**eShopOnContainers**

Liad Ohana

# 1.Create new order

This part of tests contains all the version and ramifications of create order scenario.

1.1 Create New Order Success Flow  
Test name: Create New Order Success Flow

Purpose test: check the creation of a new order.

Traceability: 1-Creating order

Requirement: 1,3,4

Test environment: identity, UUT- Order, Simulator- basket, catalog, payment

Preconditions: all Simulators connected to RabbitMQ and Order connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
| **Expected Result** | **Description** | **Step** |
| Create new Order entity with OrderStatusID =1 in DB  ordering service send a message to RabbitMQ-> queue:Basket -> Routing key: OrderStartedIntegrationEvent (Output Messages#1)  ordering service sends a message to RabbitMQ-> queue:Ordering.Signalrhub -> Routing key: OrderStatusChangedToSubmittedIntegrationEvent (Output Messages#2)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:   1. Update Order entity Set OrderStatusID =2 in DB 2. a message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) 3. same message has been sent to Ordering.Signalrhub queue. Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4)   message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) | Send message (Input Messages#1) into Basket Simulator :  RabbitMQ -> queue: Ordering -> Routing key : UserCheckoutAcceptedIntegrationEvent | 1 |
| Update Order entity Set OrderStatusID =3 in DB  Ordering sends a message to RabbitMQ-> queue:Payment -> Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent (Output Messages#3)  The same message has been sent to RabbitMQ-> queue: Signalrhub | Send message ( Input Messages#2) into Catalog Simulator :  RabbitMQ -> queue: Catalog-> Routing key : OrderStockConfirmedIntegrationEvent | 2 |
| Update Order entity Set OrderStatusID =4 in DB | Send message (Input Messages#6 ) into Payment Simulator :  RabbitMQ -> queue: Payment -> Routing key :  OrderPaymentSucceededIntegrationEvent | 3 |

## 1.2 Create new order when there is not enough quantity of unit in the stock

Test name: Create new order when there is not enough quantity of unit in the stock.

Purpose test: check the creation of a new order when the stock of items is not enough.

Traceability: 1-Creating order

Requirement:1,4

Test environment: identity ,UUT- Order, Simulator- basket, catalog, payment

Preconditions: all Simulators connected to RabbitMQ and Order connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
| **Expected Result** | **Description** | **Step** |
| Create new Order entity with OrderStatusID =1 in DB  ordering service send a message to RabbitMQ-> queue:Basket -> Routing key: OrderStartedIntegrationEvent (Output Messages#1)  ordering service sends a message to RabbitMQ-> queue:Ordering.Signalrhub -> Routing key: OrderStatusChangedToSubmittedIntegrationEvent (Output Messages#2)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:   1. Update Order entity Set OrderStatusID =2 in DB 2. a message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) 3. same message has been sent to Ordering.Signalrhub queue. Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4)   message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) | Send message (Input Messages#1) into Basket Simulator :  RabbitMQ -> queue: Ordering -> Routing key : UserCheckoutAcceptedIntegrationEvent | 1 |
| a message sent to RabbitMQ-> queue: Signalrhub -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Input Messages#5)  Ordering service gets the message from catalog  Update Order entity Set OrderStatusID = 6 in DB | Send message (Input Messages#3) into Catalog Simulator :  RabbitMQ -> queue: Catalog-> Routing key : OrderStockRejectedIntegrationEvent | 2 |

## 1.3 Create new order when there a problem with the payment

Test name: Create new order when there is a problem with the payment.

Purpose test: check the creation of a new order with problem in payment.

