvents")
    public String replay(){
        replayService.reply();
        return "Replaying events";
    }
}
```

Module Products-service : Query Side Test : Get All Products



GET {{baseUrl}}/query/products/all Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

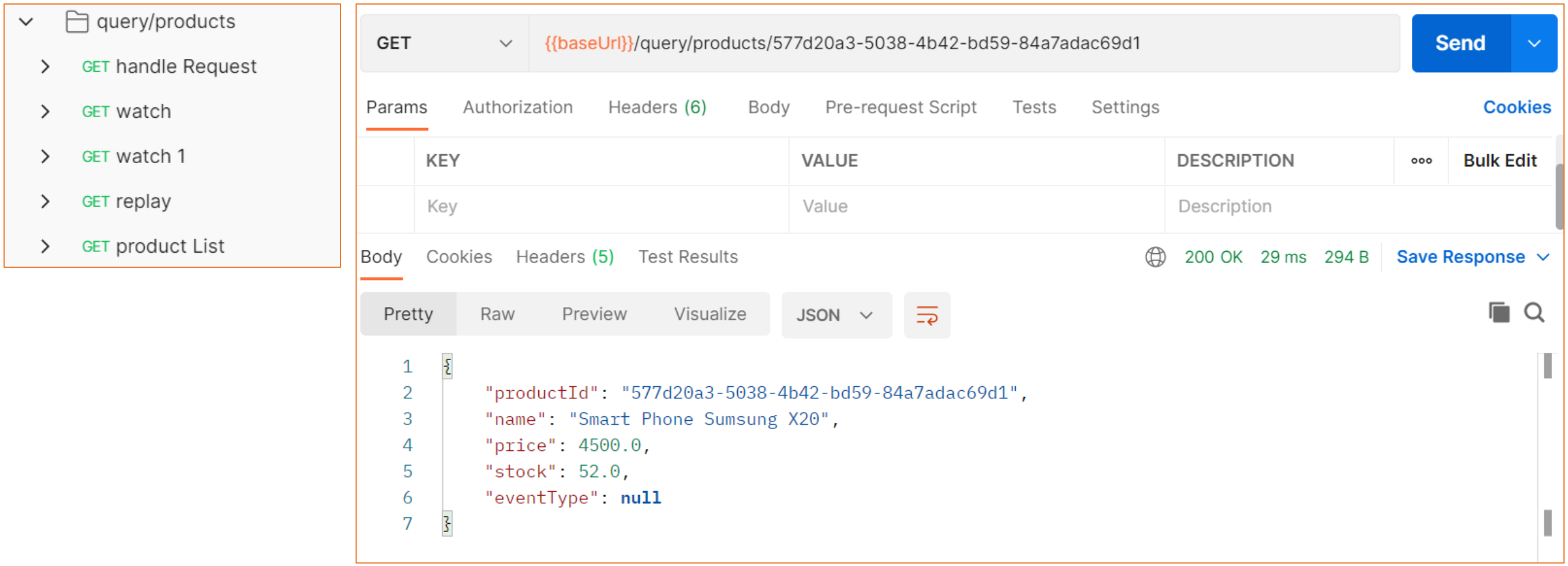
KEY	VALUE	DESCRIPTION	ooo	Bulk Edit
Key	Value	Description		

Body Cookies Headers (5) Test Results 200 OK 711 ms 527 B Save Response

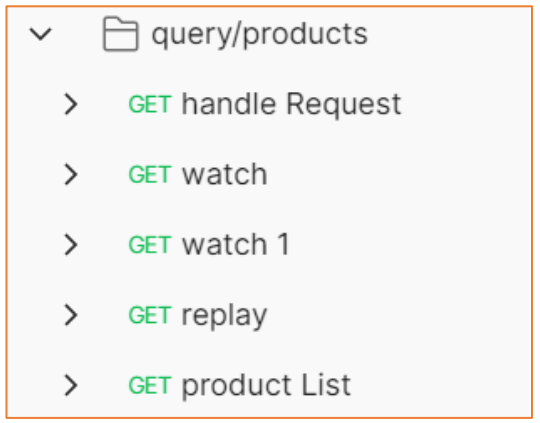
Pretty Raw Preview Visualize JSON


```
[
    {
        "productId": "a3f40583-0126-4d22-b73b-cce1ba5d571b",
        "name": "Computer",
        "price": 3200.0,
        "stock": 89.0,
        "eventType": null
    },
    {
        "productId": "0fba7fe6-30a5-4758-bd44-c83a6b8ae917",
        "name": "Printer",
        "price": 3200.0,
```

Module Products-service : Query Side Test : Get Product



Module Products-service : Query Side Test : Replay Events



GET {{baseUrl}}/query/products/replayEvents Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

KEY	VALUE	DESCRIPTION	ooo	Bulk Edit
Key	Value	Description		

Body Cookies Headers (5) Test Results 200 OK 1026 ms 180 B Save Response

Pretty Raw Preview Visualize Text

```
1  Replaying events
```

```
: resetData .Delating All Products
: Worker assigned to segment Segment[0/0] for processing
: Using current Thread for last segment worker: TrackingSegmentWorker{processor=com.adria.ecomcqrs.query.services, segment=Segment[0/0]}
: Fetched token: ReplayToken{currentToken=null, tokenAtReset=IndexTrackingToken{globalIndex=11}} for segment: Segment[0/0]
: ********* Query Side **********
: ProductCreatedEvent occured
: ********* Query Side **********
: ProductCreatedEvent occured
: ********* Query Side **********
: ProductOrderedEvent occured
: ********* Query Side **********
: ProductOrderedEvent occured
: ********* Query Side **********
: ProductCreatedEvent occured
: ********* Query Side **********
: ProductPriceUpdatedEvent occured
: ********* Query Side **********
: ProductStockProvisionedEvent occured
```

Module Products-service : Query Side Test : Replay Events

AxonDashboard: query

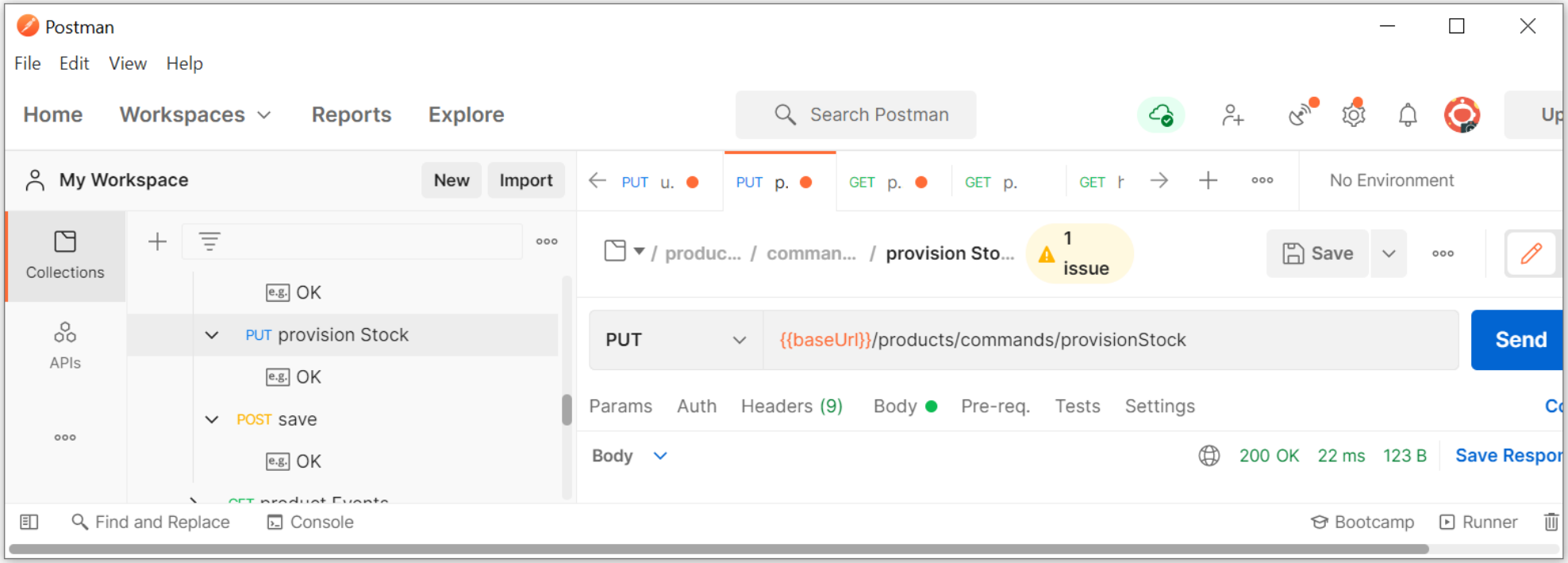
localhost:8094/query/products/w

localhost:8094/query/products/watch

Applications

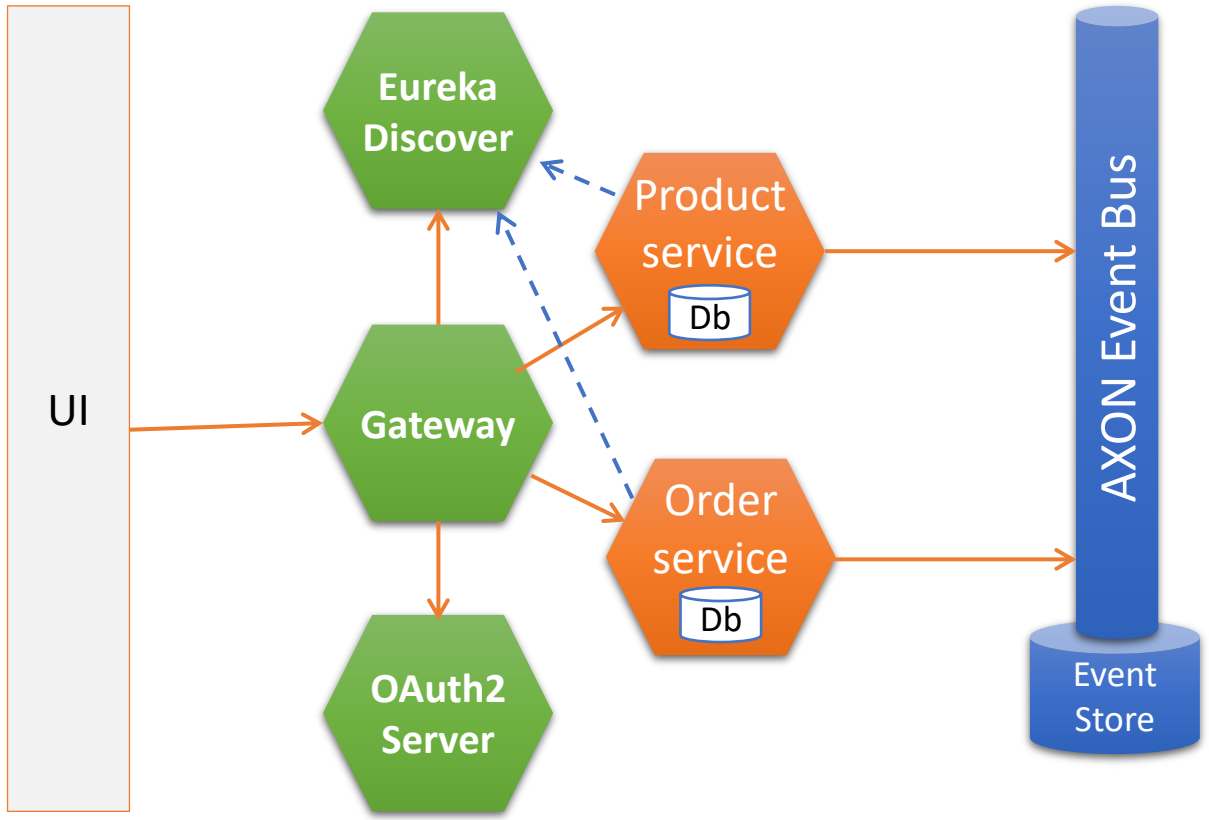
Liste de lecture

```
data:{"productId":"577d20a3-5038-4b42-bd59-84a7adac69d1","name":"Smart Phone Sumsung X20","price":4500.0,"stock":92.0,"eventType":"ProductPriceUpdatedEvent"}

data:{"productId":"577d20a3-5038-4b42-bd59-84a7adac69d1","name":"Smart Phone Sumsung X20","price":4500.0,"stock":132.0,"eventType":"ProductStockProvisionedEvent"}

data:{"productId":"577d20a3-5038-4b42-bd59-84a7adac69d1","name":"Smart Phone Sumsung X20","price":4500.0,"stock":172.0,"eventType":"ProductStockProvisionedEvent"}
```

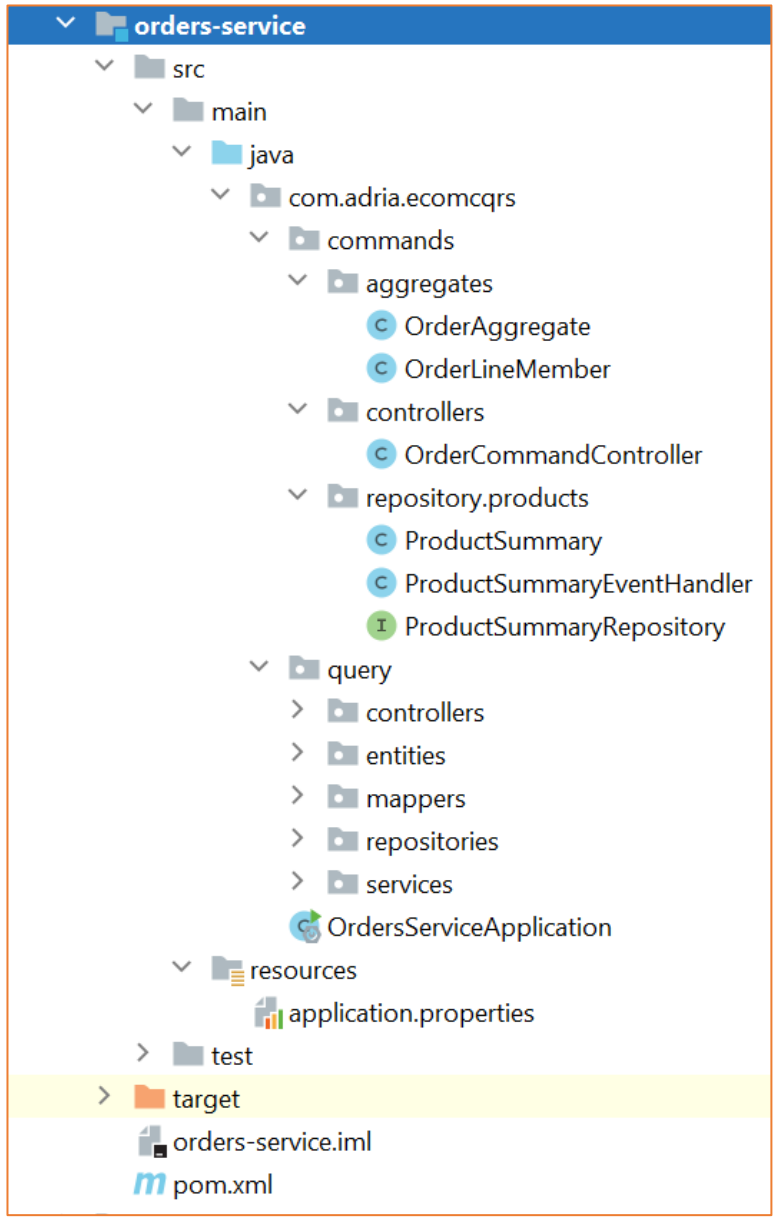


Order-service

Commands Side



Module Order-service




```
<properties>
   <maven.compiler.source>8</maven.compiler.source>
   <maven.compiler.target>8</maven.compiler.target>
   <spring-cloud.version>2020.0.3</spring-cloud.version>
</properties>
<packaging>jar</packaging>
```

Module Order-service : Maven Dependencies

```
<dependencies>
   <dependency>
      <groupId>com.adria</groupId>
      <artifactId>core-api</artifactId>
      <version>0.0.1-SNAPSHOT</version>
      <scope>compile</scope>
   </dependency>
   <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
   </dependency>
   <dependency>
      <groupId>com.google.guava</groupId>
      <artifactId>guava</artifactId>
      <version>24.0-jre</version>
   </dependency>
</dependencies>
```

Module Order-service : Maven Dependencies

```
<dependencyManagement>
   <dependencies>
      <dependency>
         <groupId>org.springframework.cloud</groupId>
         <artifactId>spring-cloud-dependencies</artifactId>
         <version>${spring-cloud.version}</version>
         <type>pom</type>
         <scope>import</scope>
      </dependency>
   </dependencies>
</dependencyManagement>
```

Module Order-service : application.properties

```
server.port=8093
spring.application.name=order-service
spring.datasource.url=jdbc:h2:mem:orders-db
#spring.datasource.url=jdbc:h2:file:./ordersdb4;PASSWORD=1234;USER=user1
axon.axonserver.enabled=true
spring.h2.console.enabled=true
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.H2Dialect
eureka.instance.prefer-ip-address=true
spring.cloud.discovery.enabled=false
#axon.eventhandling.processors.customerRegistry.mode=subscribing
```

Module Order-service : OrderAggregate

```
//@Entity
@Aggregate
@Slf4j
public class OrderAggregate {
    //@Id
    @AggregateIdentifier
    protected String orderId;
    protected Date date;
    @Enumerated(EnumType.STRING)
    protected OrderStatus status;
    //@OneToMany(fetch = FetchType.EAGER,cascade = CascadeType.ALL)
    //@JoinColumn(name = "orderId")
    @AggregateMember(eventForwardingMode = ForwardMatchingInstances.class)
    protected List<OrderLineMember> orderLines=new ArrayList<>();

    public OrderAggregate() {
    }
```

Module Order-service : OrderAggregate

```
@CommandHandler
  public OrderAggregate(CreateOrderCommand command) {
    log.info("CreateOrderCommand received ....");
    AggregateLifecycle.apply(new OrderCreatedEvent(
          UUID.randomUUID().toString(),
          new Date(),
          command.getStatus()
    ));
  }
  //@EventHandler
  @EventSourcingHandler
  public void on(OrderCreatedEvent event){
    log.info("OrderCreatedEvent Occured ....");
    this.orderId=event.getOrderId();
    this.date=event.getDate();
    this.status=event.getStatus();
  }
```

Module Order-service : OrderAggregate

```
@CommandHandler
   public void handle(AddOrderLineCommand command, ProductSummaryRepository
productSummaryRepository) {
      if(this.status.equals(OrderStatus.CONFIRMED)) {
         throw new OrderAlreadyConfirmedException(String.format("Order %s Already confirmed",this.orderId));
      }
      ProductSummary
productSummary=productSummaryRepository.findById(command.getProductId()).orElse(null);
      if(productSummary==null) throw new ProductNotFoundException(String.format("Product %s
NotFound",command.getProductId()));
      OrderLineMember orderLine=orderLines.stream().filter(ol-
>ol.getProductId().equals(command.getProductId())).findFirst().orElse(null);
      if(orderLine!=null){
         throw new OrderLineAlreadyExistException(String.format("Order Line %s Already exists
",command.getProductId()));
      }
      AggregateLifecycle.apply(new OrderLineAddedEvent(
            command.getOrderId(),   command.getProductId(),   command.getQuantity(), command.getPrice()
      ));
   }
```

Module Order-service : OrderAggregate

```
//@EventHandler
   @EventSourcingHandler
   public void on(OrderLineAddedEvent event){
      log.info("OrderLineAddedEvent Occured ....");
      orderLines.add(new OrderLineMember(
               event.getProductId(),
               event.getQuantity(),
               event.getPrice(),
            this
      ));
   }
```

Module Order-service : OrderAggregate

```
@CommandHandler
  public void handle(RemoveOrderLineCommand command) {
     log.info("RemoveOrderLineCommand Received ....");
     if(this.status.equals(OrderStatus.CONFIRMED)) {
        throw new OrderAlreadyConfirmedException(String.format("Order %s Already confirmed",this.orderId));
     }
     OrderLineMember orderLine=orderLines.stream().filter(ol-
>ol.getProductId().equals(command.getProductId())).findFirst().orElse(null);
     if(orderLine==null){
        throw new OrderLineNotExistException(String.format("Order Line %s Not Exist in the
Order",command.getProductId()));
     }
     AggregateLifecycle.apply(new OrderLineRemovedEvent(
           command.getOrderId(),
           command.getProductId()
     ));
  }
```

Module Order-service : OrderAggregate

```
//@EventHandler
   @EventSourcingHandler
   public void on(OrderLineRemovedEvent event){
      log.info("OrderLineRemovedEvent Occured ....");
      OrderLineMember orderLine=orderLines.stream().filter(ol-
>ol.getProductId().equals(event.getProductId())).findFirst().orElse(null);
      orderLines.remove(orderLine);
   }
```

Module Order-service : OrderAggregate

```
@CommandHandler
  public void handle(ConfirmOrderCommand command, ProductSummaryRepository productSummaryRepository) {
     log.info("ConfirmOrderCommand Received ....");
     if(this.status.equals(OrderStatus.CONFIRMED)) {
        throw new OrderAlreadyConfirmedException(String.format("Order %s Already confirmed",this.orderId));
     }
     if(this.orderLines.size()==0) {
        throw new OrderEmptyException(String.format("Order %s Empty",this.orderId));
     }
     orderLines.forEach(ol->{
        ProductSummary productSummary=productSummaryRepository.findById(ol.getProductId()).get();

        if(productSummary.getStock()-ol.getQuantity()<0){
          throw new ProductQuantityNotAvailableException(
                String.format("Quantity %s of Product %s ,  is Not AvailableException => Stock = %s",
                      ol.getQuantity(),ol.getProductId(), productSummary.getStock())
          ) ;
        }
     });
     AggregateLifecycle.apply(new OrderConfirmedEvent(
           command.getOrderId(),      command.getStatus()
     ));
  }
```

Module Order-service : OrderAggregate

```
//@EventHandler
   @EventSourcingHandler
   public void on(OrderConfirmedEvent event){
      log.info("OrderConfirmedEvent Occured ....");
      this.status=event.getStatus();
      orderLines.forEach(ol->{
         AggregateLifecycle.apply(new ProductOrderedEvent(
               this.orderId,
               ol.getProductId(),
               ol.getQuantity(),
               ol.getPrice()
         ));
      });
   }
}
```

Module Order-service : OrderLine Aggregate Member

```
@Data @NoArgsConstructor
@Slf4j
//@Entity
public class OrderLineMember {
   //@Id @GeneratedValue(strategy = GenerationType.IDENTITY)
   //private Long id;
    @EntityId
    private String productId;
    private double quantity;
    private double price;
    private OrderAggregate orderAggregate;

    public OrderLineMember(String productId, double quantity, double price, OrderAggregate
orderAggregate) {
        this.productId = productId;    this.quantity = quantity;    this.price = price;
        this.orderAggregate=orderAggregate;
    }
```

Module Order-service : OrderLine Aggregate Member

```
   @CommandHandler
    public void handle(UpdateOrderLineCommand command){
        log.info(String.format("UpdateOrderLineCommand received  %s
",command.getProductId()));
        if(this.orderAggregate.status.equals(OrderStatus.CONFIRMED)) {
            throw new OrderAlreadyConfirmedException(String.format("Order %s Already
confirmed",this.orderAggregate.orderId));
        }
        AggregateLifecycle.apply(new OrderLineUpdatedEvent(
                command.getOrderId(),command.getProductId(),
                command.getQuantity(), command.getPrice()
        ));
    }
```

Module Order-service : OrderLine Aggregate Member

```
//@EventHandler
   @EventSourcingHandler
   private void on(OrderLineUpdatedEvent event){
      log.info(String.format("OrderLineUpdatedEvent received  %s ",event.getProductId()));
      this.quantity= event.getQuantity();
      this.price= event.getPrice();
   }
}
```

Module Order-service : Product Projection Model For Order Aggregate

```
@Data @AllArgsConstructor @NoArgsConstructor
 @Entity
public class ProductSummary {
    @Id
    private String productId;
    private Double price;
    private Double stock;
}
```

```
public interface ProductSummaryRepository extends JpaRepository<ProductSummary,String> {
}
```

Module Order-service : Product Projection Model For Order Aggregate