Traceability: 1-Creating order

Requirement:1,3,4

Test environment: identity ,UUT- Order, Simulator- basket, catalog, payment

Preconditions: all Simulators connected to RabbitMQ and Order connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
| **Expected Result** | **Description** | **Step** |
| Create new Order entity with OrderStatusID =1 in DB  ordering service send a message to RabbitMQ-> queue:Basket -> Routing key: OrderStartedIntegrationEvent (Output Messages#1)  ordering service sends a message to RabbitMQ-> queue:Ordering.Signalrhub -> Routing key: OrderStatusChangedToSubmittedIntegrationEvent (Output Messages#2)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:   1. Update Order entity Set OrderStatusID =2 in DB 2. a message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) 3. same message has been sent to Ordering.Signalrhub queue. Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4)   message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) | Send message (Input Messages#1) into Basket Simulator :  RabbitMQ -> queue: Ordering -> Routing key : UserCheckoutAcceptedIntegrationEvent | 1 |
| Update Order entity Set OrderStatusID =3 in DB  ordering service send a message to RabbitMQ-> queue:Ordering -> Routing key: OrderStockConfirmedIntegrationEvent (Input Messages#2)  message sent to RabbitMQ-> queue: Signalrhub -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (input Messages#5)  Ordering sends a message to RabbitMQ-> queue:Payment -> Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent (Output Messages#3)  The same message has been sent to RabbitMQ-> queue: Signalrhub | Send message ( Input Messages#2) into Catalog Simulator :  RabbitMQ -> queue: Catalog-> Routing key : OrderStockConfirmedIntegrationEvent | 2 |
| Ordering sends a message to RabbitMQ-> queue:Payment -> Routing key: OrderPaymentFailedIntegrationEvent (Input Messages#7)  Update Order entity Set OrderStatusID =6 in DB | Send message (Input Messages #7 ) into Payment Simulator :  RabbitMQ -> queue: Payment -> Routing key :  OrderPaymentFailedIntegrationEvent | 3 |

## 1.4 Create order with 0 quantity item

Test name: Create new order with 0 quantity of item

Purpose test: check if new order created when the list of item is contain 0 quantity of item.

Traceability: 1-Creating order

Requirement:1

Test environment: identity ,UUT- Order, Simulator- basket, catalog, payment

Preconditions: all Simulators connected to RabbitMQ and Order connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
| **Expected Result** | **Description** | **Step** |
| Not created new entity of order in DB | Send message (Input Messages#1.1(empty quantity )) into Basket Simulator :  RabbitMQ -> queue: Ordering -> Routing key : UserCheckoutAcceptedIntegrationEvent | 1 |

## 1.5 Create order with empty list of items

Test name: Create new order with empty list of items

Purpose test: check if new order created when the list of item is empty.

Traceability: 1-Creating order

Requirement:1

Test environment: identity ,UUT- Order, Simulator- basket, catalog, payment

Preconditions: all Simulators connected to RabbitMQ and Order connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
| **Expected Result** | **Description** | **Step** |
| Not created new entity of order in DB | Send message (Input Messages#1.2(empty list)) into Basket Simulator :  RabbitMQ -> queue: Ordering -> Routing key : UserCheckoutAcceptedIntegrationEvent | 1 |

# 2. Cancelling order

This part of tests contain all the version and ramifications of cancelling order scenario.

## 2.1 Canceling order in status submitted

Test name: Canceling order in status 1

Purpose test: check the ability of canceled order in status 1 .

Traceability: 1- Canceling order

Requirement:1

Test environment: RabbitMQ , identity ,UUT- Order, Simulator- basket, catalog, payment , BackgroundTasks is off

Preconditions: all Simulators connected to RabbitMQ and Order connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
| **Expected Result** | **Description** | **Step** |
| Create new Order entity with OrderStatusID =1 in DB  ordering service send a message to RabbitMQ-> queue:Basket -> Routing key: OrderStartedIntegrationEvent (Output Messages#1)  ordering service sends a message to RabbitMQ-> queue:Ordering.Signalrhub -> Routing key: OrderStatusChangedToSubmittedIntegrationEvent (Output Messages#2) | Send message (Input Messages#1) into Basket Simulator :  RabbitMQ -> queue: Ordering -> Routing key : UserCheckoutAcceptedIntegrationEvent | 1 |
| Status code 200  Update Order entity Set OrderStatusID =6 in DB | Send API message into Postman  ( Postman message #2) | 2 |

## 2.2 Canceling order in status awaitingvalidation

Test name: Canceling order in status 2

Purpose test: check the ability of canceled order in status 2 .