```
@Service
@AllArgsConstructor
@Slf4j
public class ProductSummaryEventHandler {
    private ProductSummaryRepository productSummaryRepository;
    @EventHandler
    public void on (ProductCreatedEvent event){
        log.info("############### Command Side ########################");
        log.info("ProductCreatedEvent received");
        productSummaryRepository.save(new ProductSummary(
            event.getProductId(),
            event.getPrice(),
            event.getStock()
        ));
    }
```

Module Order-service : Product Projection Model For Order Aggregate

```
@EventHandler
    public void on (ProductPriceUpdatedEvent event){
        log.info("############### Command Side ########################");
        log.info("ProductPriceUpdatedEvent received");
        ProductSummary
productSummary=productSummaryRepository.findById(event.getProductId()).get();
        productSummary.setPrice(event.getPrice());
        productSummaryRepository.save(productSummary);
    }
```

Module Order-service : Product Projection Model For Order Aggregate

```
@EventHandler
    public void on (ProductStockProvisionedEvent event){
        log.info("############### Command Side ########################");
        log.info("ProductStockProvisionedEvent received");
        ProductSummary
productSummary=productSummaryRepository.findById(event.getProductId()).get();
        productSummary.setStock(event.getQuantity()+productSummary.getStock());
        productSummaryRepository.save(productSummary);
    }
```

Module Order-service : Product Projection Model For Order Aggregate

```
@EventHandler
  public void on (ProductOrderedEvent event){
     log.info("############### Command Side ########################");
     log.info("ProductOrderedEvent received");
     ProductSummary
productSummary=productSummaryRepository.findById(event.getProductId()).get();
     productSummary.setStock(productSummary.getStock()-event.getQuantity());
     productSummaryRepository.save(productSummary);
  }

}
```

Module Order-service : OrderCommandController

```
//@CrossOrigin("*")
@RestController
@RequestMapping("/commands/orders")
@AllArgsConstructor
@Slf4j
public class OrderCommandController {
    private final CommandGateway commandGateway;
    private final EventStore eventStore;
    @PostMapping("/create")
    public CompletableFuture<String> createOrder(){
        return commandGateway.send(new CreateOrderCommand(
                OrderStatus.CREATED
        ));
    }
```

Module Order-service : OrderCommandController

```
@PutMapping ("/addOrderLine")
   public CompletableFuture<String> addItem(@RequestBody AddOrderLineRequestDTO
request){
       return commandGateway.send(new AddOrderLineCommand(
               request.getOrderId(),
               request.getProductId(),
               request.getQuantity(),
               request.getPrice()
       ));
   }
```

Module Order-service : OrderCommandController

```
@PutMapping("/removeOrderLine")
   public CompletableFuture<String> removeItem(@RequestBody
RemoveOrderLineRequestDTO request){
      return commandGateway.send(new RemoveOrderLineCommand(
            request.getOrderId(),
            request.getProductId()
      ));
   }
```

Module Order-service : OrderCommandController

```
@PutMapping ("/updateOrderLine")
   public CompletableFuture<String> updateItem(@RequestBody
UpdateOrderLineRequestDTO request){
      return commandGateway.send(new UpdateOrderLineCommand(
            request.getOrderId(),
            request.getProductId(),
            request.getQuantity(),
            request.getPrice()
      ));
   }
```

Module Order-service : OrderCommandController

```
 @PutMapping ("/confirm")
   public CompletableFuture<String> confirmOrder(@RequestBody
ConfirmOrderRequestDTO request){
      return commandGateway.send(new ConfirmOrderCommand(
            request.getOrderId(), OrderStatus.CONFIRMED
      ));
   }
   @GetMapping("/{orderId}/events")
   public Stream orderEvents(@PathVariable String orderId){
      return eventStore.readEvents(orderId).asStream();
   }

}
```

Module Order-service : OrdersServiceApplication

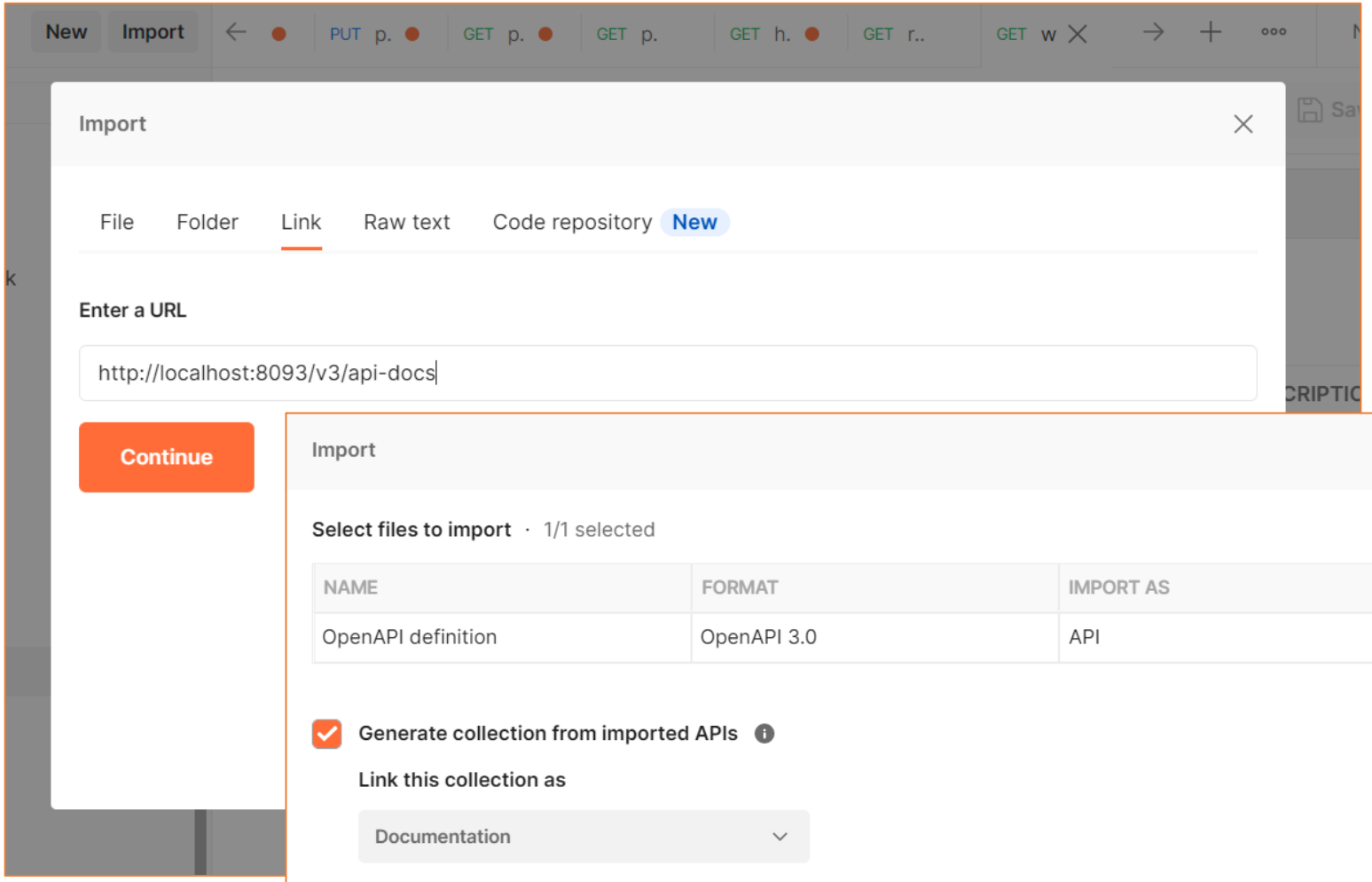
```
@SpringBootApplication
public class OrdersServiceApplication {
    public static void main(String[] args) {
        SpringApplication.run(OrdersServiceApplication.class);
    }

    //@Bean
    public Repository<OrderAggregate> accountAggregateRepository(EntityManagerProvider emp,
ParameterResolverFactory prf, EventBus eb){
        return GenericJpaRepository.builder(OrderAggregate.class)
                .entityManagerProvider(emp)
                .parameterResolverFactory(prf)
                .eventBus(eb)
                .build();
    }
}
```

Module Order-service : Command Side Test

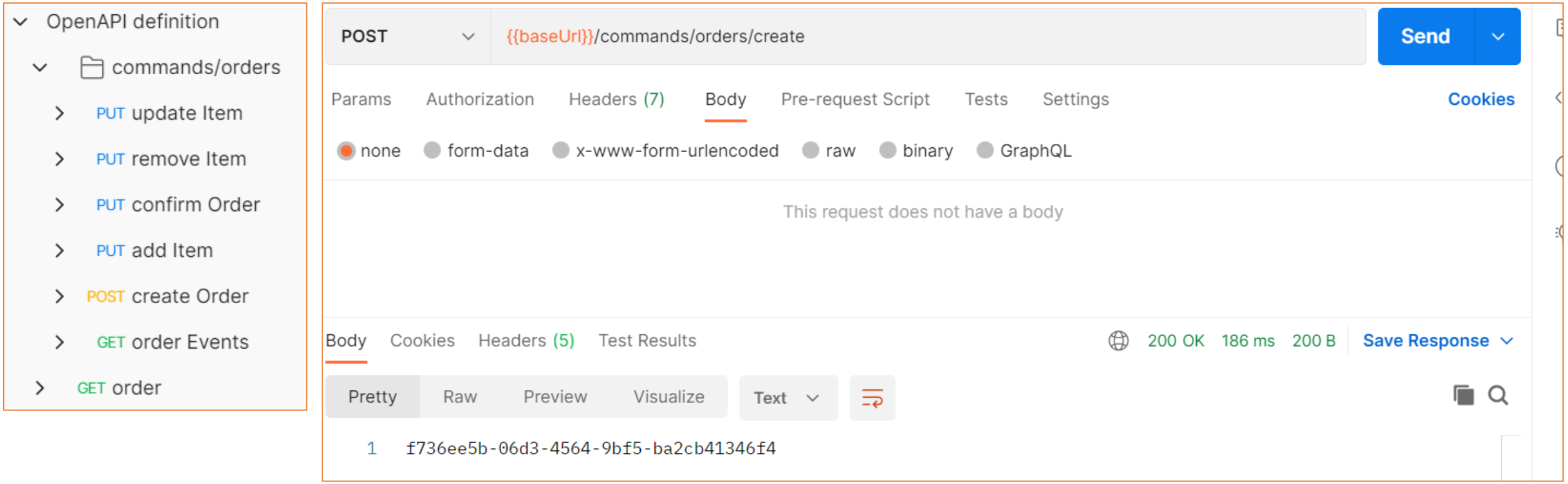
localhost:8093/v3/api-docs

Applications

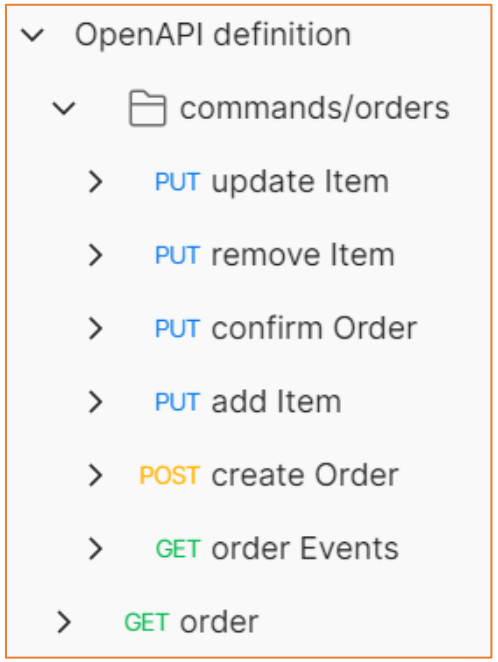

```
{
    "openapi": "3.0.1",
    "info": {
        "title": "OpenAPI definition",
        "version": "v0"
    },
    "servers": [
        {
            "url": "http://localhost:8093",
            "description": "Generated server url"
        }
    ],
    "paths": {
        "/commands/orders/updateOrderLine": {
            "put": {
                "tags": [
                    "order-command-controller"
                ],
                "operationId": "updateItem",
                "requestBody": {
                    "content": {
                        "application/json": {
                            "schema": {
                                "$ref": "#/components
                            }
                        }
                    },
                    "required": true
                },
                "responses": {
```

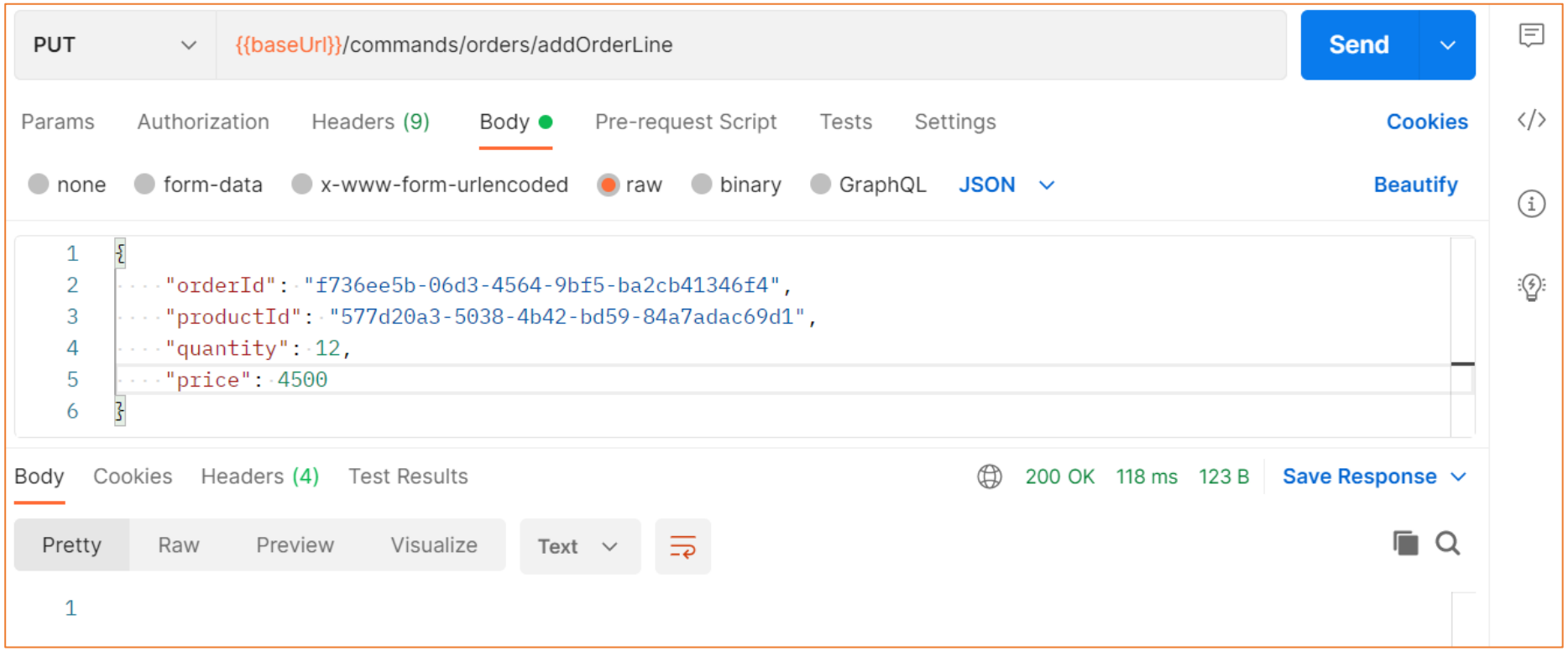



Module Order-service : Command Side Test => New Order

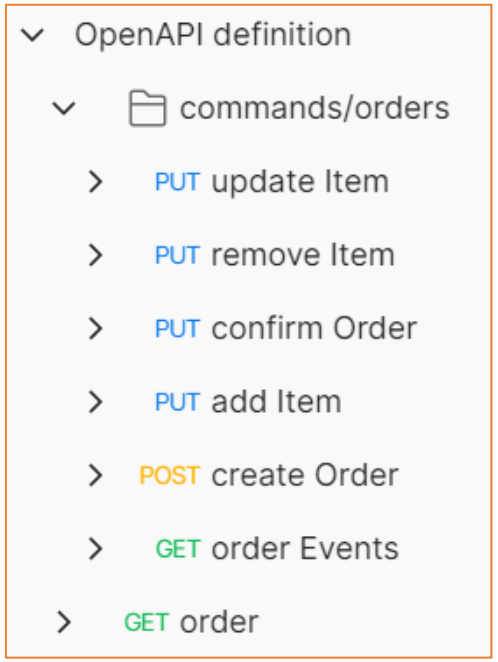


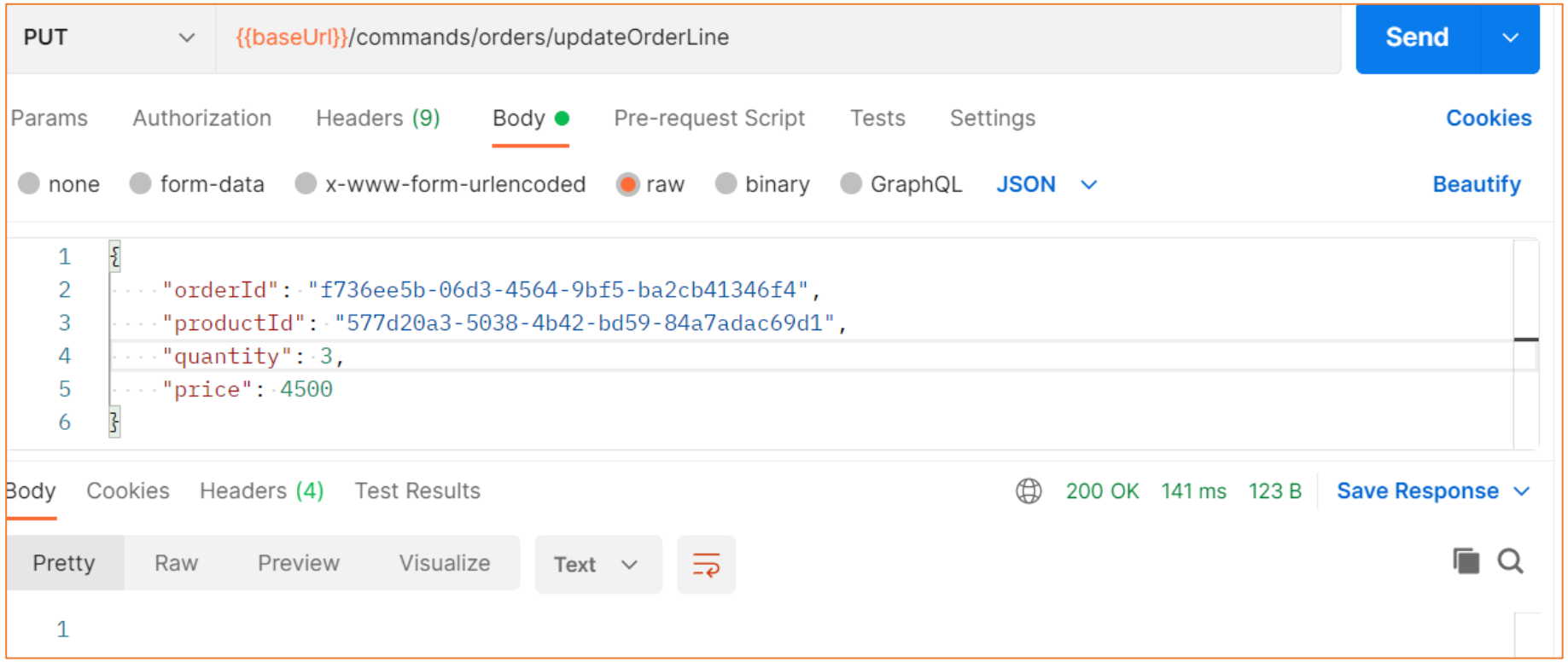
Module Order-service : Command Side Test => Add Order Item



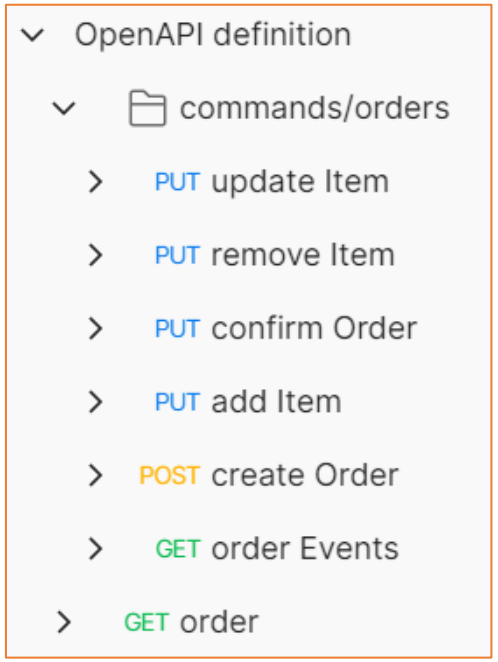


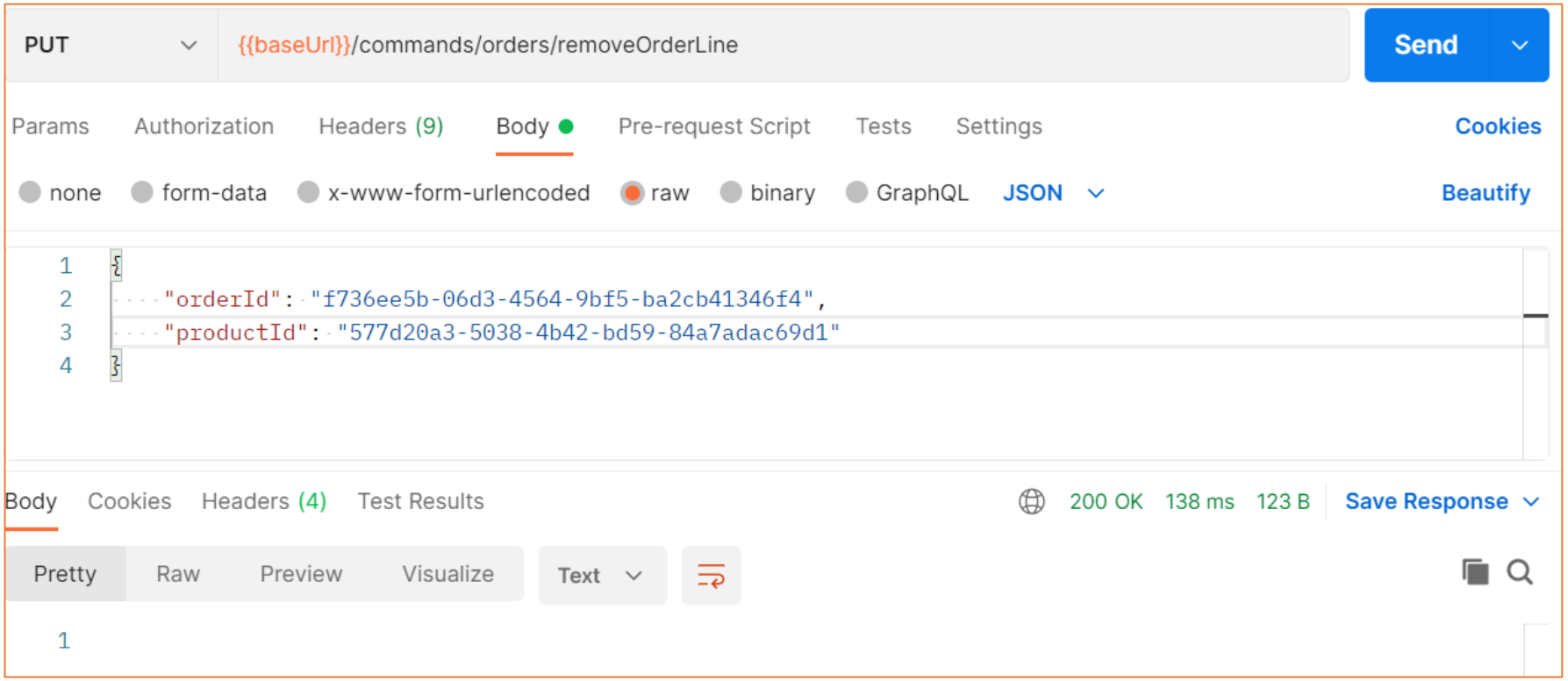
Module Order-service : Command Side Test => Update Order Item



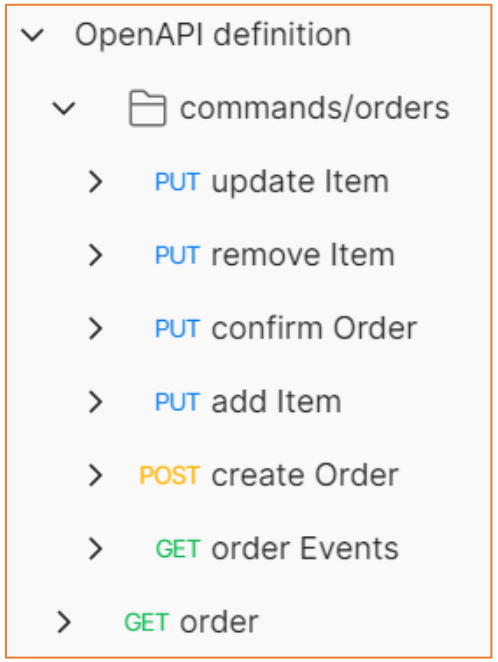


Module Order-service : Command Side Test => RemoveOrder Item





Module Order-service : Command Side Test => Order Events



GET {{baseUrl}}/commands/orders/f736ee5b-06d3-4564-9bf5-ba2cb41346f4/events Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

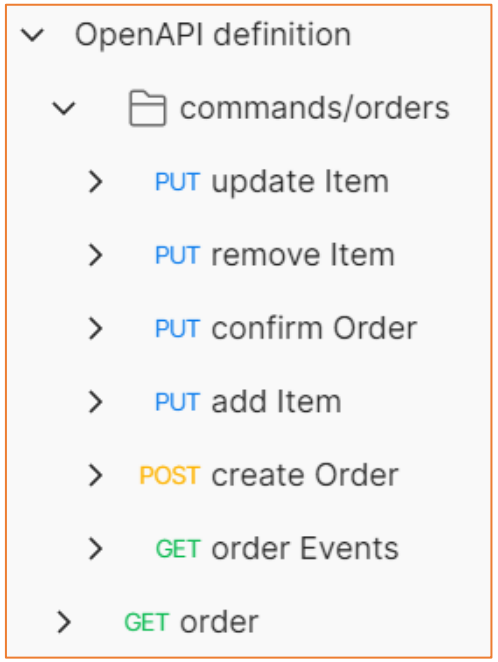
KEY	VALUE	DESCRIPTION	ooo	Bulk Edit

Body Cookies Headers (5) Test Results 200 OK 18 ms 2.68 KB Save Response

Pretty Raw Preview Visualize JSON

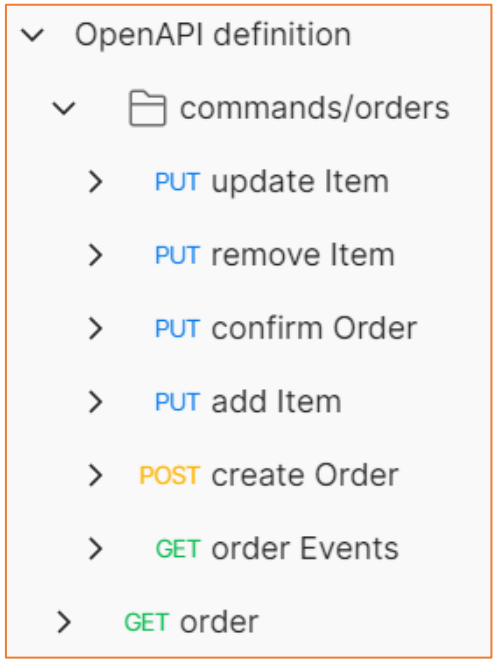