Traceability: 1- Canceling order

Requirement:1

Test environment: RabbitMQ , identity ,UUT- Order, Simulator- basket, catalog, payment

Preconditions: all Simulators connected to RabbitMQ and Order connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
| **Expected Result** | **Description** | **Step** |
| Create new Order entity with OrderStatusID =1 in DB  ordering service send a message to RabbitMQ-> queue:Basket -> Routing key: OrderStartedIntegrationEvent (Output Messages#1)  ordering service sends a message to RabbitMQ-> queue:Ordering.Signalrhub -> Routing key: OrderStatusChangedToSubmittedIntegrationEvent (Output Messages#2)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:   1. Update Order entity Set OrderStatusID =2 in DB 2. a message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) 3. same message has been sent to Ordering.Signalrhub queue. Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4)   message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) | Send message (Input Messages#1) into Basket Simulator :  RabbitMQ -> queue: Ordering -> Routing key : UserCheckoutAcceptedIntegrationEvent | 1 |
| Status code 200  Update Order entity Set OrderStatusID =6 in DB | Send API message into Postman  ( Postman message #2) | 2 |

## 2.3 Canceling order in status stockconfirmed

Test name: Canceling order in status 3

Purpose test: check the ability of canceled order in status 3 .

Traceability: 1- Canceling order

Requirement:1

Test environment:RabbitMQ , identity ,UUT- Order, Simulator- basket, catalog, payment

Preconditions: all Simulators connected to RabbitMQ and Order connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
| **Expected Result** | **Description** | **Step** |
| Create new Order entity with OrderStatusID =1 in DB  ordering service send a message to RabbitMQ-> queue:Basket -> Routing key: OrderStartedIntegrationEvent (Output Messages#1)  ordering service sends a message to RabbitMQ-> queue:Ordering.Signalrhub -> Routing key: OrderStatusChangedToSubmittedIntegrationEvent (Output Messages#2)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:   1. Update Order entity Set OrderStatusID =2 in DB 2. a message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) 3. same message has been sent to Ordering.Signalrhub queue. Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4)   message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) | Send message (Input Messages#1) into Basket Simulator :  RabbitMQ -> queue: Ordering -> Routing key : UserCheckoutAcceptedIntegrationEvent | 1 |
| Update Order entity Set OrderStatusID =3 in DB  ordering service send a message to RabbitMQ-> queue:Ordering -> Routing key: OrderStockConfirmedIntegrationEvent (Input Messages#2)  message sent to RabbitMQ-> queue: Signalrhub -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (input Messages#5)  Ordering sends a message to RabbitMQ-> queue:Payment -> Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent (Output Messages#3)  The same message has been sent to RabbitMQ-> queue: Signalrhub | Send message ( Input Messages#2) into Catalog Simulator :  RabbitMQ -> queue: Catalog-> Routing key : OrderStockConfirmedIntegrationEvent | 2 |
| Status code 200  Update Order entity Set OrderStatusID =6 in DB | Send API message into Postman  ( Postman message #2) | 3 |

## 2.4 Canceling order in status paid

Test name: Canceling order in status 4

Purpose test: check the ability of canceled order in status 4 .

Traceability: 1- Canceling order

Requirement:1

Test environment:RabbitMQ , identity ,UUT- Order, Simulator- basket, catalog, payment

Preconditions: all Simulators connected to RabbitMQ and Order connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
| **Expected Result** | **Description** | **Step** |
| Create new Order entity with OrderStatusID =1 in DB  ordering service send a message to RabbitMQ-> queue:Basket -> Routing key: OrderStartedIntegrationEvent (Output Messages#1)  ordering service sends a message to RabbitMQ-> queue:Ordering.Signalrhub -> Routing key: OrderStatusChangedToSubmittedIntegrationEvent (Output Messages#2)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:   1. Update Order entity Set OrderStatusID =2 in DB 2. a message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) 3. same message has been sent to Ordering.Signalrhub queue. Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4)   message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) | Send message (Input Messages#1) into Basket Simulator :  RabbitMQ -> queue: Ordering -> Routing key : UserCheckoutAcceptedIntegrationEvent | 1 |
| Update Order entity Set OrderStatusID =3 in DB  Ordering sends a message to RabbitMQ-> queue:Payment -> Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent (Output Messages#3)  The same message has been sent to RabbitMQ-> queue: Signalrhub | Send message ( Input Messages#2) into Catalog Simulator :  RabbitMQ -> queue: Catalog-> Routing key : OrderStockConfirmedIntegrationEvent | 2 |
| Update Order entity Set OrderStatusID =4 in DB | Send message (Input Messages#6 ) into Payment Simulator :  RabbitMQ -> queue: Payment -> Routing key :  OrderPaymentSucceededIntegrationEvent | 3 |
| Status code 400 Bad Request  Order status not change in DB  ( OrderStatusID =4) | Send API message into Postman  ( Postman message #2) | 4 |

## 2.5 Canceling order in status shipped

Test name: Canceling order in status 5

Purpose test: check the ability of canceled order in status 5.