```
[
    {
        "type": "OrderAggregate",
        "aggregateIdentifier": "f736ee5b-06d3-4564-9bf5-ba2cb41346f4",
        "sequenceNumber": 0,
        "identifier": "60ce58ee-d689-40a8-84c5-ac8019520c8c",
        "timestamp": "2021-08-26T14:43:55.045Z",
        "payload": {
            "orderId": "f736ee5b-06d3-4564-9bf5-ba2cb41346f4",
            "date": "2021-08-26T14:43:55.045+00:00",
            "status": "CREATED"
        },
        "metaData": {
```

Module Order-service : Command Side Test => Order Events



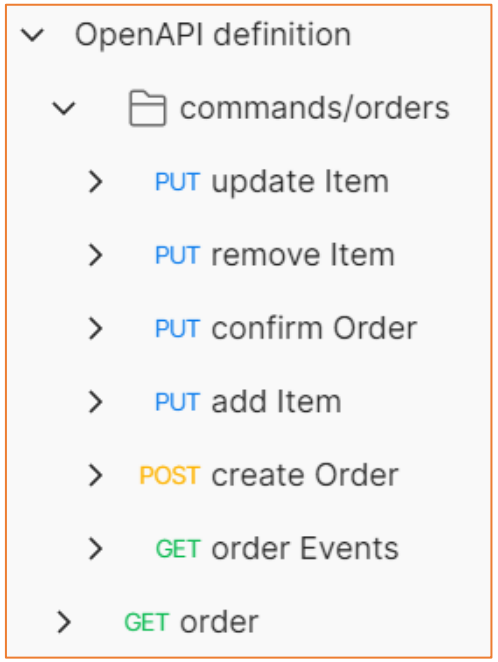

```
[
  {
    "type": "OrderAggregate",
    "aggregateIdentifier": "f736ee5b-06d3-4564-9bf5-ba2cb41346f4",
    "sequenceNumber": 0,
    "identifier": "60ce58ee-d689-40a8-84c5-ac8019520c8c",
    "timestamp": "2021-08-26T14:43:55.045Z",
    "payload": {
      "orderId": "f736ee5b-06d3-4564-9bf5-ba2cb41346f4",
      "date": "2021-08-26T14:43:55.045+00:00",
      "status": "CREATED"
    },
    "metaData": {
      "traceId": "2bb4cb3b-50c6-4b01-933f-6299687bd1d0",
      "correlationId": "2bb4cb3b-50c6-4b01-933f-6299687bd1d0"
    },
    "payloadType": "com.adria.ecomcqrs.coreapi.orders.OrderCreatedEvent"
  },
```

Module Order-service : Command Side Test => Order Events



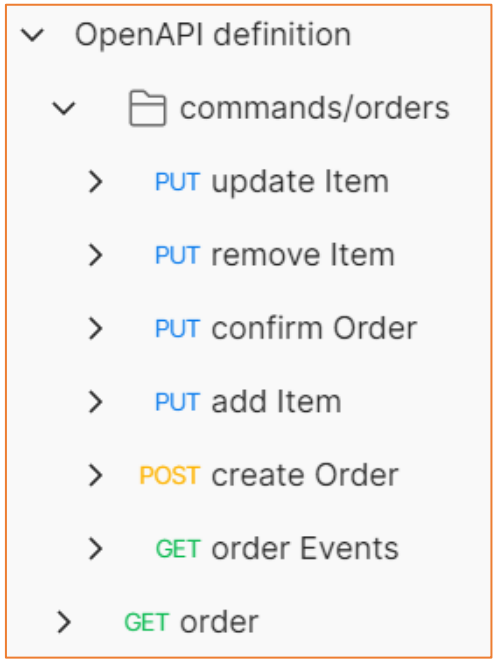

```
{
    "type": "OrderAggregate",
    "aggregateIdentifier": "f736ee5b-06d3-4564-9bf5-ba2cb41346f4",
    "sequenceNumber": 1,
    "identifier": "455f932d-2634-44bf-bb95-86896b5c0f57",
    "timestamp": "2021-08-26T14:49:28.345Z",
    "payload": {
      "orderId": "f736ee5b-06d3-4564-9bf5-ba2cb41346f4",
      "productId": "577d20a3-5038-4b42-bd59-84a7adac69d1",
      "quantity": 12.0,
      "price": 4500.0
    },
    "metaData": {
      "traceId": "d3c92019-12a3-4d83-9380-015c7444d1cd",
      "correlationId": "d3c92019-12a3-4d83-9380-015c7444d1cd"
    },
    "payloadType": "com.adria.ecomcqrs.coreapi.orders.OrderLineAddedEvent"
  },
```

Module Order-service : Command Side Test => Order Events



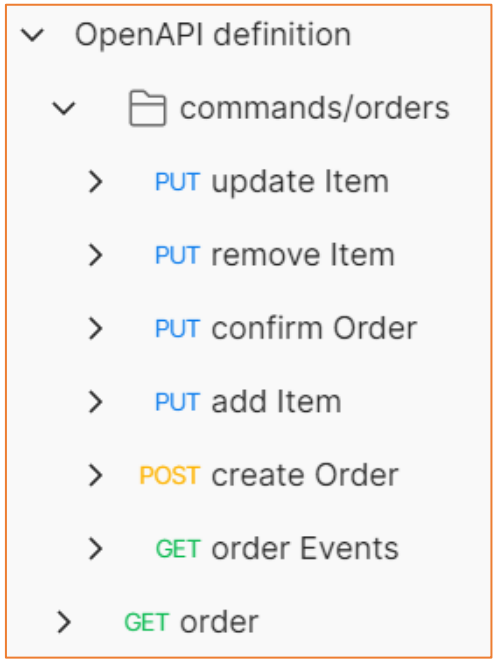

```
{
    "type": "OrderAggregate",
    "aggregateIdentifier": "f736ee5b-06d3-4564-9bf5-ba2cb41346f4",
    "sequenceNumber": 2,
    "identifier": "5093c5fe-fbe0-4a18-8f90-df05e3fe0835",
    "timestamp": "2021-08-26T14:51:47.869Z",
    "payload": {
      "orderId": "f736ee5b-06d3-4564-9bf5-ba2cb41346f4",
      "productId": "577d20a3-5038-4b42-bd59-84a7adac69d1",
      "quantity": 3.0,
      "price": 4500.0
    },
    "metaData": {
      "traceId": "45eab145-0910-4693-bb46-a64936b13ddd",
      "correlationId": "45eab145-0910-4693-bb46-a64936b13ddd"
    },
    "payloadType": "com.adria.ecomcqrs.coreapi.orders.OrderLineUpdatedEvent"
  },
```

Module Order-service : Command Side Test => Order Events



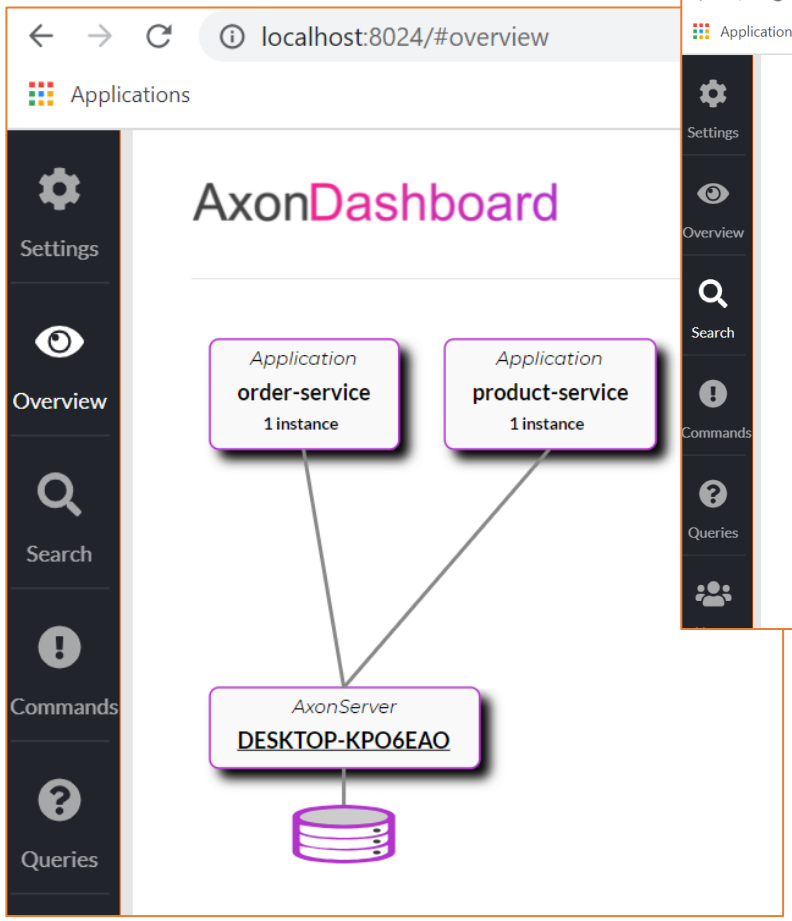

```
{
      "type": "OrderAggregate",
      "aggregateIdentifier": "f736ee5b-06d3-4564-9bf5-ba2cb41346f4",
      "sequenceNumber": 3,
      "identifier": "c26d00fb-9b9e-44d4-ad23-d7a68a455b72",
      "timestamp": "2021-08-26T14:56:08.475Z",
      "payload": {
         "orderId": "f736ee5b-06d3-4564-9bf5-ba2cb41346f4",
         "productId": "577d20a3-5038-4b42-bd59-84a7adac69d1"
      },
      "metaData": {
         "traceId": "f1d52326-43bd-4558-96d0-041359c961fc",
         "correlationId": "f1d52326-43bd-4558-96d0-041359c961fc"
      },
      "payloadType": "com.adria.ecomcqrs.coreapi.orders.OrderLineRemovedEvent"
   },
```

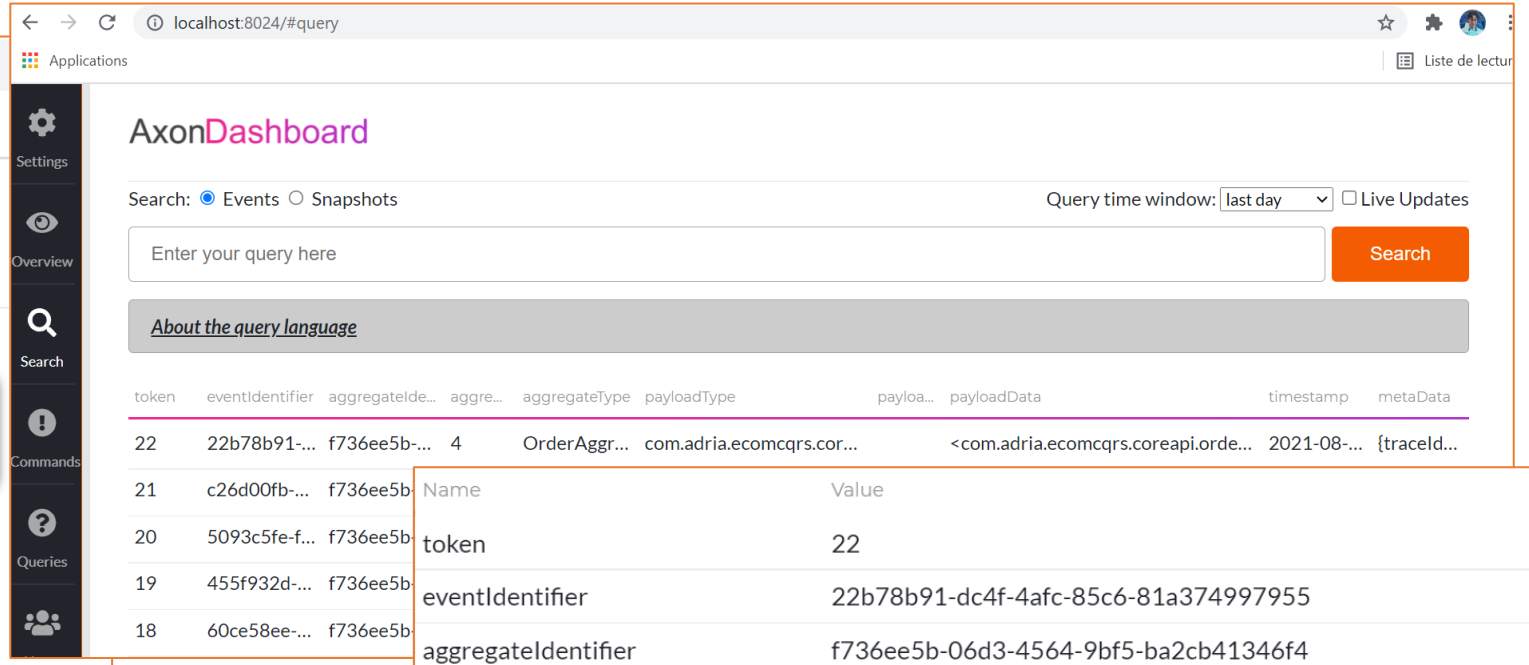
Module Order-service : Command Side Test => Order Events




```
{
    "type": "OrderAggregate",
    "aggregateIdentifier": "f736ee5b-06d3-4564-9bf5-ba2cb41346f4",
    "sequenceNumber": 4,
    "identifier": "22b78b91-dc4f-4afc-85c6-81a374997955",
    "timestamp": "2021-08-26T14:57:28.772Z",
    "payload": {
      "orderId": "f736ee5b-06d3-4564-9bf5-ba2cb41346f4",
      "productId": "577d20a3-5038-4b42-bd59-84a7adac69d1",
      "quantity": 800.0,
      "price": 4500.0
    },
    "metaData": {
      "traceId": "119d164a-5cc3-4f9a-905f-0d2ea8439646",
      "correlationId": "119d164a-5cc3-4f9a-905f-0d2ea8439646"
    },
    "payloadType": "com.adria.ecomcqrs.coreapi.orders.OrderLineAddedEvent"
  }
]
```

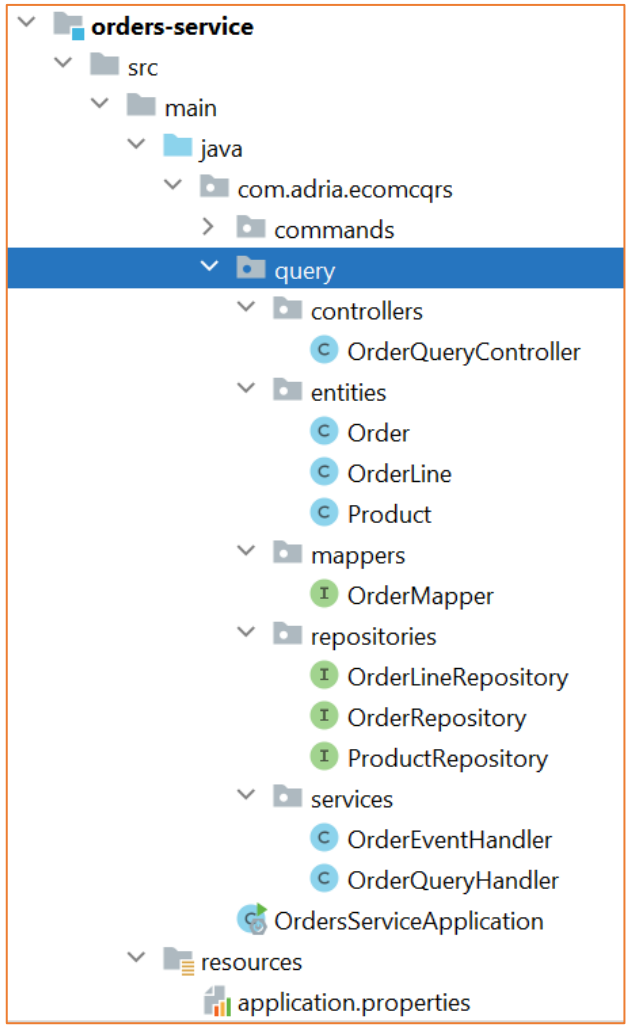
Module Order-service : Command Side Test => Order Events

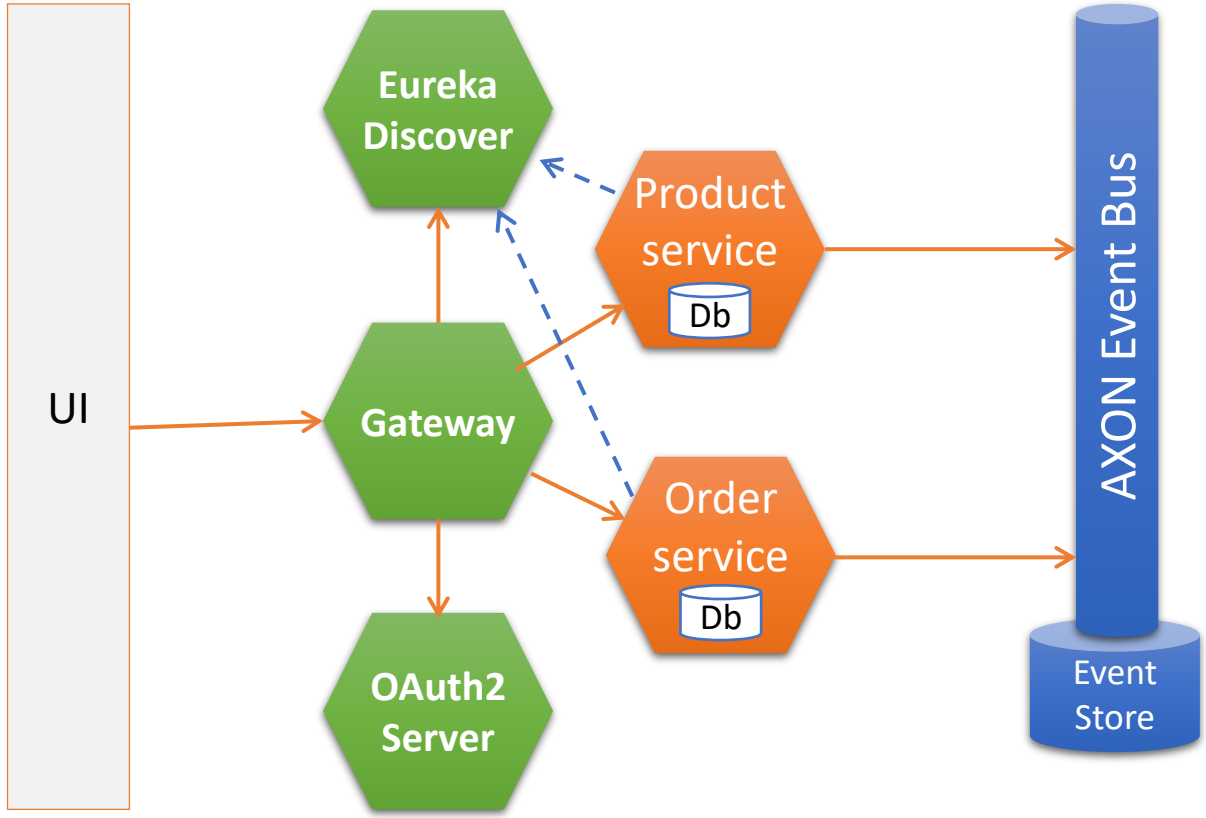




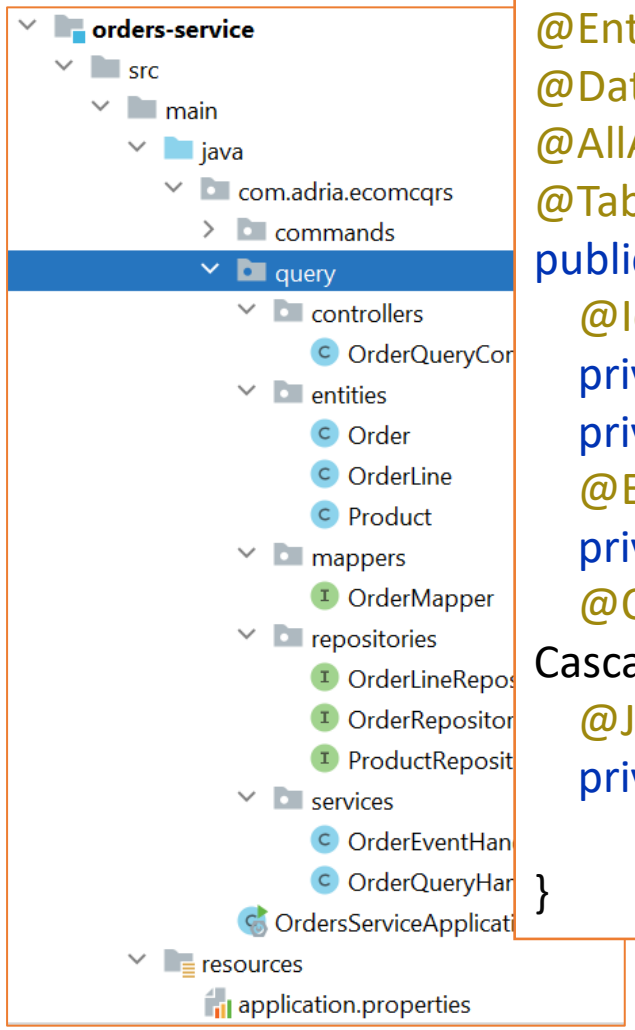
Name	Value	Copy
token	22	
eventIdentifier	22b78b91-dc4f-4afc-85c6-81a374997955	
aggregateIdentifier	f736ee5b-06d3-4564-9bf5-ba2cb41346f4	
aggregateSequenceNumber	4	
aggregateType	OrderAggregate	
payloadType	com.adria.ecomcqrs.coreapi.orders.OrderLineAddedEvent	
payloadRevision		
payloadData	<com.adria.ecomcqrs.coreapi.orders.OrderLineAddedEvent> <orderId>f736ee5b-06d3-4564-9bf5-ba2cb41346f4</orderId> <productId>577d20a3-5038-4b42-bd59-84a7adac69d1</productId> <quantity>800.0</quantity><price>4500.0</price> </com.adria.ecomcqrs.coreapi.orders.OrderLineAddedEvent>	
timestamp	2021-08-26T14:57:28.772Z	
metaData	{traceId=119d164a-5cc3-4f9a-905f-0d2ea8439646, correlationId=119d164a-5cc3-4f9a-905f-0d2ea8439646}	

Order-service Query Side





Order-service Query Side / Query Model (Jpa Entities)

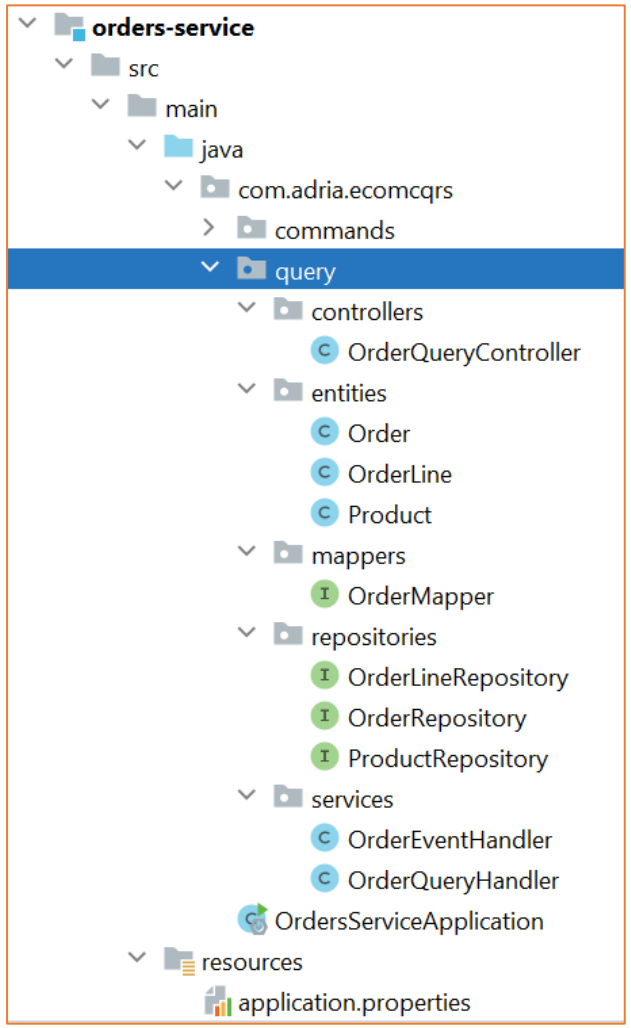



```java
@Entity
@Data @NoArgsConstructor
@AllArgsConstructor @ToString
@Table(name="T_ORDERS")
public class Order {
   @Id
   private String orderId;
   private Date date;
   @Enumerated(EnumType.STRING)
   private OrderStatus status;
   @OneToMany(fetch = FetchType.EAGER,cascade =
CascadeType.ALL)
   @JoinColumn(name="orderId")
   private List<OrderLine> orderLines=new ArrayList<>();

}
```

```java
@Entity
@Data @NoArgsConstructor
@AllArgsConstructor
public class OrderLine {
   @Id @GeneratedValue(strategy =
GenerationType.IDENTITY)
   private Long id;
   private String productId;
   private String productName;
   private Double quantity;
   private Double price;
}
```

```java
@Entity
@Data @NoArgsConstructor
@AllArgsConstructor
public class Product {
   @Id
   private String productId;
   private String name;
}
```

Order-service Query Side / Query Model (Repositories and Mappers)



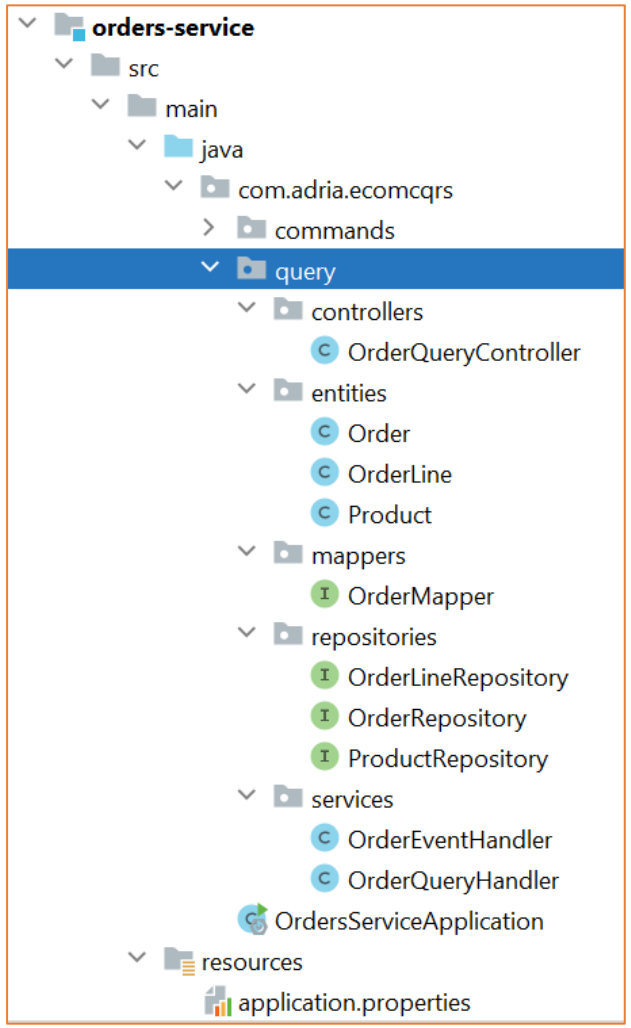

```
public interface OrderRepository extends JpaRepository<Order,String> {
}
```

```
public interface OrderLineRepository extends JpaRepository<OrderLine,Long> {
}
```

```
public interface ProductRepository extends JpaRepository<Product,String> {
}
```

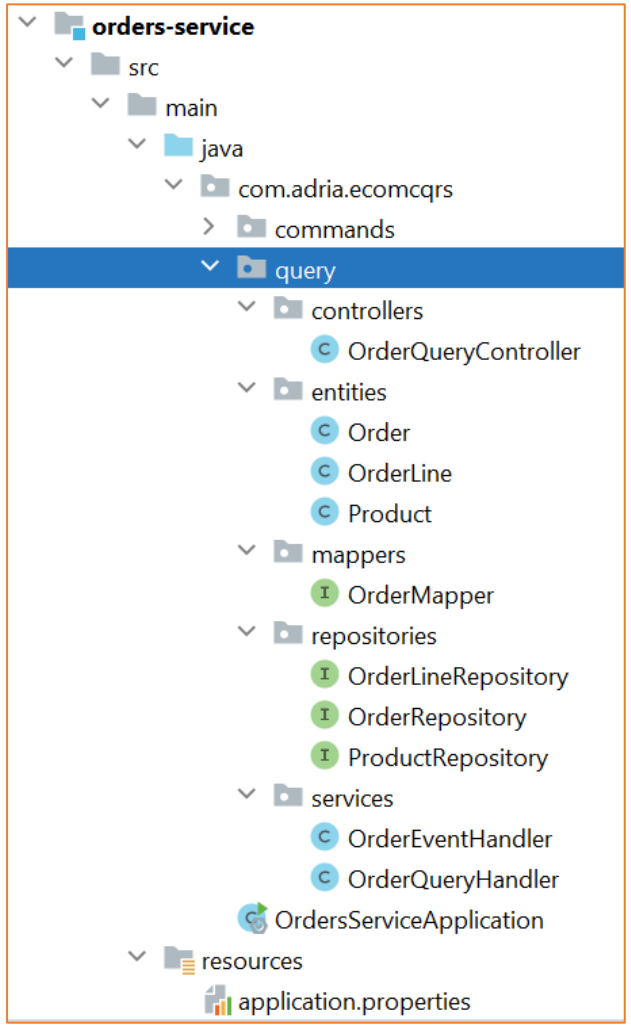
```
@Mapper(componentModel = "spring")
public interface OrderMapper {
   OrderDTO fromOrder(Order order);
   OrderLineDTO fromOrderLine(OrderLine orderLine);
}
```

Order-service Query Side / Event Handlers



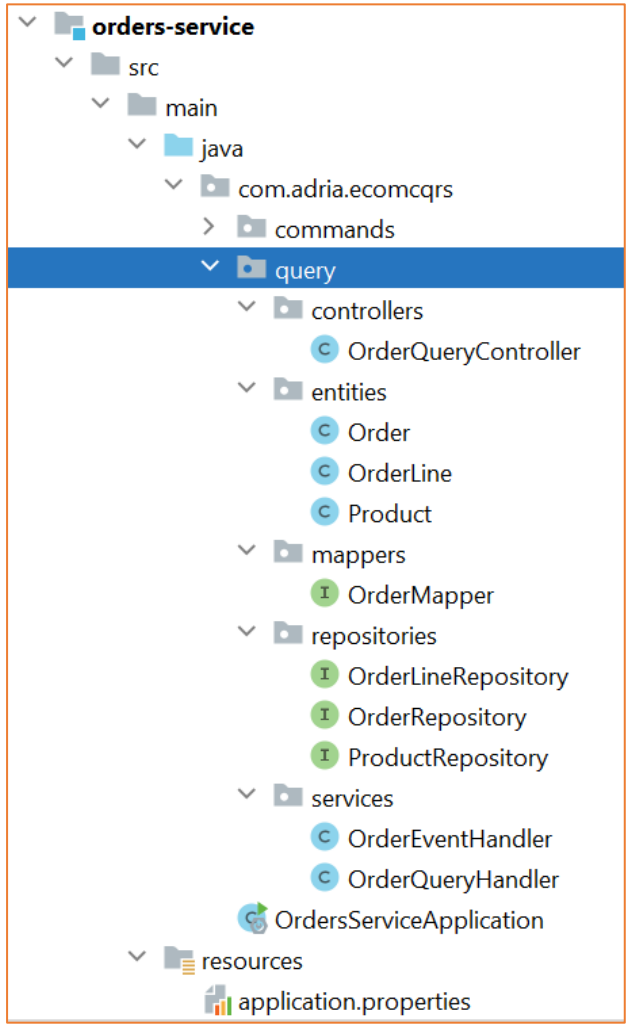

```
@Component
@Slf4j
@AllArgsConstructor
public class OrderEventHandler {
   private ProductRepository productRepository;
   private OrderLineRepository orderLineRepository;
   private OrderRepository orderRepository;


   @EventHandler
   public void on(OrderCreatedEvent event){
      log.info("**** Order Query Side *****");
      log.info("OrderCreatedEvent Received");
      orderRepository.save(new Order(
            event.getOrderId(),
            event.getDate(),
            event.getStatus(),
            new ArrayList<OrderLine>()
      ));
   }
```

Order-service Query Side / Event Handlers



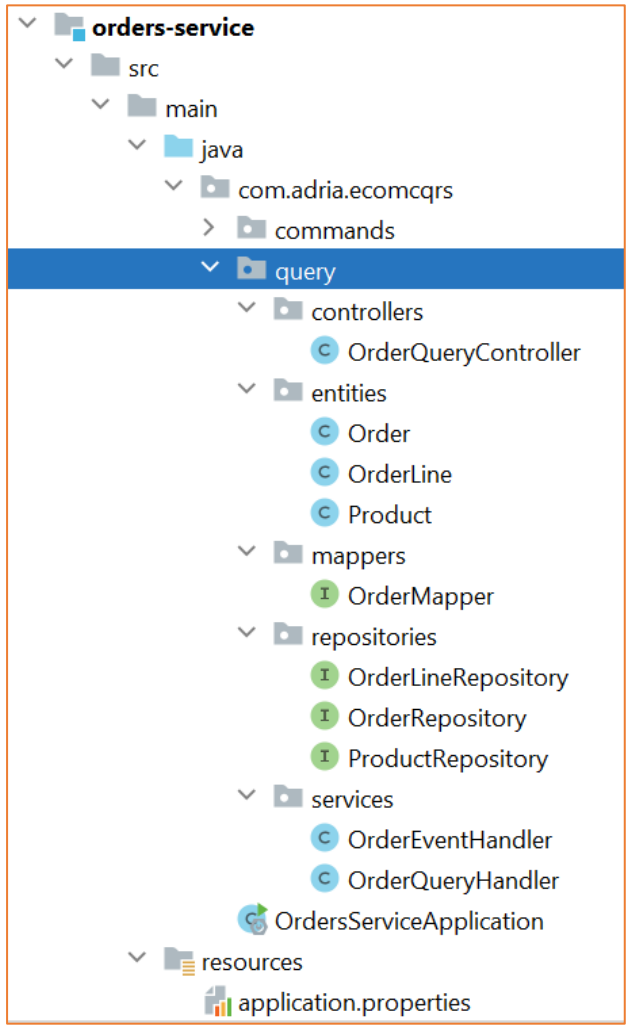

```
@EventHandler
  public void on(ProductCreatedEvent event){
    log.info("**** Order Query Side *****");  log.info("ProductCreatedEvent Received");
    productRepository.save(new Product(
      event.getProductId(),   event.getName()
    ));
  }
  @EventHandler
  public void on(OrderLineAddedEvent event){
    log.info("**** Order Query Side *****");
    log.info("OrderLineAddedEvent Received");
    Order order=orderRepository.findById(event.getOrderId()).get();
    Product product=productRepository.findById(event.getProductId()).get();
    OrderLine orderLine=new OrderLine();
    orderLine.setPrice(event.getPrice());
    orderLine.setQuantity(event.getQuantity());
    orderLine.setProductId(product.getProductId());
    orderLine.setProductName(product.getName());
    order.getOrderLines().add(orderLine);
    orderRepository.save(order);
  }
```

Order-service Query Side / Event Handlers



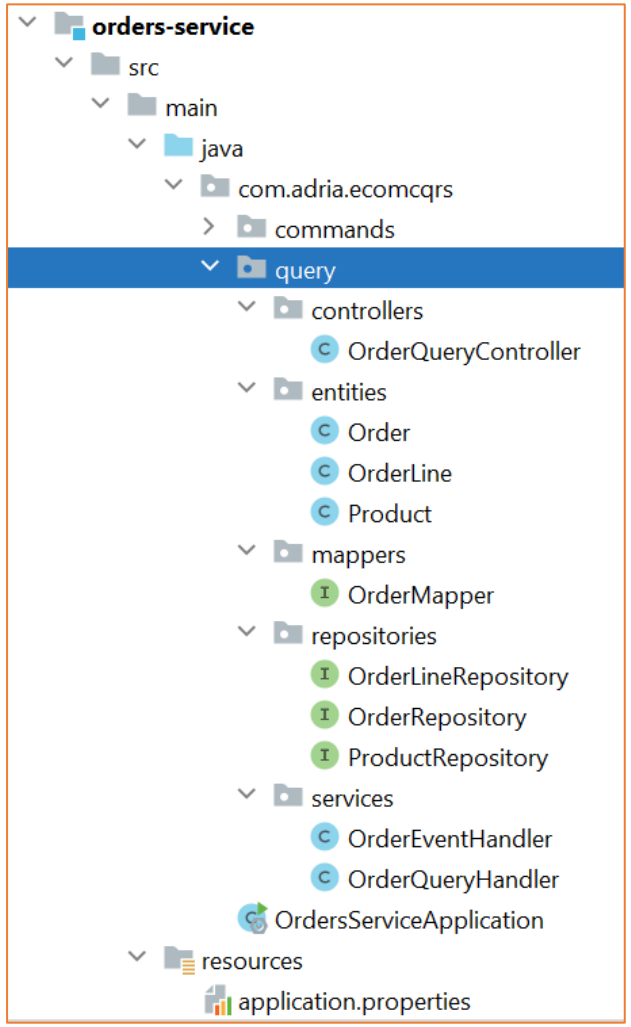

```
@EventHandler
  public void on (OrderLineRemovedEvent event) {
    log.info("**** Order Query Side *****");
    log.info("OrderLineAddedEvent Received");
    Order order=orderRepository.findById(event.getOrderId()).get();
    order.getOrderLines().removeIf(ol->ol.getProductId().equals(event.getProductId()));
    orderRepository.save(order);
  }
  @EventHandler
  public void on (OrderLineUpdatedEvent event) {
    log.info("**** Order Query Side *****");
    log.info("OrderLineUpdatedEvent Received");
    Order order=orderRepository.findById(event.getOrderId()).get();
    for(OrderLine ol : order.getOrderLines()){
      if(ol.getProductId().equals(event.getProductId())){
        ol.setQuantity(event.getQuantity());
        ol.setPrice(event.getPrice());
      }
    }
    orderRepository.save(order);
  }
```

Order-service Query Side / Event Handlers



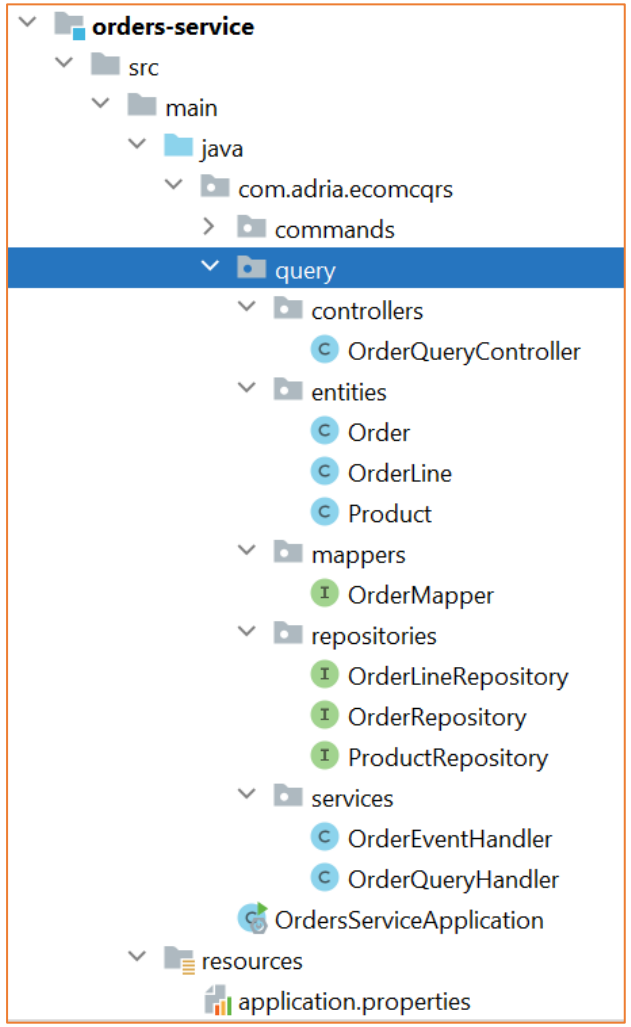

```
@EventHandler
   public void on (OrderConfirmedEvent event) {
      log.info("**** Order Query Side *****");
      log.info("OrderCreatedEvent Received");
      Order order=orderRepository.findById(event.getOrderId()).get();
      order.setStatus(event.getStatus());
      orderRepository.save(order);
   }
}
```

Order-service Query Side / Query Handlers



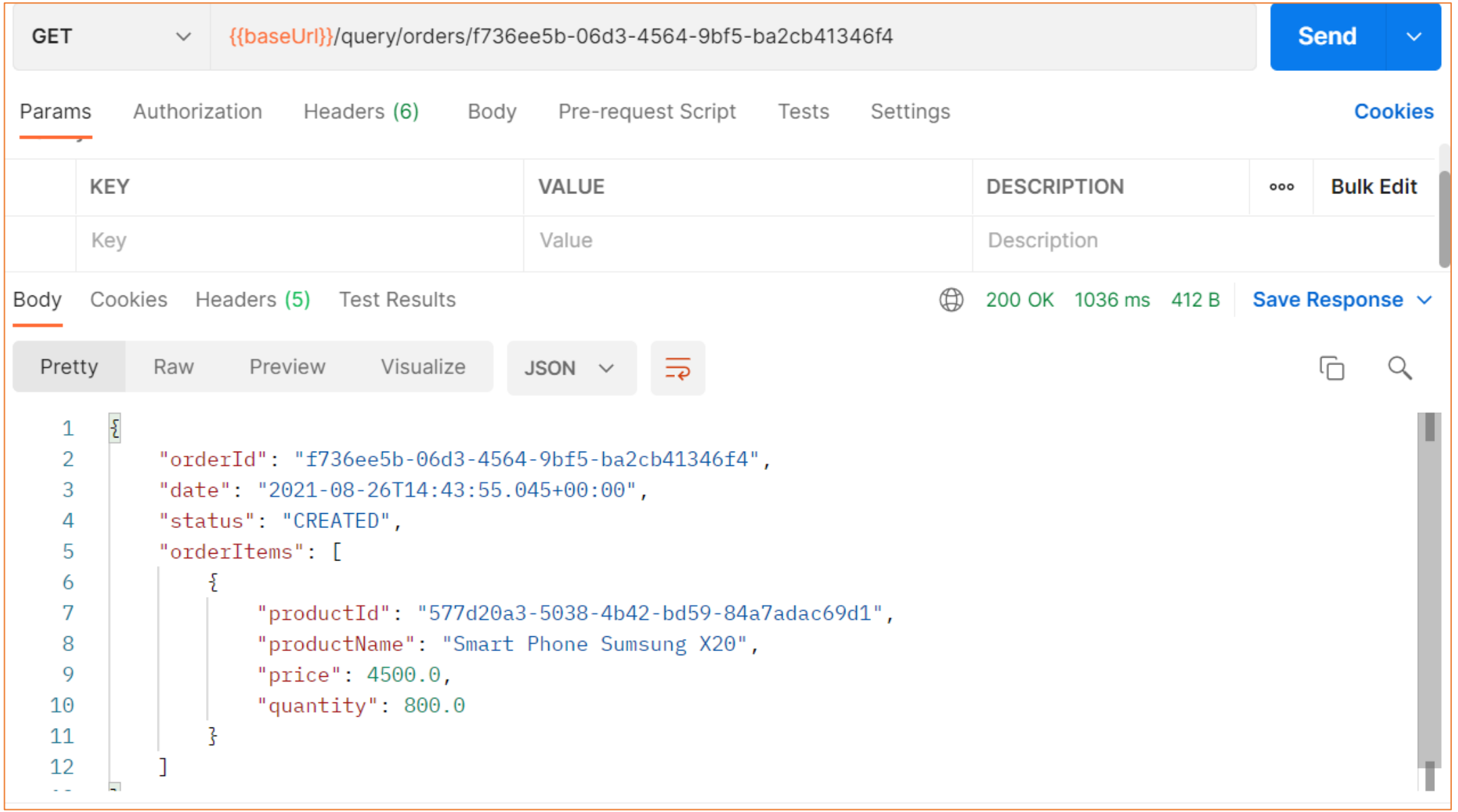