Traceability: 1- Canceling order

Requirement:1

Test environment: RabbitMQ , identity ,UUT- Order, Simulator- basket, catalog, payment

Preconditions: all Simulators connected to RabbitMQ and Order connected to RabbitMQ

|  |  |  |  |
| --- | --- | --- | --- |
| **Actual Result** | **Expected Result** | **Description** | **Step** |
|  | Create new Order entity with OrderStatusID =1 in DB  ordering service send a message to RabbitMQ-> queue:Basket -> Routing key: OrderStartedIntegrationEvent (Output Messages#1)  ordering service sends a message to RabbitMQ-> queue:Ordering.Signalrhub -> Routing key: OrderStatusChangedToSubmittedIntegrationEvent (Output Messages#2)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:   1. Update Order entity Set OrderStatusID =2 in DB 2. a message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) 3. same message has been sent to Ordering.Signalrhub queue. Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4)   message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) | Send message (Input Messages#1) into Basket Simulator :  RabbitMQ -> queue: Ordering -> Routing key : UserCheckoutAcceptedIntegrationEvent | 1 |
|  | Update Order entity Set OrderStatusID =3 in DB  Ordering sends a message to RabbitMQ-> queue:Payment -> Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent (Output Messages#3)  The same message has been sent to RabbitMQ-> queue: Signalrhub | Send message ( Input Messages#2) into Catalog Simulator :  RabbitMQ -> queue: Catalog-> Routing key : OrderStockConfirmedIntegrationEvent | 2 |
|  | Update Order entity Set OrderStatusID =4 in DB | Send message (Input Messages#6 ) into Payment Simulator :  RabbitMQ -> queue: Payment -> Routing key :  OrderPaymentSucceededIntegrationEvent | 3 |
|  | Status code 200  Update Order entity Set OrderStatusID =5 in DB | Send API message into Postman  ( Postman message #1(ship)) | 4 |
|  | Status code 400 Bad Request  Order status not change in DB  ( OrderStatusID =5) | Send API message into Postman  ( Postman message #2) | 5 |

# 3. Update order

This part of tests contain all the version and ramifications of update order scenario.  
Update order is divided into 2 parts:  
1.Cancelling order (part 2 in this file)

2.Update order status to shipped

So in this part I provide all tests that connect to update order status to shipped

## 3.1 Update order from status submitted to shipped

Test name: succeed to update status from 1 to 5

Purpose test: check the ability of update status of order.

Traceability: 1- Update order

Requirement:1

Test environment: RabbitMQ ,UUT- Order, Simulator- basket, catalog, payment , identity

Preconditions: all Simulators connected to RabbitMQ and Order connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
| **Expected Result** | **Description** | **Step** |
| Create new Order entity with OrderStatusID =1 in DB  ordering service send a message to RabbitMQ-> queue:Basket -> Routing key: OrderStartedIntegrationEvent (Output Messages#1)  ordering service sends a message to RabbitMQ-> queue:Ordering.Signalrhub -> Routing key: OrderStatusChangedToSubmittedIntegrationEvent (Output Messages#2) | Send message (Input Messages#1) into Basket Simulator :  RabbitMQ -> queue: Ordering -> Routing key : UserCheckoutAcceptedIntegrationEvent | 1 |
| Status code 400 Bad Request  Order entity Set OrderStatusID =1 in DB | Send API message into Postman  ( Postman message #1(ship))) | 2 |

## 3.2 Update order from status awaitingvalidation to shipped

Test name: succeed to update status from 2 to 5

Purpose test: check the ability of update status of order.