```
@Component
@AllArgsConstructor
@Slf4j
public class OrderQueryHandler {
   private OrderRepository orderRepository;
   private OrderMapper orderMapper;
   @QueryHandler
   public OrderDTO handle(GetOrderById query){
      log.info("Query Handler Get Order By Id =>"+query.getOrderId());
      Order order=orderRepository.findById(query.getOrderId()).get();
      log.info(order.toString());
      OrderDTO orderDTO=orderMapper.fromOrder(order);
      orderDTO.setOrderItems(order.getOrderLines().stream().map(ol-
>orderMapper.fromOrderLine(ol)).collect(Collectors.toList()));
      return orderDTO;
   }
}
```

Order-service Query Side / Query Handlers

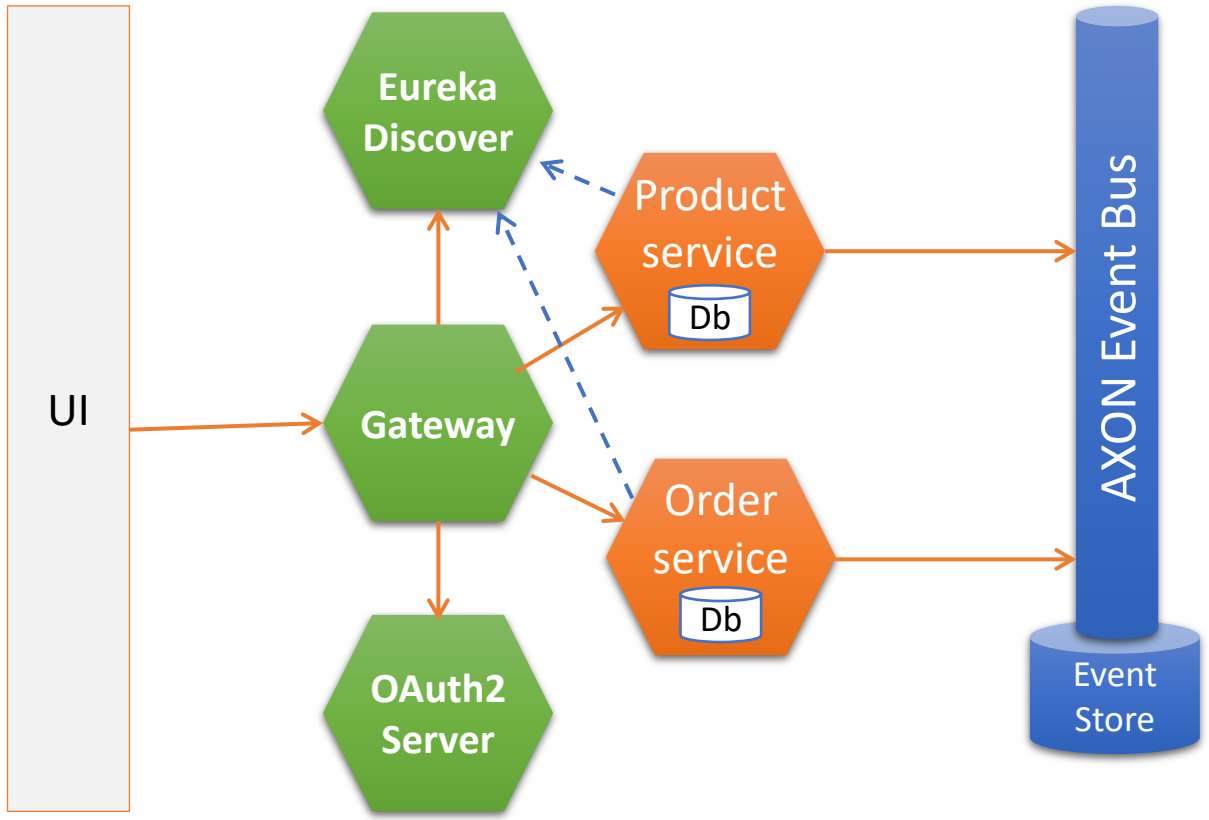



```
//@CrossOrigin("*")
@RestController
@AllArgsConstructor
@Slf4j
public class OrderQueryController {
   private QueryGateway queryGateway;
   @GetMapping("/query/orders/{orderId}")
   public CompletableFuture<OrderDTO> order(@PathVariable String orderId){
      log.info("Get Order Http Request orderId="+orderId);
      return queryGateway.query(new GetOrderById(orderId),
ResponseTypes.instanceOf(OrderDTO.class));
   }
}
```

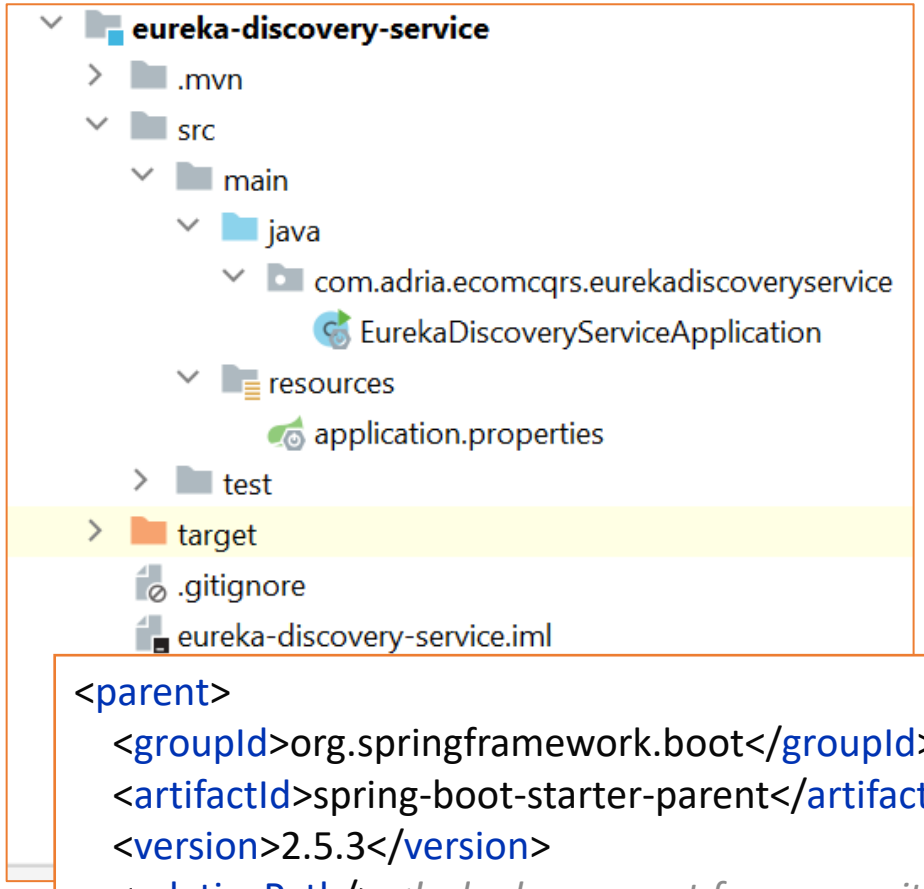
Order-service Query Side / Test



Eureka Discovery Service



Order-service Query Side / Query Model (Jpa Entities)




```
@SpringBootApplication
@EnableEurekaServer
public class EurekaDiscoveryServiceApplication {

   public static void main(String[] args) {
      SpringApplication.run(EurekaDiscoveryServiceApplication.class, args);
   }

}
```

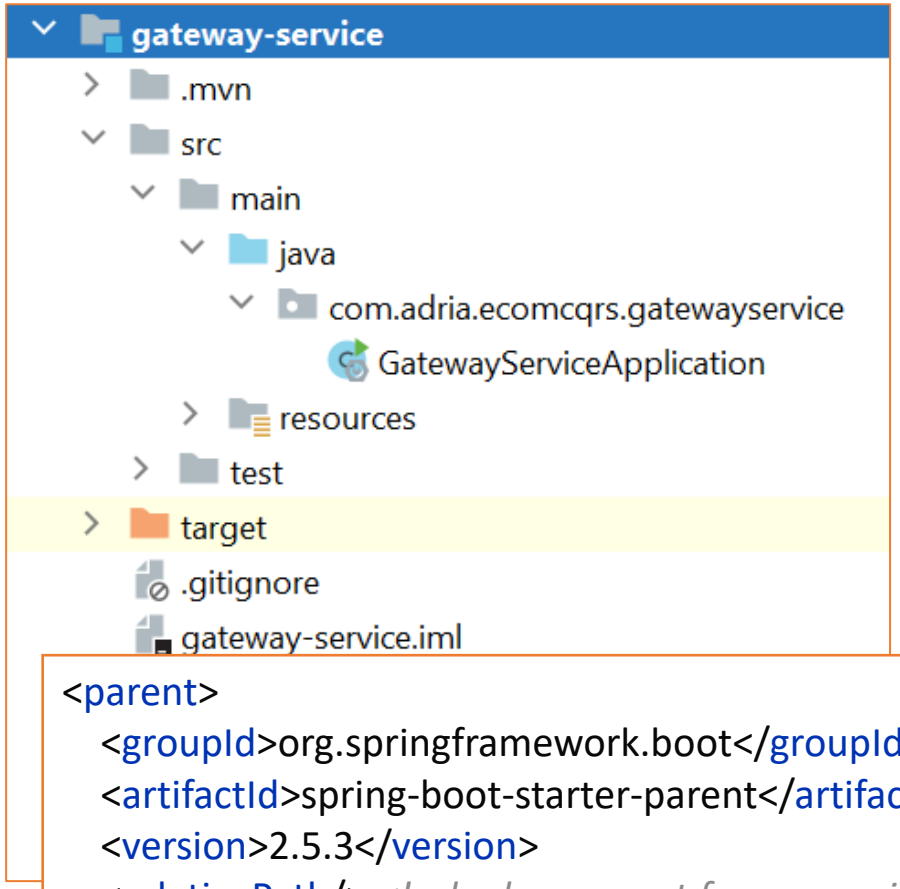
```
server.port=8761
# dont register server itself as a client
eureka.client.fetch-registry=false
# Does not register itself in the service registry
eureka.client.register-with-eureka=false
```

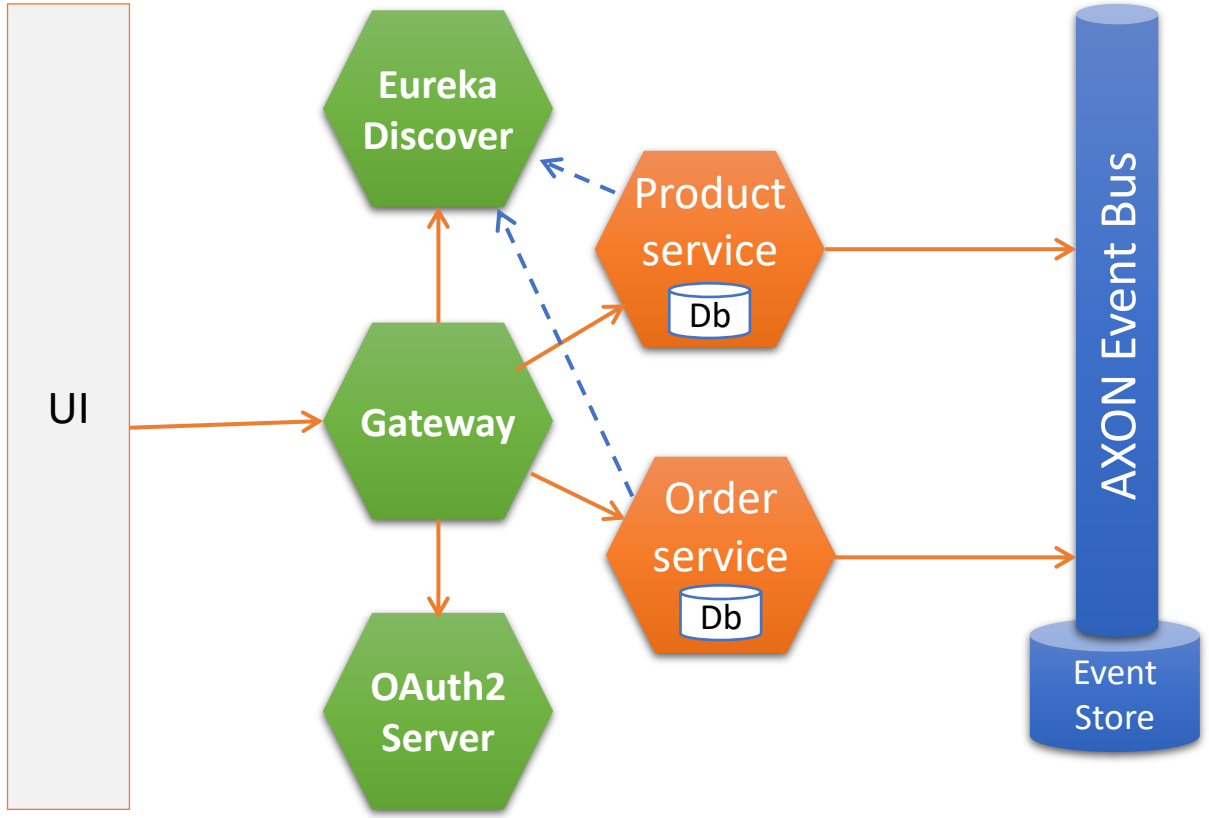
```
<parent>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-parent</artifactId>
   <version>2.5.3</version>
   <relativePath/> <!-- lookup parent from repository -->
</parent>
```

```
<dependency>
   <groupId>org.springframework.cloud</groupId>
   <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>
</dependency>
```

```
<properties>
   <java.version>1.8</java.version>
   <spring-cloud.version>2020.0.3</spring-cloud.version>
</properties>
```

Spring Cloud Gateway






```
<parent>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-parent</artifactId>
   <version>2.5.3</version>
   <relativePath/> <!-- lookup parent from repository -->
</parent>
```

```
<properties>
   <java.version>1.8</java.version>
   <spring-cloud.version>2020.0.3</spring-cloud.version>
</properties>
```

```
<dependency>
   <groupId>org.springframework.cloud</groupId>
   <artifactId>spring-cloud-starter-gateway</artifactId>
</dependency>
```

Spring Cloud Gateway

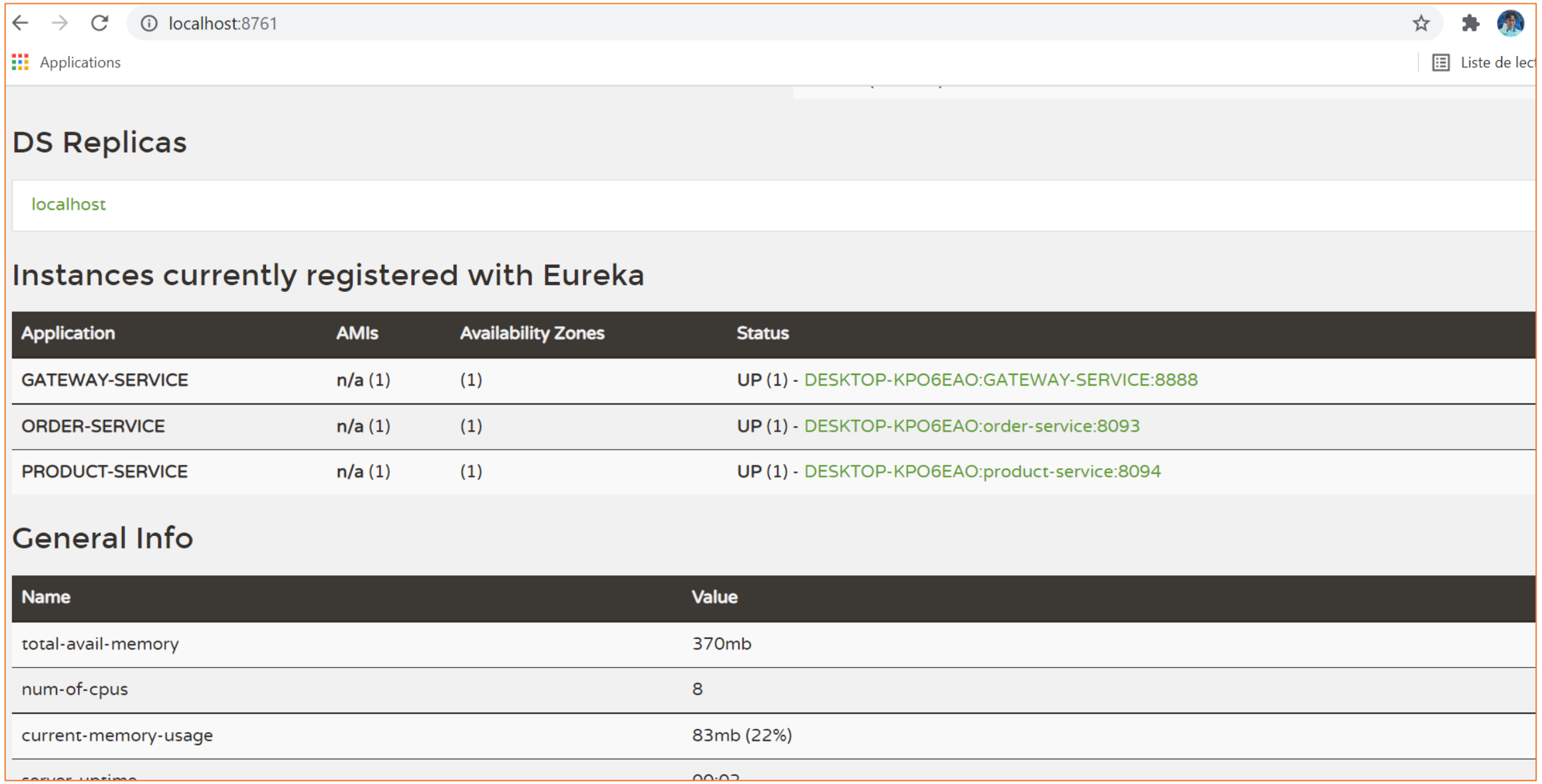
```
@SpringBootApplication
public class GatewayServiceApplication {
    public static void main(String[] args) {
        SpringApplication.run(GatewayServiceApplication.class, args);
    }

    @Bean
    DiscoveryClientRouteDefinitionLocator discoveryClientRouteDefinitionLocator(ReactiveDiscoveryClient rdc,
DiscoveryLocatorProperties dlp) {
        return new DiscoveryClientRouteDefinitionLocator(rdc, dlp);
    }
```

```
server.port=8888
spring.application.name=GATEWAY-SERVICE
spring.cloud.discovery.enabled=true
eureka.instance.prefer-ip-address=true
```

Spring Cloud Gateway

```
@Bean
   public WebFilter corsFilter() {
      return (ServerWebExchange ctx, WebFilterChain chain) -> {
         ServerHttpRequest request = ctx.getRequest();
         if (CorsUtils.isCorsRequest(request)) {
            ServerHttpResponse response = ctx.getResponse();
            HttpHeaders headers = response.getHeaders();
            headers.add("Access-Control-Allow-Origin", "*");
            headers.add("Access-Control-Allow-Methods", "GET, PUT, POST, DELETE, OPTIONS");
            headers.add("Access-Control-Max-Age", "3600");
            headers.add("Access-Control-Allow-Headers", "x-requested-with, authorization, Content-Type, Authorization,
credential, X-XSRF-TOKEN");
            if (request.getMethod() == HttpMethod.OPTIONS) {
               response.setStatusCode(HttpStatus.OK);
               return Mono.empty();
            }
         }
         return chain.filter(ctx);
      };
   }
}
```

Eureka Discovery Service

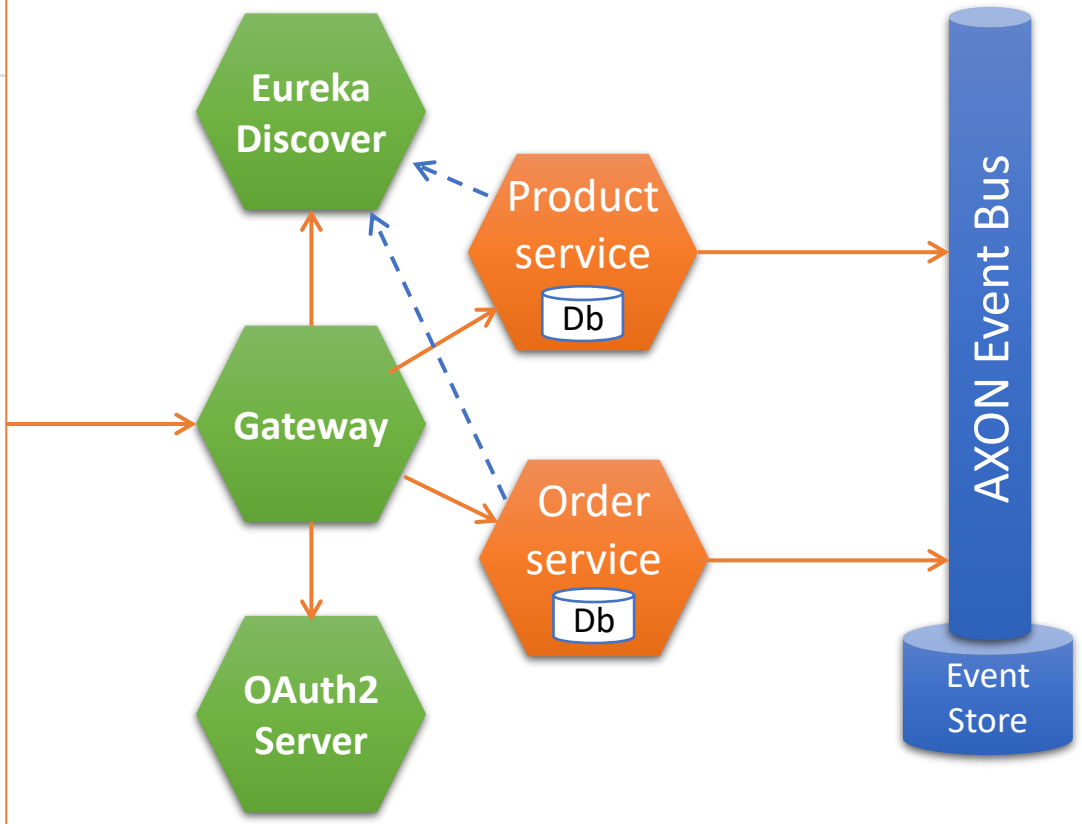


Eureka Discovery Service

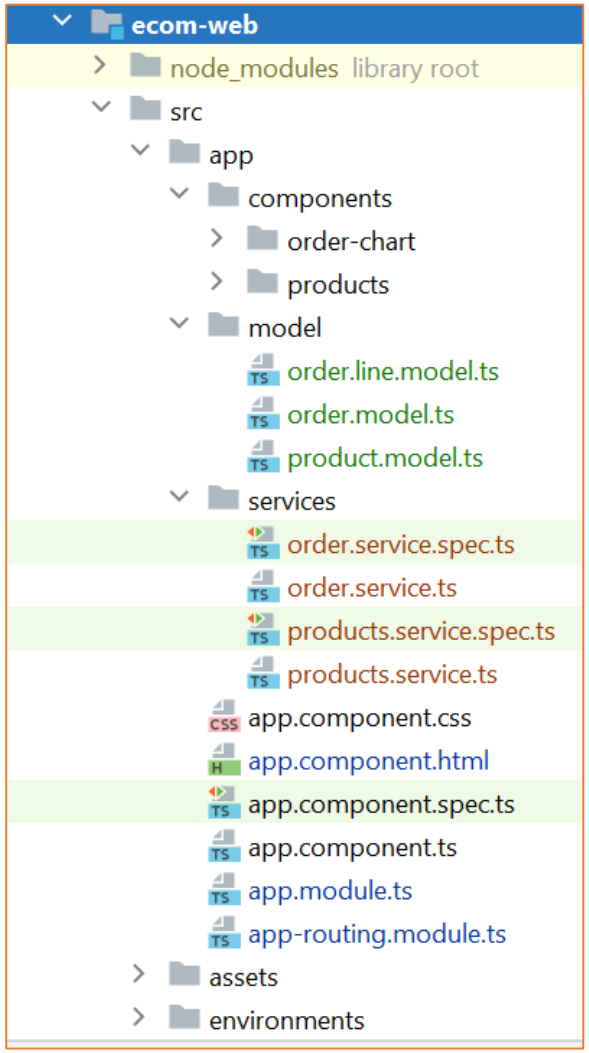
localhost:8888/PRODUCT-SERVICE/query/products/all

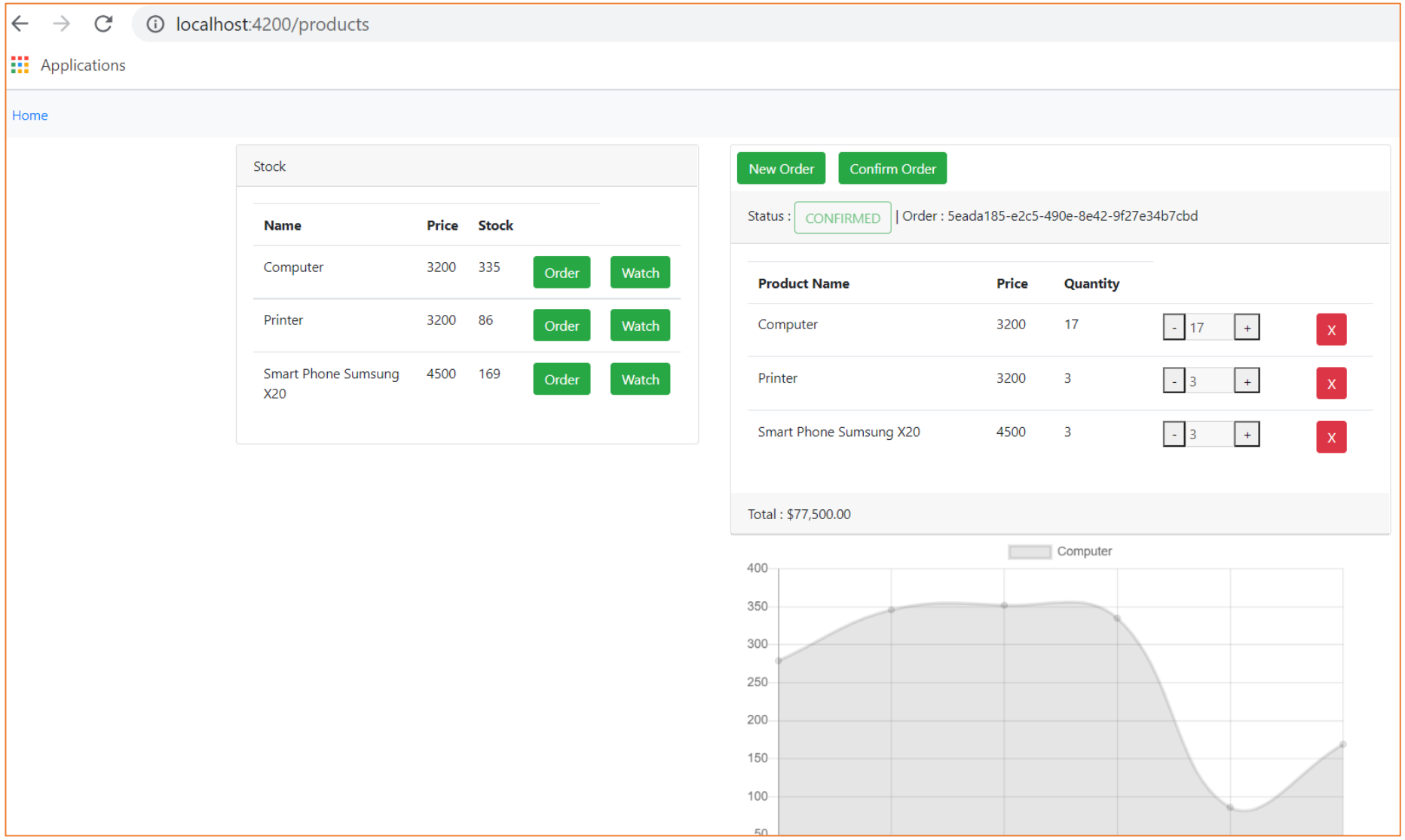
Applications


```
[
  {
    "productId": "a3f40583-0126-4d22-b73b-cce1ba5d571b",
    "name": "Computer",
    "price": 3200,
    "stock": 89,
    "eventType": null
  },
  {
    "productId": "0fba7fe6-30a5-4758-bd44-c83a6b8ae917",
    "name": "Printer",
    "price": 3200,
    "stock": 89,
    "eventType": null
  },
  {
    "productId": "577d20a3-5038-4b42-bd59-84a7adac69d1",
    "name": "Smart Phone Sumsung X20",
    "price": 4500,
    "stock": 172,
    "eventType": null
  }
]
```

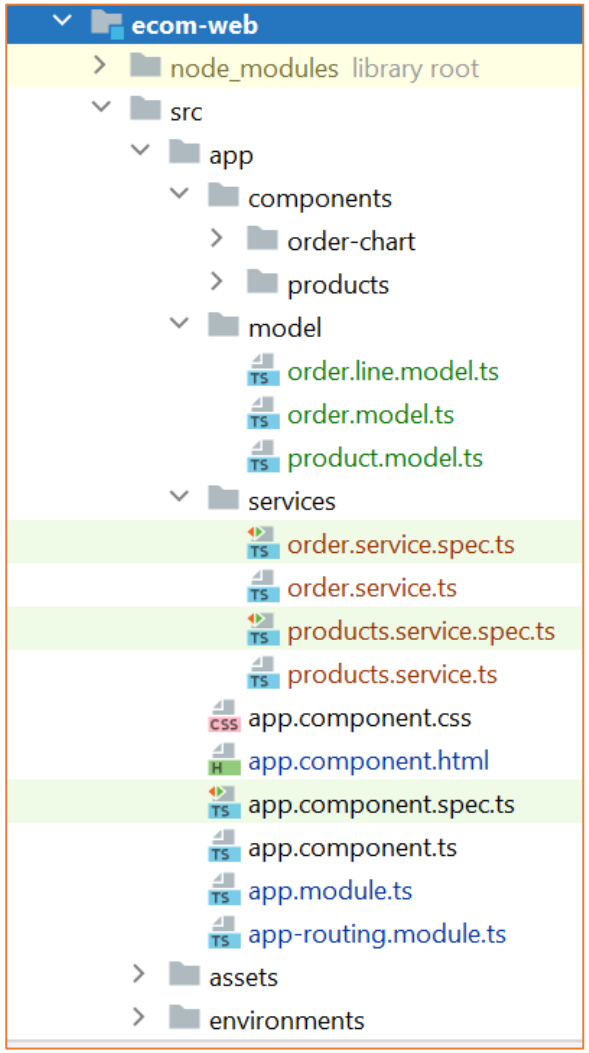



Angular UI Web Application





Angular UI Web Application



package.json

```
"dependencies": {
  "@angular/animations": "~12.0.0",
  "@angular/common": "~12.0.0",
  "@angular/compiler": "~12.0.0",
  "@angular/core": "~12.0.0",
  "@angular/forms": "~12.0.0",
  "@angular/platform-browser": "~12.0.0",
  "@angular/platform-browser-dynamic": "~12.0.0",
  "@angular/router": "~12.0.0",
  "bootstrap": "^4.6.0",
  "chart.js": "^2.9.3",
  "ng2-charts": "^2.2.3",
  "rxjs": "~6.6.0",
  "tslib": "^2.1.0",
  "zone.js": "~0.11.4"
},
```

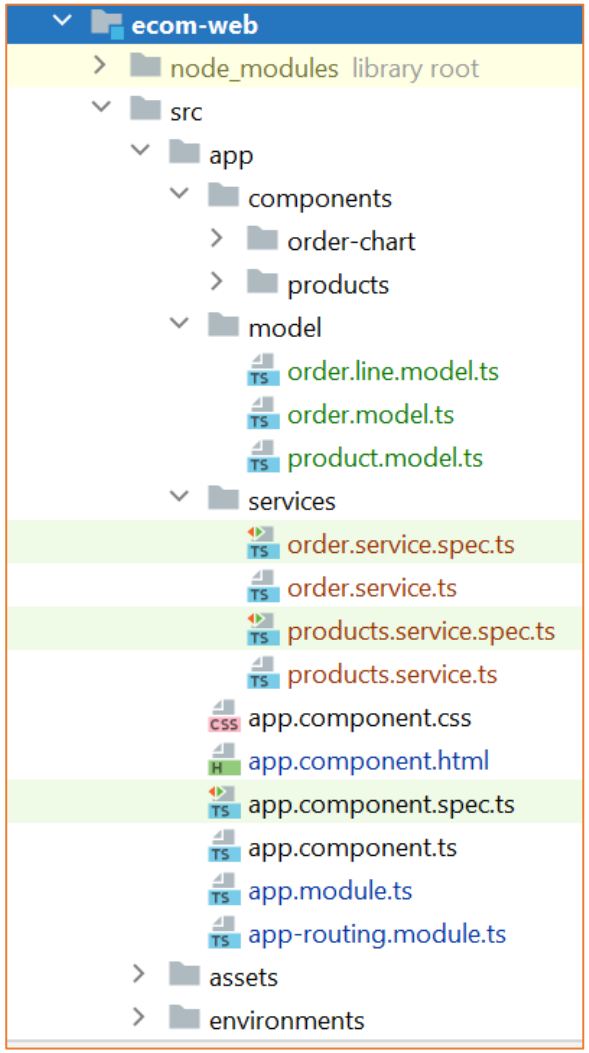
environnement.ts

```
export const environment = {
  production: false ,
  gatewayHost : "http://localhost:8888"
};
```

angular.json

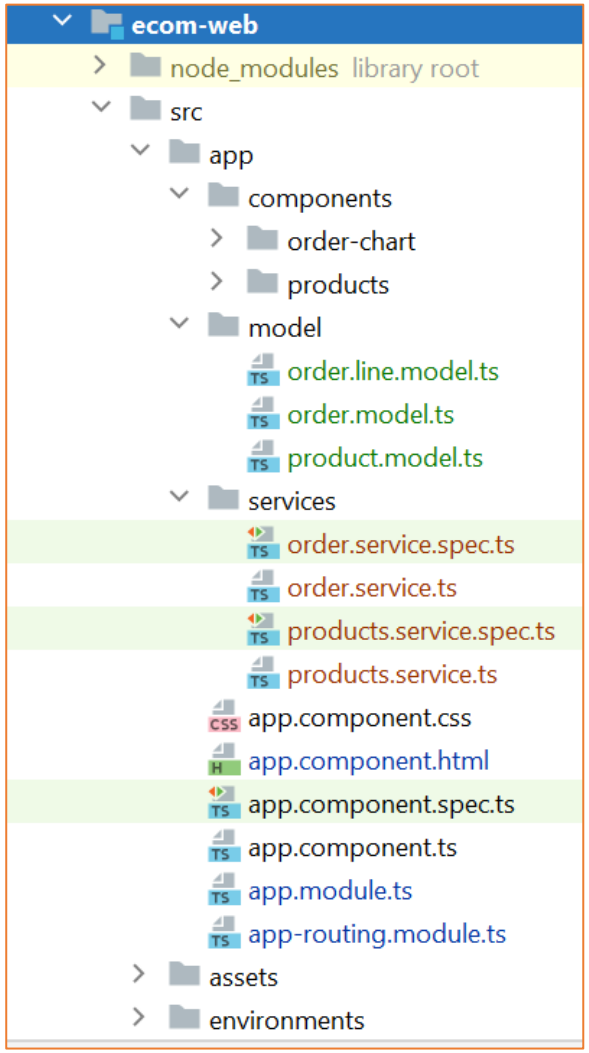
```
"styles": [
  "src/styles.css",
  "node_modules/bootstrap/dist/css/bootstrap.min.css"
],
```

Angular UI Web Application : app.module.ts



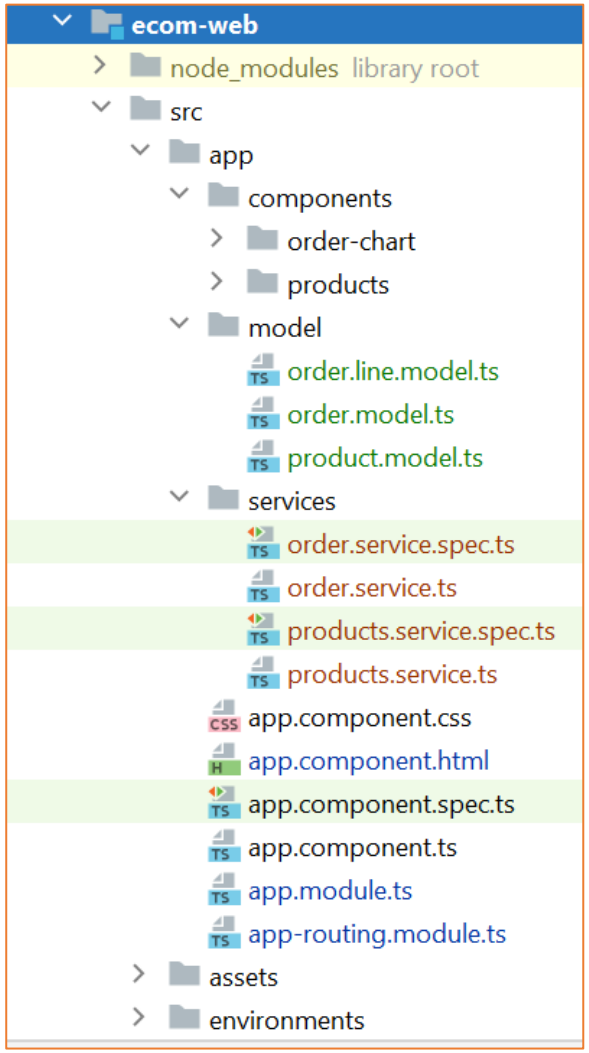

```
@NgModule({
  declarations: [
    AppComponent,
    ProductsComponent,
    OrderChartComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule, HttpClientModule, FormsModule,
    ChartsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Angular UI Web Application : app-routing.module.ts




```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import {ProductsComponent} from "./components/products/products.component";

const routes: Routes = [
 {
  path : "products", component : ProductsComponent
 } ,
 {
  path : "" , redirectTo : "products" , pathMatch : 'full'
 }
];

@NgModule({
 imports: [RouterModule.forRoot(routes)],
 exports: [RouterModule]
})
export class AppRoutingModule { }
```

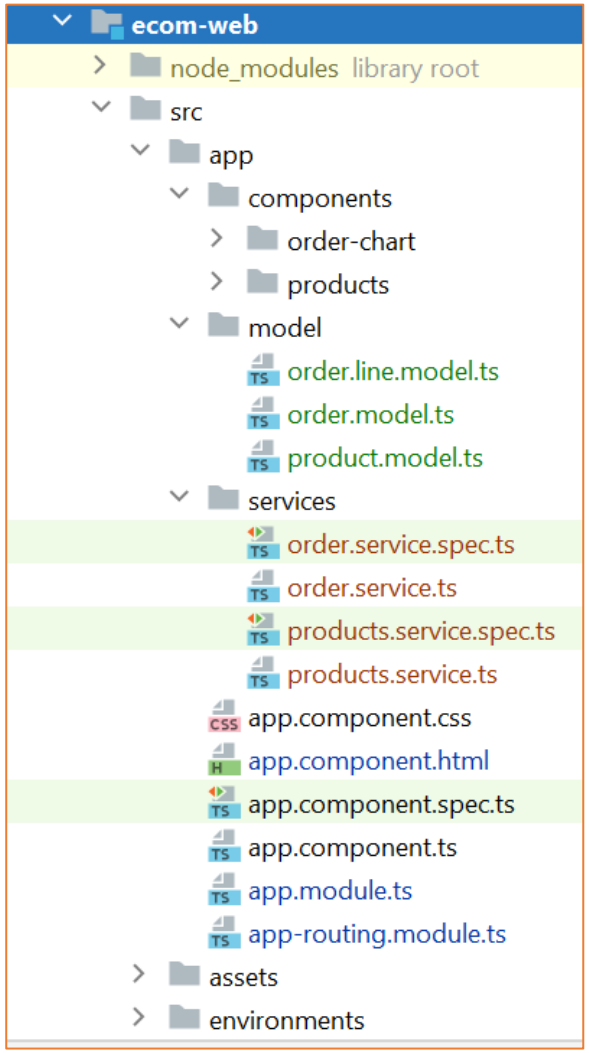
Angular UI Web Application : AppComponent




```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'ecom-web';
}
```

```
<nav class="navbar navbar-expand-sm bg-light">

  <!-- Links -->
  <ul class="navbar-nav">
    <li class="nav-item">
      <a class="nav-link" routerLink="/">Home</a>
    </li>
  </ul>
</nav>
<router-outlet>
</router-outlet>
```

Angular UI Web Application : Model

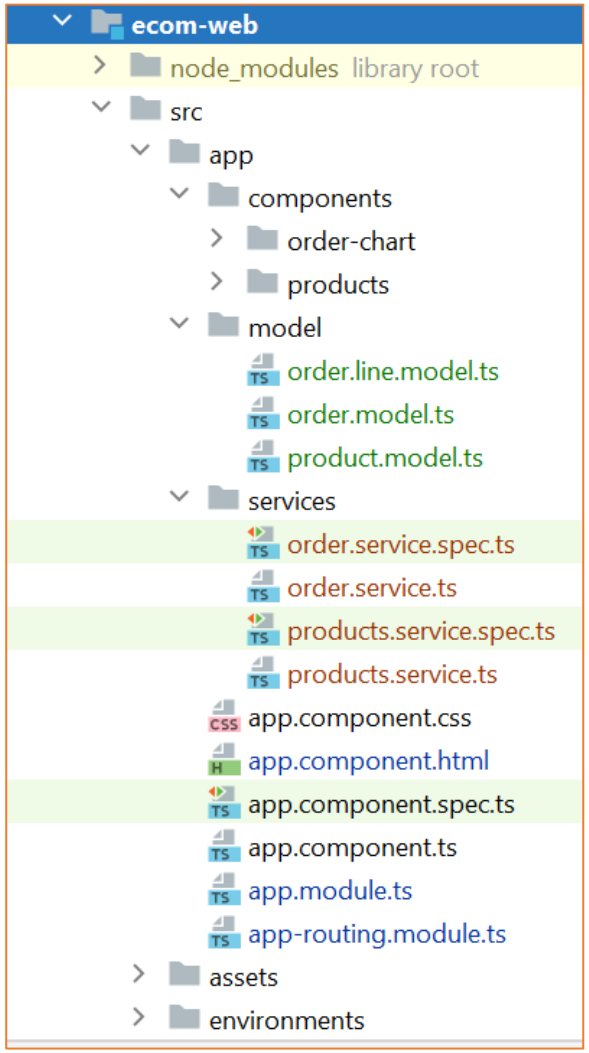



```
export interface Product {
  productId : string ,
  name : string ,
  price : number ,
  stock : number
}
```

```
export interface OrderLine {
  productId : string ,
  productName : string,
  price : number,
  quantity : number
}
```

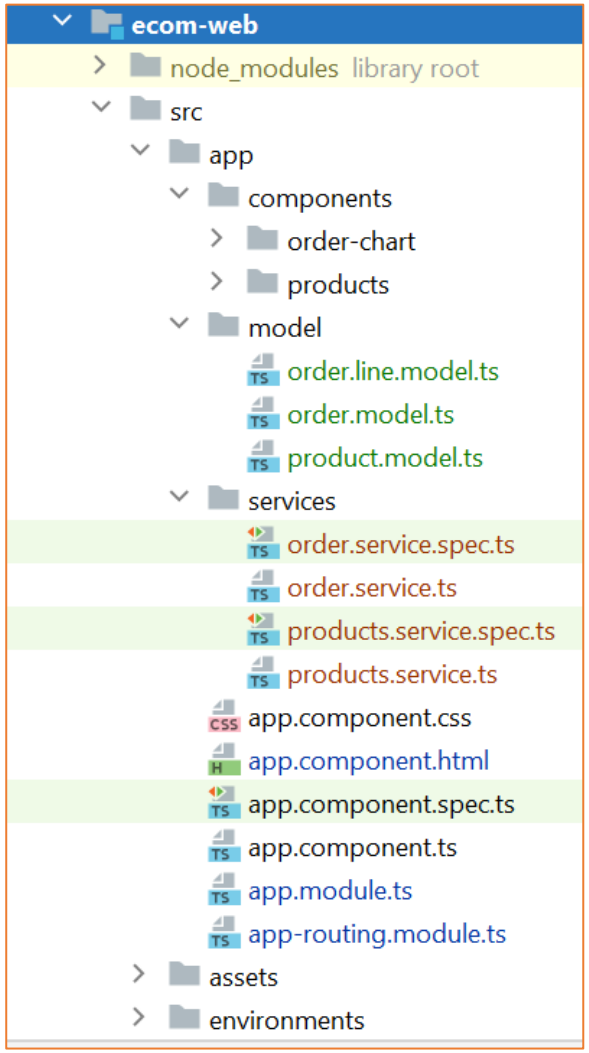
```
import {OrderLine} from "./order.line.model";

export interface Order {
  orderId : string | null,
  status : string,
  orderItems : OrderLine[]
}
```

Angular UI Web Application : order.service.ts



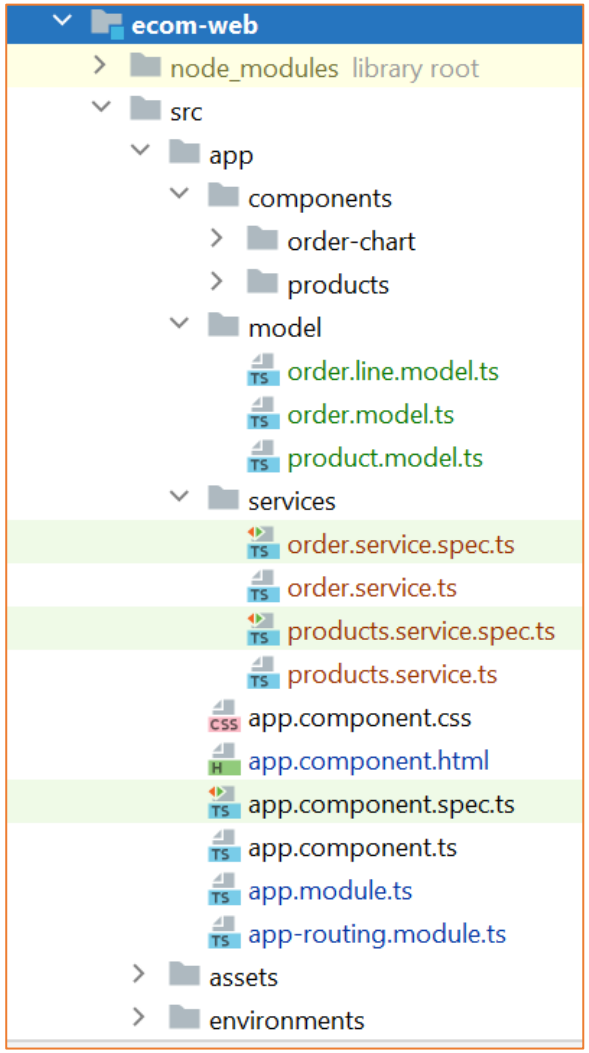

```
@Injectable({
  providedIn: 'root'
})
export class OrderService {

  constructor(private http : HttpClient) { }

  public addOrder()  {
    return this.http.post(environment.gatewayHost+"/ORDER-
SERVICE/commands/orders/create",{},{observe : 'response', responseType:'text'});
  }
```

Angular UI Web Application : order.service.ts



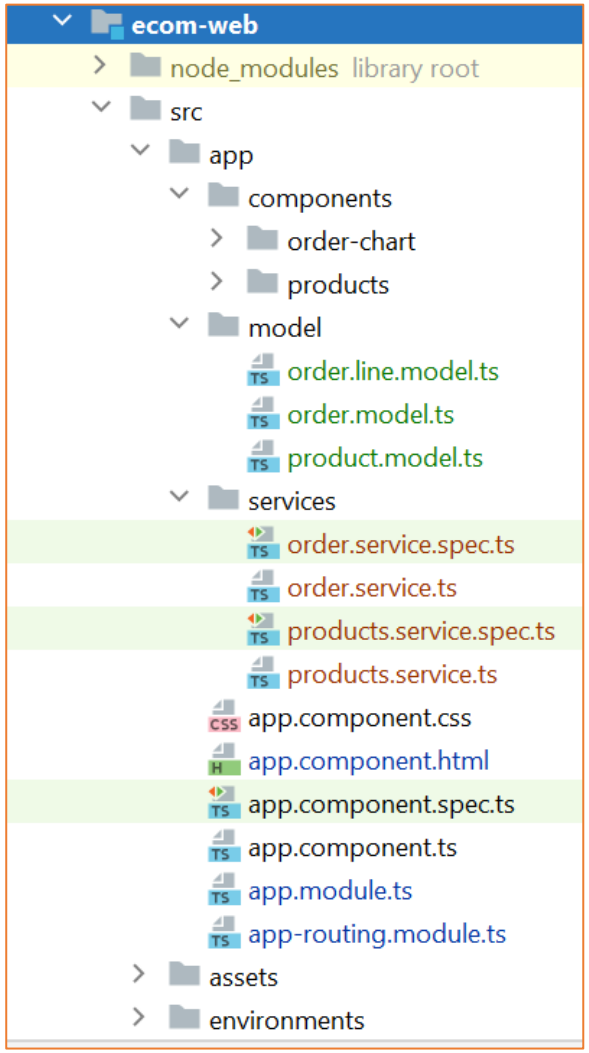

```
public addOrderLine(order:Order, orderLine:OrderLine, update :boolean)  {
   let url:string;
   if (!update){
     url=environment.gatewayHost+"/ORDER-SERVICE/commands/orders/addOrderLine";
   } else {
     url=environment.gatewayHost+"/ORDER-
SERVICE/commands/orders/updateOrderLine";
   }
   return this.http.put(url,{
     orderId:order.orderId,
     productId:orderLine.productId,
     price : orderLine.price,
     quantity:orderLine.quantity
   },{observe : 'response', responseType:'text'});
  }
```