Traceability: 1- Update order

Requirement:1

Test environment: RabbitMQ, UUT- Order, Simulator- basket, catalog, payment , identity

Preconditions: all Simulators connected to RabbitMQ and Order connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
| **Expected Result** | **Description** | **Step** |
| Create new Order entity with OrderStatusID =1 in DB  ordering service send a message to RabbitMQ-> queue:Basket -> Routing key: OrderStartedIntegrationEvent (Output Messages#1)  ordering service sends a message to RabbitMQ-> queue:Ordering.Signalrhub -> Routing key: OrderStatusChangedToSubmittedIntegrationEvent (Output Messages#2)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:   1. Update Order entity Set OrderStatusID =2 in DB 2. a message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) 3. same message has been sent to Ordering.Signalrhub queue. Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4)   message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) | Send message (Input Messages#1) into Basket Simulator :  RabbitMQ -> queue: Ordering -> Routing key : UserCheckoutAcceptedIntegrationEvent | 1 |
| Status code 400 Bad Request  Order status not change in DB  OrderStatusID =2 | Send API message into Postman  ( Postman message #1(ship)) | 2 |

## 3.3 Update order from status stockconfirmed to shipped

Test name: succeed to update status from 3 to 5

Purpose test: check the ability of update status of order.

Traceability: 1- Update order

Requirement:1

Test environment: RabbitMQ, UUT- Order, Simulator- basket, catalog, payment, identity

Preconditions: all Simulators connected to RabbitMQ and Order connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
| **Expected Result** | **Description** | **Step** |
| Create new Order entity with OrderStatusID =1 in DB  ordering service send a message to RabbitMQ-> queue:Basket -> Routing key: OrderStartedIntegrationEvent (Output Messages#1)  ordering service sends a message to RabbitMQ-> queue:Ordering.Signalrhub -> Routing key: OrderStatusChangedToSubmittedIntegrationEvent (Output Messages#2)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:   1. Update Order entity Set OrderStatusID =2 in DB 2. a message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) 3. same message has been sent to Ordering.Signalrhub queue. Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4)   message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) | Send message (Input Messages#1) into Basket Simulator :  RabbitMQ -> queue: Ordering -> Routing key : UserCheckoutAcceptedIntegrationEvent | 1 |
| Update Order entity Set OrderStatusID =3 in DB  Ordering sends a message to RabbitMQ-> queue:Payment -> Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent (Output Messages#3)  The same message has been sent to RabbitMQ-> queue: Signalrhub | Send message ( Input Messages#2) into Catalog Simulator :  RabbitMQ -> queue: Catalog-> Routing key : OrderStockConfirmedIntegrationEvent | 2 |
| Status code 400 Bad Request  Order entity Set OrderStatusID =3 in DB | Send API message into Postman  ( Postman message #1(ship)) | 4 |

## 3.4 Update order from status paid to shipped

Test name: succeed to update status from 4 to 5

Purpose test: check the ability of update status of order.

Traceability: 1- Update order

Requirement:1

Test environment: RabbitMQ, UUT- Order, Simulator- basket, catalog, payment, identity

Preconditions: all Simulators connected to RabbitMQ and Order connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
| **Expected Result** | **Description** | **Step** |
| Create new Order entity with OrderStatusID =1 in DB  ordering service send a message to RabbitMQ-> queue:Basket -> Routing key: OrderStartedIntegrationEvent (Output Messages#1)  ordering service sends a message to RabbitMQ-> queue:Ordering.Signalrhub -> Routing key: OrderStatusChangedToSubmittedIntegrationEvent (Output Messages#2)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:   1. Update Order entity Set OrderStatusID =2 in DB 2. a message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) 3. same message has been sent to Ordering.Signalrhub queue. Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4)   message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) | Send message (Input Messages#1) into Basket Simulator :  RabbitMQ -> queue: Ordering -> Routing key : UserCheckoutAcceptedIntegrationEvent | 1 |
| Update Order entity Set OrderStatusID =3 in DB  Ordering sends a message to RabbitMQ-> queue:Payment -> Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent (Output Messages#3)  The same message has been sent to RabbitMQ-> queue: Signalrhub | Send message ( Input Messages#2) into Catalog Simulator :  RabbitMQ -> queue: Catalog-> Routing key : OrderStockConfirmedIntegrationEvent | 2 |
| Update Order entity Set OrderStatusID =4 in DB | Send message (Input Messages#6 ) into Payment Simulator :  RabbitMQ -> queue: Payment -> Routing key :  OrderPaymentSucceededIntegrationEvent | 3 |
| Status code 200  Update Order entity Set OrderStatusID =5 in DB | Send API message into Postman  ( Postman message #1(ship)) | 4 |

## 3.5 Update order from status cancelled to shipped

Test name: succeed to update status from 6 to 5

Purpose test: check the ability of update status of order.