Angular UI Web Application : order.service.ts



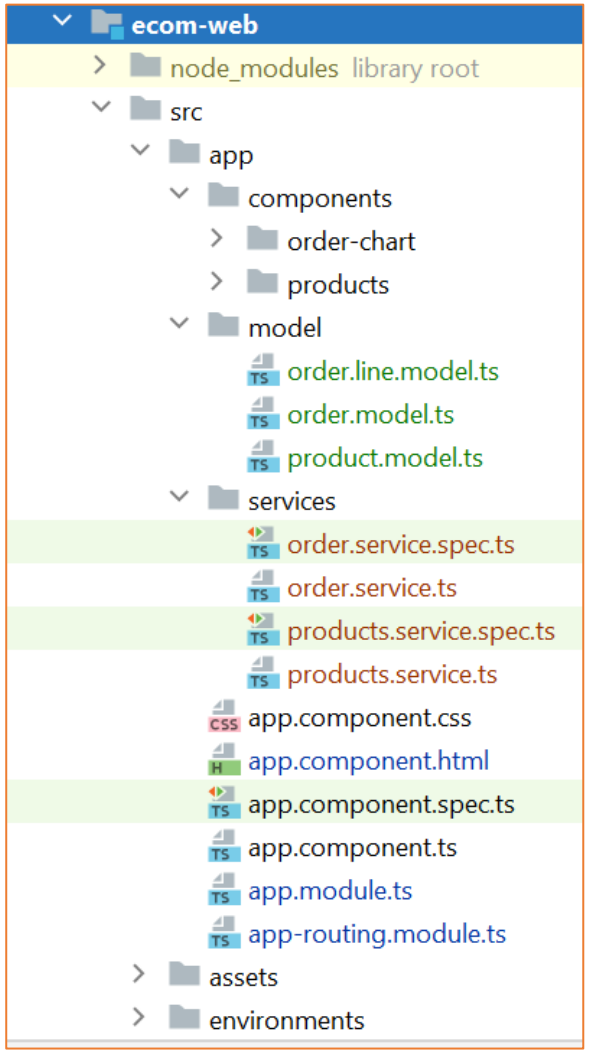

```
public saveCurrentOrder(orderId:string){
   window.localStorage.setItem("orderId",orderId);
  }
  public loadOrder(orderId:string):Observable<Order>{
   return this.http.get<Order>(environment.gatewayHost+"/ORDER-
SERVICE/query/orders/"+orderId);
  }
  public totalOrder(order:Order):number{
   let total=0;
   order.orderItems.forEach(ol=>{
    total+=ol.price*ol.quantity;
   })
   return  total;
  }
```

Angular UI Web Application : order.service.ts



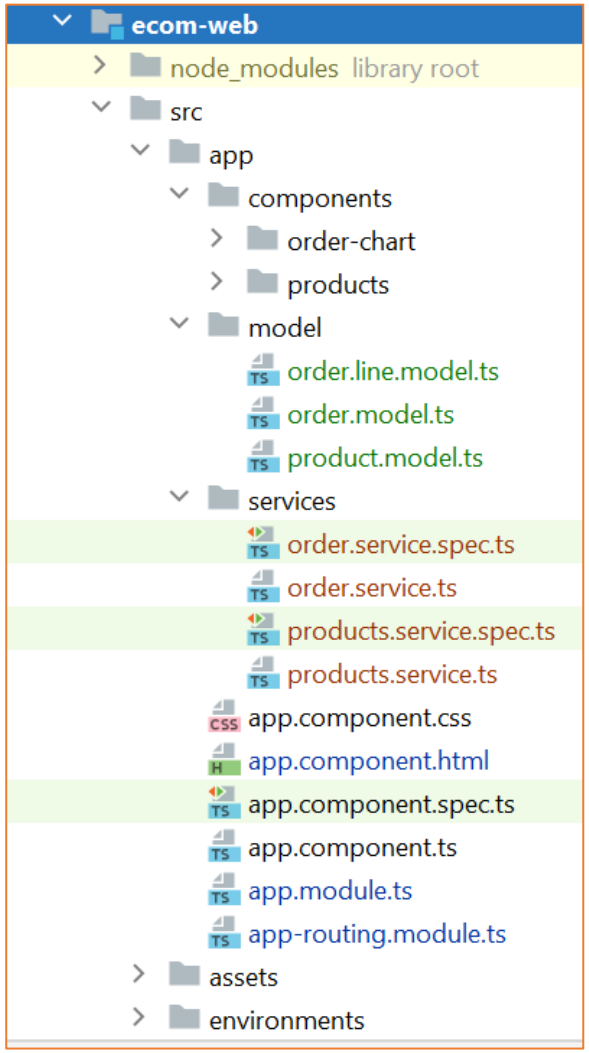

```
public confirmOrder(orderId:string){
   let url=environment.gatewayHost+"/ORDER-SERVICE/commands/orders/confirm";
   return this.http.put(url,{orderId:orderId,value:""},{observe : 'response',
responseType:'text'})
  }

  public removeOrderLine(orderId:string, productId:string){
   let url=environment.gatewayHost+"/ORDER-
SERVICE/commands/orders/removeOrderLine";
   return this.http.put(url,{orderId:orderId,productId:productId},{observe : 'response',
responseType:'text'})
  }
}
```

Angular UI Web Application : products.service.ts



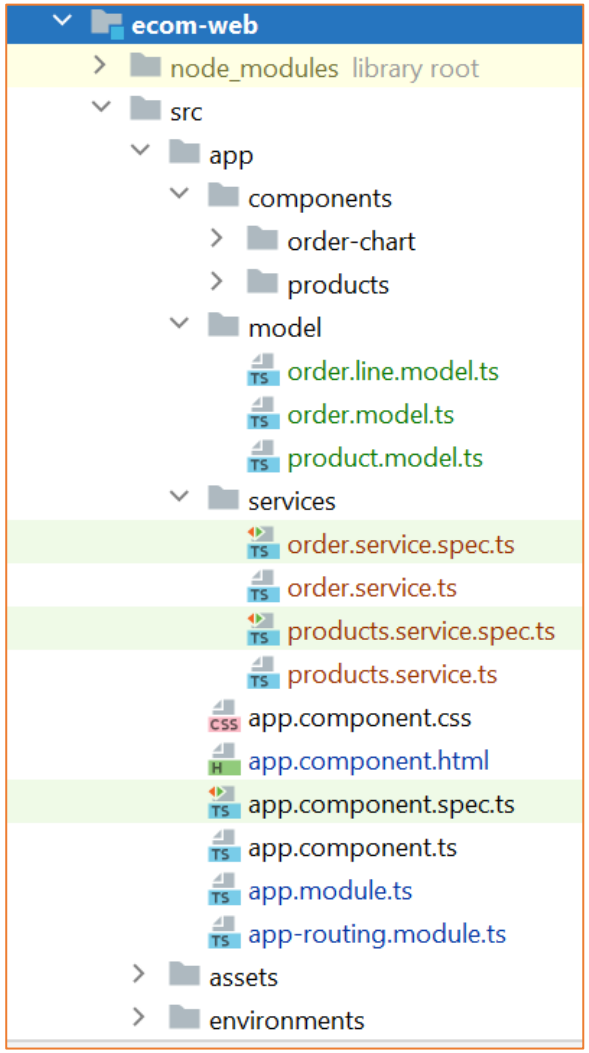

```
@Injectable({
  providedIn: 'root'
})
export class ProductsService {

  constructor(private http : HttpClient) { }


  public getProducts():Observable<Product[]> {
    return this.http.get<Product[]>(environment.gatewayHost+"/PRODUCT-
SERVICE/query/products/all");
  }
}
```

Angular UI Web Application : products.service.ts



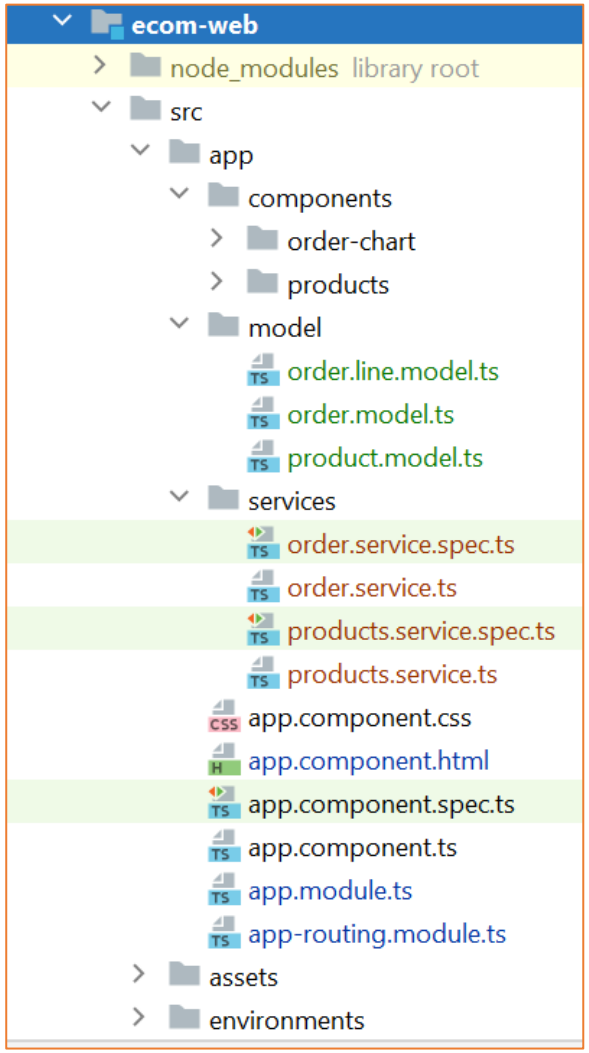

```
@Injectable({
  providedIn: 'root'
})
export class ProductsService {

  constructor(private http : HttpClient) { }


  public getProducts():Observable<Product[]> {
    return this.http.get<Product[]>(environment.gatewayHost+"/PRODUCT-
SERVICE/query/products/all");
  }
}
```

Angular UI Web Application : products.component.ts



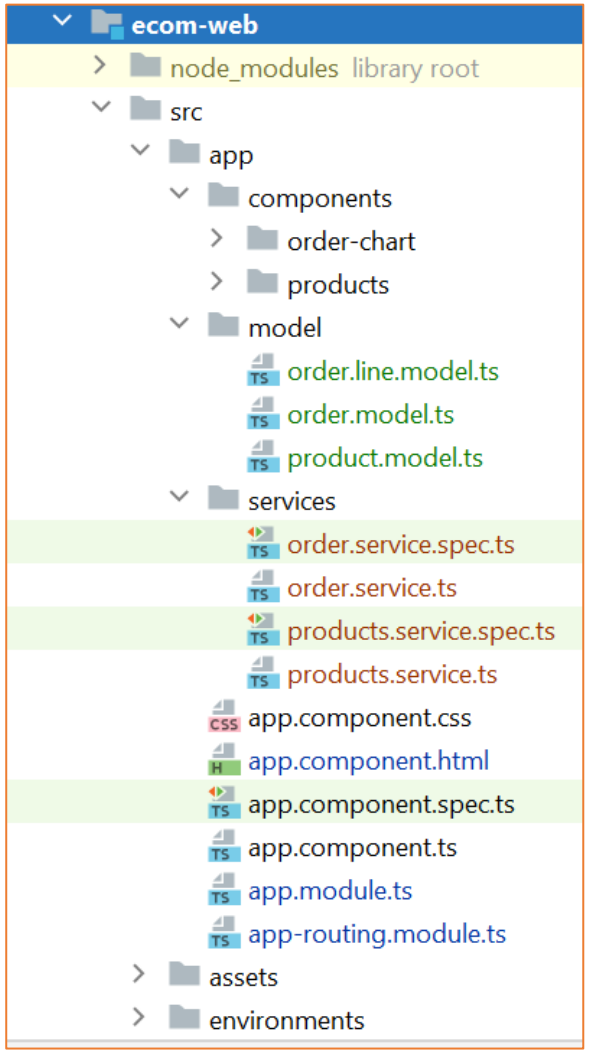

```
@Component({
  selector: 'app-products',
  templateUrl: './products.component.html',
  styleUrls: ['./products.component.css']
})
export class ProductsComponent implements OnInit {

  public products : Product[] | undefined;
  public order : Order | undefined;
  public errorMessage:string | undefined;
  public currentProduct:Product | undefined;

  @ViewChild(OrderChartComponent) orderChart:OrderChartComponent | undefined;

  constructor(private productsService : ProductsService, public orderService : OrderService) { }
```

Angular UI Web Application : products.component.ts



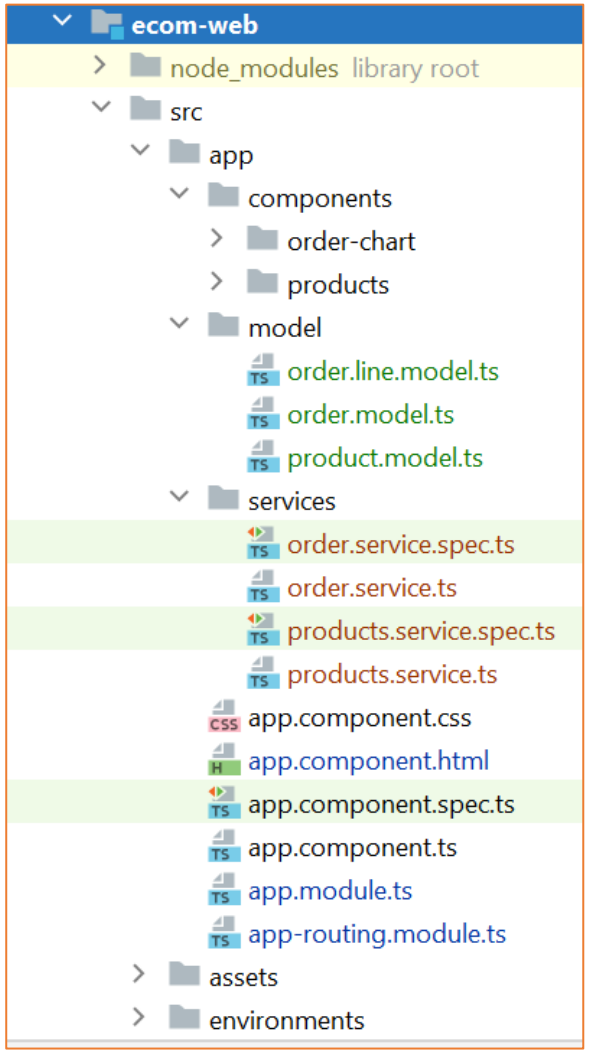

```
ngOnInit(): void {
   this.handleLoadProducts();
   let orderId =localStorage.getItem("orderId");
   if(orderId!=null)
    this.handleLoadOrder(orderId);
   this.connect();
 }

 connect():void{
   let source=new EventSource(environment.gatewayHost+"/PRODUCT-
SERVICE/query/products/watch");
   source.addEventListener("message",message=>{
     let product=JSON.parse(message.data);
     if(product.eventType==="ProductCreatedEvent"){
       this.products?.push(product);
     }else{
       this.products=this.products?.map((p)=>p.productId==product.productId?product:p);
     }
   });
 }
```

Angular UI Web Application : products.component.ts



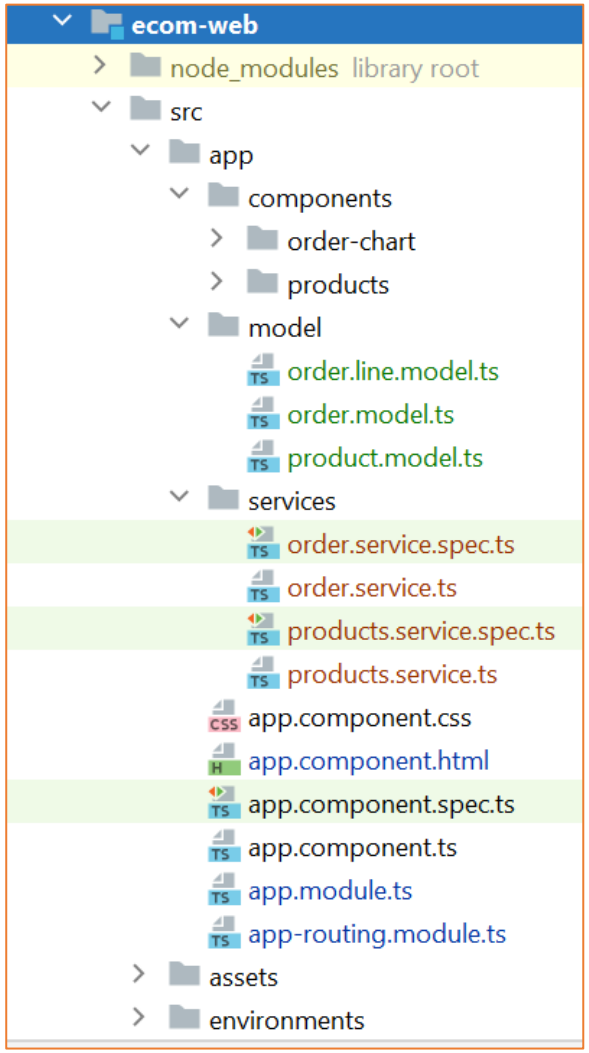

```
handleLoadProducts(){
  this.productsService.getProducts().subscribe(data=>{
    this.products=data;
  });
 }
 handleLoadOrder(orderId:string){
  this.orderService.loadOrder(orderId).subscribe(data=>{
    this.order=data;
    this.errorMessage=undefined;
  },err=>{
    this.errorMessage=JSON.parse(err.error).message;
  })
 }

 handleNewOrder(product : Product) {
  if(this.order==undefined){
    this.addNewOrder();
  }
  //if(this.order?.status==="CONFIRMED") return;
  this.addOrderLine(product,1,false);
 }
```

Angular UI Web Application : products.component.ts



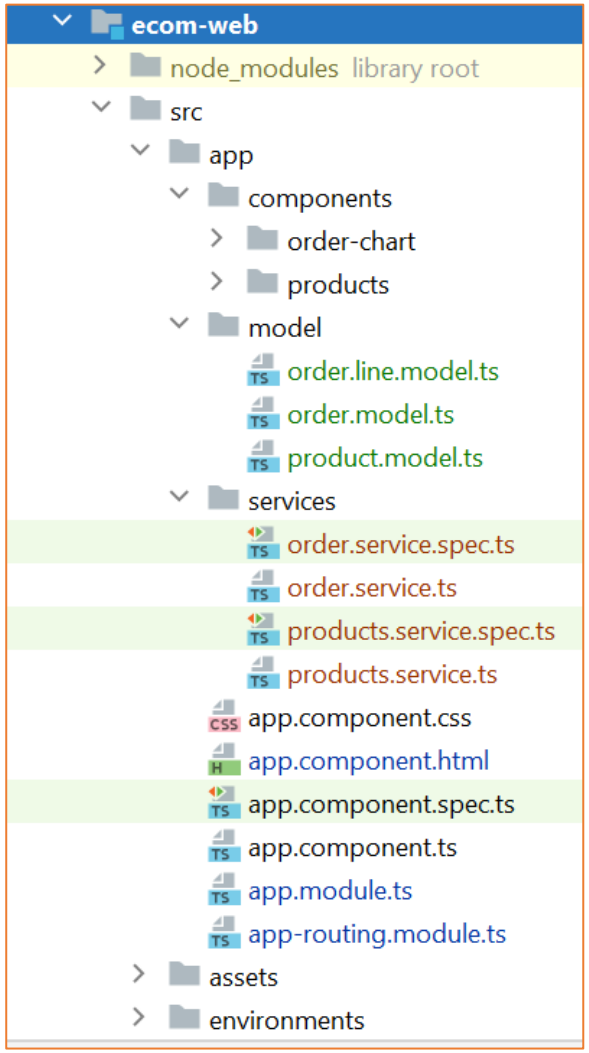

```
addNewOrder(){
  this.orderService.addOrder().subscribe(data=>{
     let orderId= data.body;
     if(orderId!=null){
      this.orderService.saveCurrentOrder(orderId);
      this.handleLoadOrder(orderId)
     }
     this.errorMessage=undefined;
   },err=>{
   this.errorMessage=JSON.parse(err.error).message;
  });
 }
```

Angular UI Web Application : products.component.ts



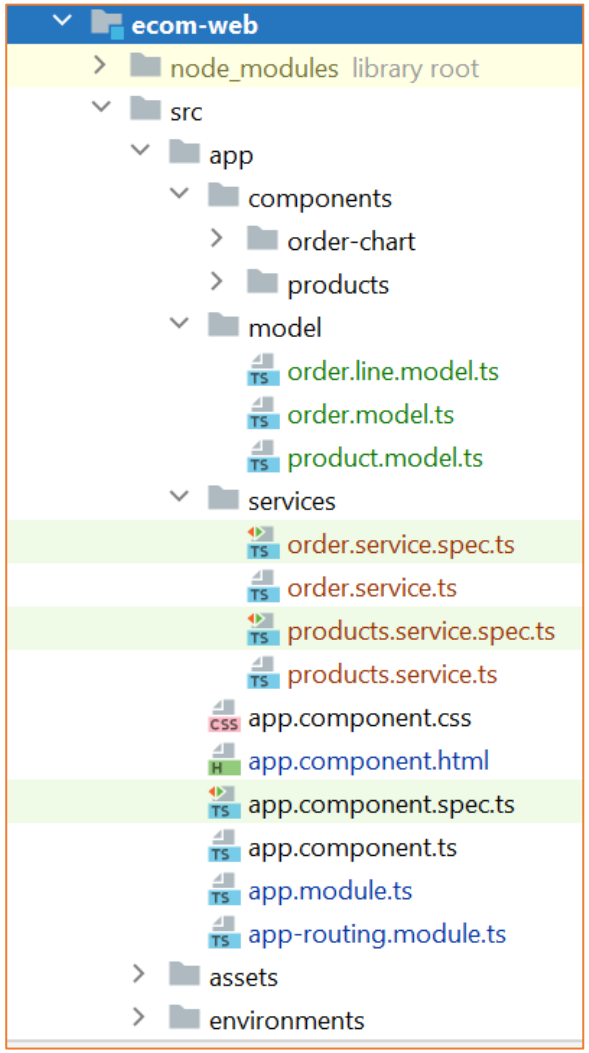

```
addOrderLine(product:Product, quantity : number, update :boolean){
  if(this.order){
   //if(this.order?.status==="CONFIRMED") return;
   let orderLine:OrderLine={
    productId:product.productId,
    productName : product.name,
    price : product.price,
    quantity : quantity
   }
   this.orderService.addOrderLine(this.order,orderLine,update).subscribe(
    data=>{
     if (!update) this.order?.orderItems.push(orderLine);
     this.errorMessage=undefined;
    }, err=>{
     this.errorMessage=JSON.parse(err.error).message;
    }
   )
  }
 }
```

Angular UI Web Application : products.component.ts



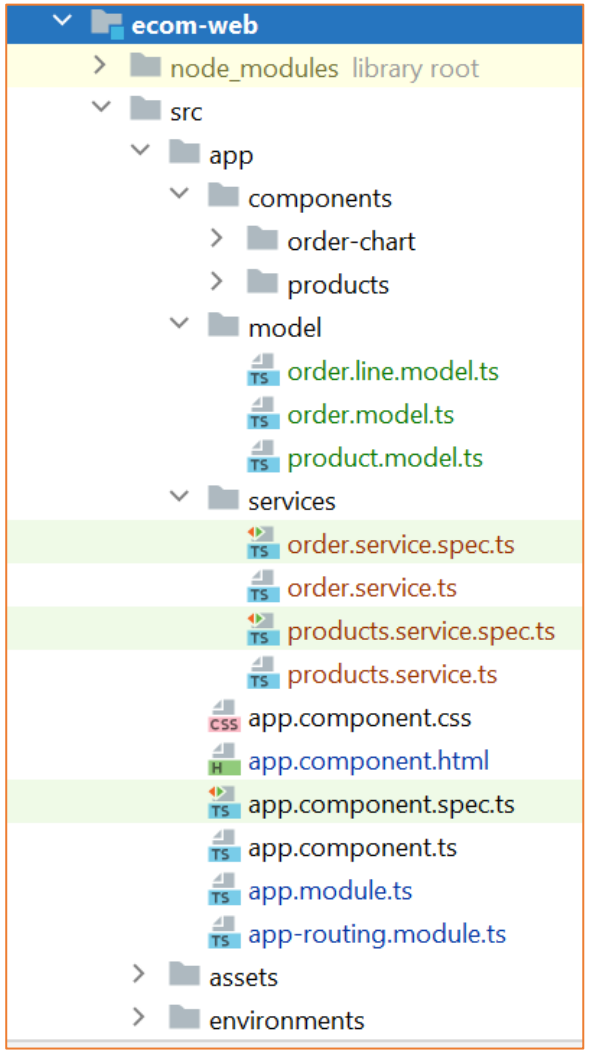

```
decQuantity(ol: OrderLine) {
  if(this.order?.status==="CONFIRMED") return;
  if(ol.quantity>0){
   --ol.quantity;
   let prod: Product={
    productId : ol.productId,
    name : ol.productName,
    price : 0,
    stock : 0
   }
   this.addOrderLine(prod,ol.quantity,true);
  }
 }
```

Angular UI Web Application : products.component.ts



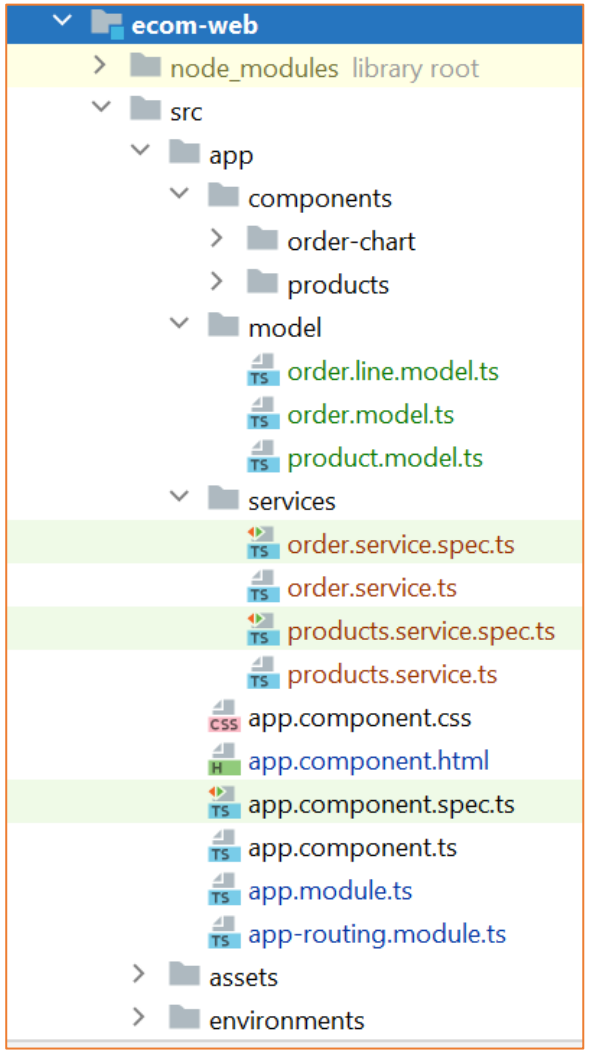

```
incQuantity(ol: OrderLine) {
  if(this.order?.status==="CONFIRMED") return;
  ++ol.quantity;
  let prod: Product={
    productId : ol.productId,
    name : ol.productName,
    price : ol.price,
    stock : 0
  }
  this.addOrderLine(prod,ol.quantity,true);
 }
```

Angular UI Web Application : products.component.ts



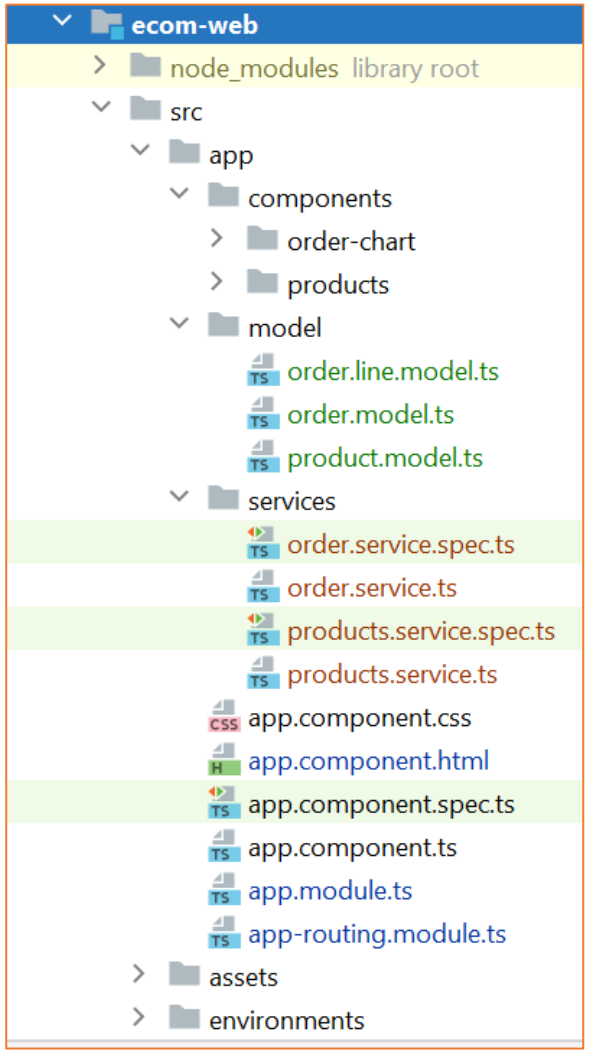

```
handleChangeQuantity(ol:OrderLine) {
  if(this.order?.status==="CONFIRMED") return;
  let prod: Product={
    productId : ol.productId,
    name : ol.productName,
    price : ol.price,
    stock : 0
  }
  this.addOrderLine(prod,ol.quantity,true);
 }
```

Angular UI Web Application : products.component.ts



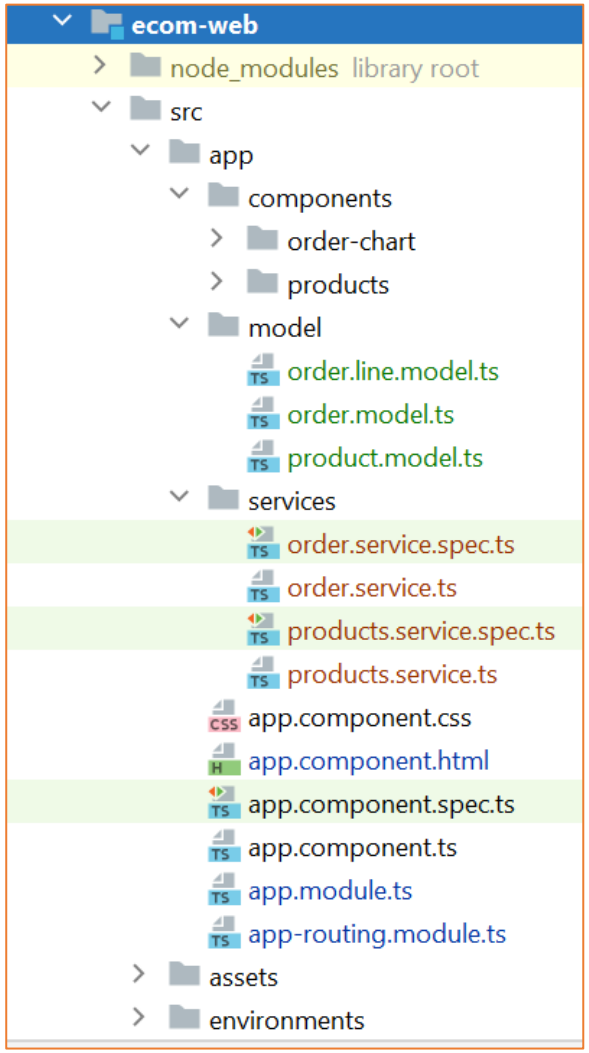

```
confirmOrder() {
  if(this.order?.status==="CONFIRMED") return;
  if(this.order?.orderId){
   let orderId=this.order.orderId;
   this.orderService.confirmOrder(this.order.orderId)
    .subscribe(data=>{
     this.handleLoadOrder(orderId);
     this.handleLoadProducts();
    },err=>{
     this.errorMessage=JSON.parse(err.error).message;
    });
  }
}
```

Angular UI Web Application : products.component.ts



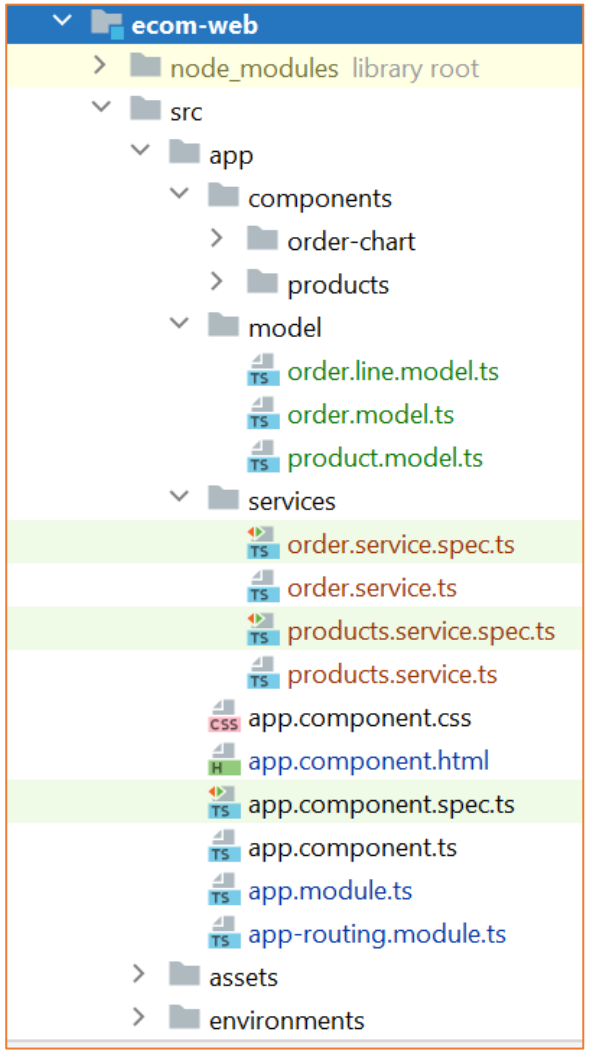

```
deleteItem(ol: OrderLine) {
   if(this.order?.orderId){
    let orderId=this.order.orderId;
    this.orderService.removeOrderLine(this.order.orderId,ol.productId)
      .subscribe(data=>{
       this.handleLoadOrder(orderId);
      },err=>{
       this.errorMessage=JSON.parse(err.error).message;
      });
   }
  }

  handleWatch(product: Product) {
   this.orderChart?.connect(product);
  }
}
```

Angular UI Web Application : products.component.html



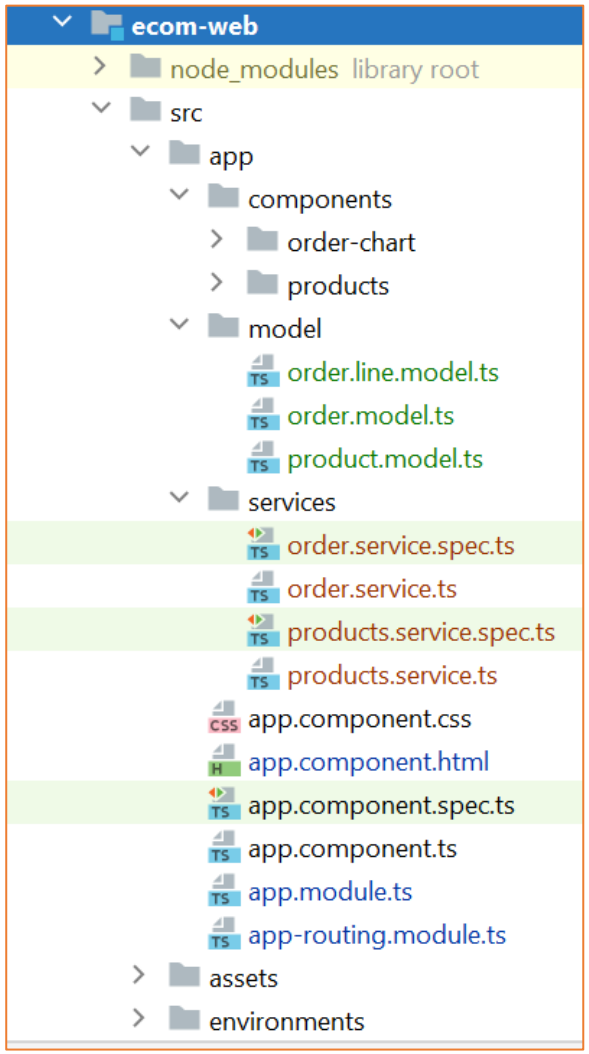

```
<div class="container">
 <div class="row mt-2">
 <div class="col-md-5" *ngIf="products">
  <div class="card">
   <div class="card-header">Stock</div>
   <div class="card-body">
    <table class="table">
     <tr>
      <th>Name</th><th>Price</th><th>Stock</th><th></th>
     </tr>
     <tr *ngFor="let product of products">
      <td>{{product.name}}</td>
      <td>{{product.price}}</td>
      <td>{{product.stock}}</td>
      <td>
       <button class="btn btn-success" (click)="handleNewOrder(product)">Order</button>
      </td>
      <td>
       <button class="btn btn-success" (click)="handleWatch(product)">Watch</button>
      </td>
     </tr>
    </table>
   </div>
  </div>  </div
```

Angular UI Web Application : products.component.html



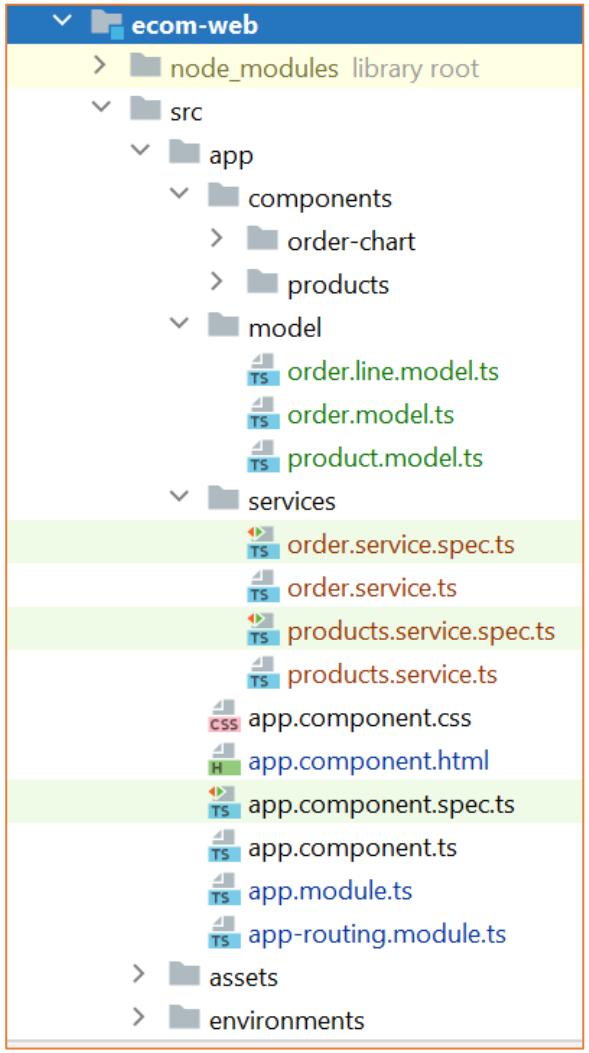

```
<div class="col-md-7" *ngIf="order">
  <div class="text-danger m-2" *ngIf="errorMessage">
   {{errorMessage}}
  </div>
  <div class="card">
   <div class="m-2">
    <button class="btn btn-success" (click)="addNewOrder()">New Order</button>
    <button class="btn btn-success ml-3" (click)="confirmOrder()">Confirm Order</button>
   </div>
   <div class="card-header">
    Status : <button [ngClass]="order.status==='CONFIRMED'?'btn btn-outline-success disabled':'btn btn-outline-danger'">{{order.status}}</button>
    | Order : {{order.orderId}}
   </div>
   <div class="card-body">
```

Angular UI Web Application : products.component.html



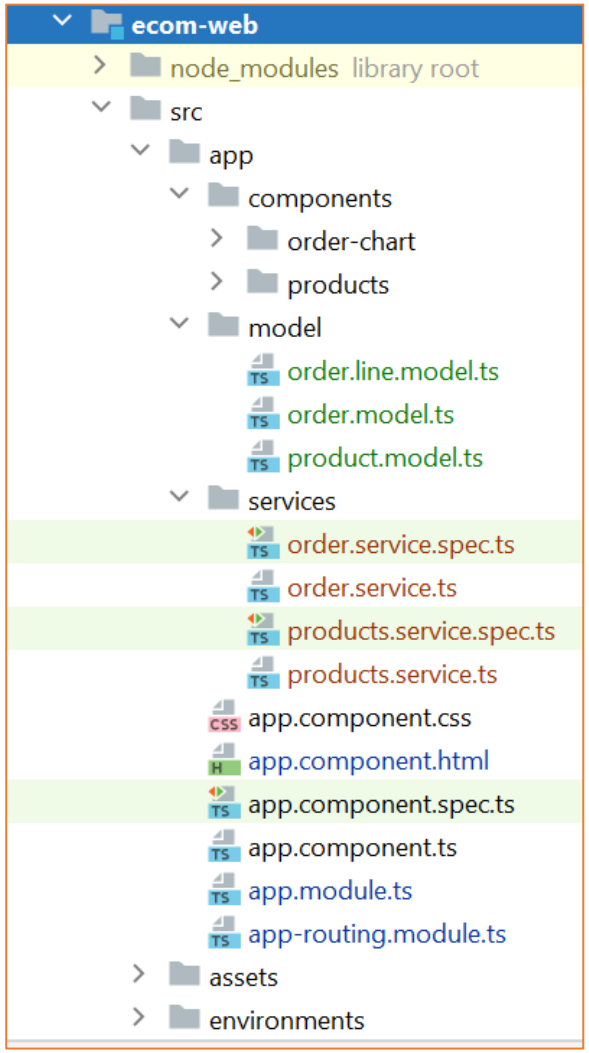

```
<table class="table">
      <tr>
       <th>Product Name</th><th>Price</th><th>Quantity</th>
      </tr>
      <tr *ngFor="let ol of order.orderItems">
       <td>{{ol.productName}}</td>
       <td>{{ol.price}}</td>
       <td>{{ol.quantity}}</td>
       <td>
        <button (click)="decQuantity(ol)">-</button>
         <input [disabled]="order.status==='CONFIRMED'" (change)="handleChangeQuantity(ol)"
type="text" size="2" [(ngModel)]="ol.quantity">
        <button (click)="incQuantity(ol)">+</button>
       </td>
       <td><button class="btn btn-danger" (click)="deleteItem(ol)">X</button></td>
      </tr>
     </table>
    </div>
    <div class="card-header">
     Total : {{orderService.totalOrder(order)|currency}}
    </div>
   </div>
   <app-order-chart></app-order-chart>
 </div>  </div> </div>
```

Angular UI Web Application : order-chart.component.ts



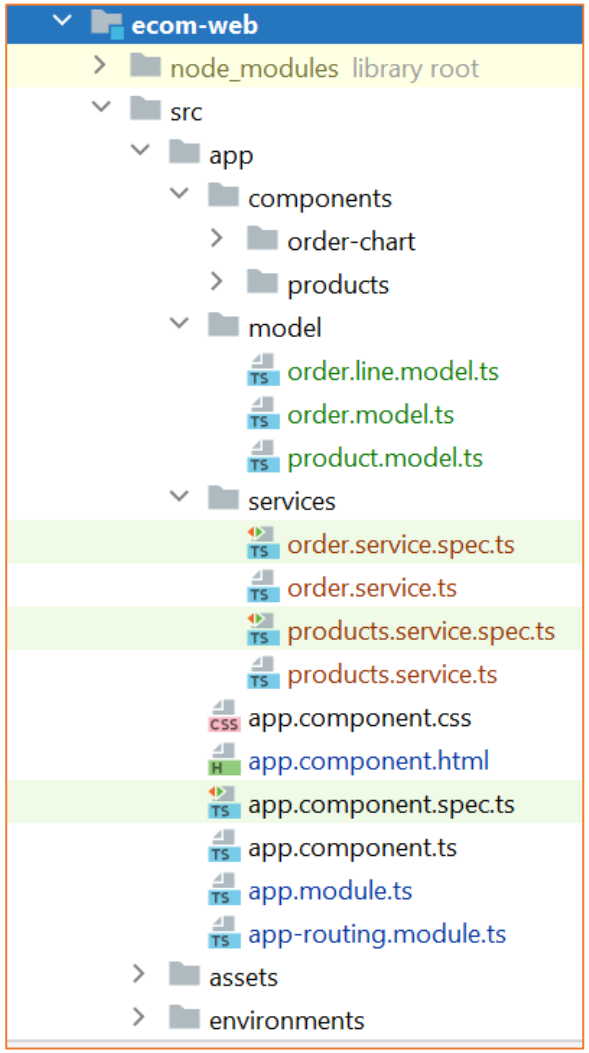

```
@Component({
 selector: 'app-order-chart',
 templateUrl: './order-chart.component.html',
 styleUrls: ['./order-chart.component.css']
})
export class OrderChartComponent implements OnInit {
 source:EventSource|undefined;
 lineChartData : ChartDataSets[] = [
  { data: [], label: 'Product' }
 ];
 lineChartLabels: Label[] = [];
 lineChartOptions = {
  responsive: true,
 };
 lineChartColors : Color[] = [
  {
   borderColor: 'black',
   backgroundColor: 'rgba(255,255,0,0.28)',
  },
 ];
 lineChartLegend = true;
 lineChartPlugins = [];
 lineChartType :ChartType ="line";
```

Angular UI Web Application : order-chart.component.ts




```
constructor() { }
  ngOnInit(): void {
  }
  connect(product:Product):void{
    if(product){
      if(this.source){
        this.source.removeEventListener("message",(evt)=>{
          console.log(evt);
        });
        this.lineChartData[0].data=[];
        this.lineChartLabels=[];
      }
      this.lineChartData[0]={ data: [], label: 'Product' },
      this.lineChartData[0].label=product.name;
      this.source=new EventSource(environment.gatewayHost+"/PRODUCT-
SERVICE/query/products/watchProduct/"+product?.productId);
      this.source.addEventListener("message",message=>{
        let product=JSON.parse(message.data);
        this.lineChartData[0].data?.push(product.stock)
        this.lineChartLabels.push(new Date().getTime().toString())
      });
    }
  }
}
```

Angular UI Web Application : order-chart.component.html




```
<div class="chart-wrapper">
 <canvas baseChart
        [datasets]="lineChartData"
        [labels]="lineChartLabels"
        [options]="lineChartOptions"
        [colors]="lineChartColors"
        [legend]="lineChartLegend"
        [chartType]="lineChartType"
        [plugins]="lineChartPlugins">
 </canvas>
</div>
```

Angular UI Web Application : UI Test