Traceability: 1- Update order

Requirement:1

Test environment:RabbitMQ, UUT- Order, Simulator- basket, catalog, payment, identity

Preconditions: all Simulators connected to RabbitMQ and Order connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
| **Expected Result** | **Description** | **Step** |
| Create new Order entity with OrderStatusID =1 in DB  ordering service send a message to RabbitMQ-> queue:Basket -> Routing key: OrderStartedIntegrationEvent (Output Messages#1)  ordering service sends a message to RabbitMQ-> queue:Ordering.Signalrhub -> Routing key: OrderStatusChangedToSubmittedIntegrationEvent (Output Messages#2)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:   1. Update Order entity Set OrderStatusID =2 in DB 2. a message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) 3. same message has been sent to Ordering.Signalrhub queue. Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4)   message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) | Send message (Input Messages#1) into Basket Simulator :  RabbitMQ -> queue: Ordering -> Routing key : UserCheckoutAcceptedIntegrationEvent | 1 |
| Status code 200  Update Order entity Set OrderStatusID =6 in DB  and the process is done here | Send API message (PUT) with Postman to Ordering -> input message (x):  x-requestid=uuid request body –  {  "orderNumber":x  }  ( Postman message #2) | 2 |
| Status code 400 Bad Request  Order status not change in DB  OrderStatusID =6 | Send API message into Postman  ( Postman message #1(ship)) | 3 |

# 4. Order Tracking

This part of tests contain all the version and ramifications of order tracking scenario.

## 4.1 Order tracking in status submitted

Test name: Order Tracking in status submitted (1)

Purpose test: check the online tracking of order in status 1 .

Traceability: 1- Tracking order

Requirement:1

Test environment:RabbitMQ , identity ,UUT- Order, Simulator- basket, catalog, payment , BackgroundTasks is off

Preconditions: all Simulators connected to RabbitMQ and Order connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
| **Expected Result** | **Description** | **Step** |
| Create new Order entity with OrderStatusID =1 in DB  ordering service send a message to RabbitMQ-> queue:Basket -> Routing key: OrderStartedIntegrationEvent (Output Messages#1)  ordering service sends a message to RabbitMQ-> queue:Ordering.Signalrhub -> Routing key: OrderStatusChangedToSubmittedIntegrationEvent (Output Messages#2) | Send message (Input Messages#1) into Basket Simulator :  RabbitMQ -> queue: Ordering -> Routing key : UserCheckoutAcceptedIntegrationEvent | 1 |
| Status code 200  Return Order details with OrderStatusID = submitted | Send API message into Postman  ( Postman message #3) | 2 |

## 4.2 Order tracking in status awaitingvalidation

Test name: Order Tracking in status awaitingvalidation (2)

Purpose test: check the online tracking of order in status 2.

Traceability: 1- Tracking order

Requirement:1

Test environment: RabbitMQ , identity ,UUT- Order, Simulator- basket, catalog, payment

Preconditions: all Simulators connected to RabbitMQ and Order connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
| **Expected Result** | **Description** | **Step** |
| Create new Order entity with OrderStatusID =1 in DB  ordering service send a message to RabbitMQ-> queue:Basket -> Routing key: OrderStartedIntegrationEvent (Output Messages#1)  ordering service sends a message to RabbitMQ-> queue:Ordering.Signalrhub -> Routing key: OrderStatusChangedToSubmittedIntegrationEvent (Output Messages#2)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:   1. Update Order entity Set OrderStatusID =2 in DB 2. a message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) 3. same message has been sent to Ordering.Signalrhub queue. Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4)   message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) | Send message (Input Messages#1) into Basket Simulator :  RabbitMQ -> queue: Ordering -> Routing key : UserCheckoutAcceptedIntegrationEvent | 1 |
| Status code 200  Return Order details with OrderStatusID = awaitingvalidation | Send API message into Postman  ( Postman message #3) | 2 |

## 4.3 Order tracking in status stockconfirmed

Test name: Order Tracking in status stockconfirmed (3)

Purpose test: check the online tracking of order in status 3.

Traceability: 1- Tracking order

Requirement:1

Test environment:RabbitMQ , identity ,UUT- Order, Simulator- basket, catalog, payment

Preconditions: all Simulators connected to RabbitMQ and Order connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
| **Expected Result** | **Description** | **Step** |
| Create new Order entity with OrderStatusID =1 in DB  ordering service send a message to RabbitMQ-> queue:Basket -> Routing key: OrderStartedIntegrationEvent (Output Messages#1)  ordering service sends a message to RabbitMQ-> queue:Ordering.Signalrhub -> Routing key: OrderStatusChangedToSubmittedIntegrationEvent (Output Messages#2)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:   1. Update Order entity Set OrderStatusID =2 in DB 2. a message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) 3. same message has been sent to Ordering.Signalrhub queue. Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4)   message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) | Send message (Input Messages#1) into Basket Simulator :  RabbitMQ -> queue: Ordering -> Routing key : UserCheckoutAcceptedIntegrationEvent | 1 |
| Update Order entity Set OrderStatusID =3 in DB  ordering service send a message to RabbitMQ-> queue:Ordering -> Routing key: OrderStockConfirmedIntegrationEvent (Input Messages#2)  message sent to RabbitMQ-> queue: Signalrhub -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (input Messages#5)  Ordering sends a message to RabbitMQ-> queue:Payment -> Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent (Output Messages#3)  The same message has been sent to RabbitMQ-> queue: Signalrhub | Send message ( Input Messages#2) into Catalog Simulator :  RabbitMQ -> queue: Catalog-> Routing key : OrderStockConfirmedIntegrationEvent | 2 |
| Status code 200  Return Order details with OrderStatusID = stockconfirmed | Send API message into Postman  ( Postman message #3) | 3 |

## 4.4 Order tracking in status paid

Test name: Order Tracking in status paid (4)

Purpose test: check the online tracking of order in status 4.

Traceability: 1- Tracking order

Requirement:1

Test environment: RabbitMQ , identity ,UUT- Order, Simulator- basket, catalog, payment

Preconditions: all Simulators connected to RabbitMQ and Order connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
| **Expected Result** | **Description** | **Step** |
| Create new Order entity with OrderStatusID =1 in DB  ordering service send a message to RabbitMQ-> queue:Basket -> Routing key: OrderStartedIntegrationEvent (Output Messages#1)  ordering service sends a message to RabbitMQ-> queue:Ordering.Signalrhub -> Routing key: OrderStatusChangedToSubmittedIntegrationEvent (Output Messages#2)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:   1. Update Order entity Set OrderStatusID =2 in DB 2. a message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) 3. same message has been sent to Ordering.Signalrhub queue. Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4)   message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) | Send message (Input Messages#1) into Basket Simulator :  RabbitMQ -> queue: Ordering -> Routing key : UserCheckoutAcceptedIntegrationEvent | 1 |
| Update Order entity Set OrderStatusID =3 in DB  Ordering sends a message to RabbitMQ-> queue:Payment -> Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent (Output Messages#3)  The same message has been sent to RabbitMQ-> queue: Signalrhub | Send message ( Input Messages#2) into Catalog Simulator :  RabbitMQ -> queue: Catalog-> Routing key : OrderStockConfirmedIntegrationEvent | 2 |
| Update Order entity Set OrderStatusID =4 in DB and process is done here | Send message (Input Messages#6 ) into Payment Simulator :  RabbitMQ -> queue: Payment -> Routing key :  OrderPaymentSucceededIntegrationEvent | 3 |
| Status code 200  Return Order details with OrderStatusID = paid | Send API message into Postman  ( Postman message #3) | 4 |

## 4.5 Order tracking in status shipped

Test name: Order Tracking in status shipped (5)

Purpose test: check the online tracking of order in status 5.

Traceability: 1- Tracking order

Requirement:1

Test environment: RabbitMQ, identity ,UUT- Order, Simulator- basket, catalog, payment

Preconditions: all Simulators connected to RabbitMQ and Order connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
| **Expected Result** | **Description** | **Step** |
| Create new Order entity with OrderStatusID =1 in DB  ordering service send a message to RabbitMQ-> queue:Basket -> Routing key: OrderStartedIntegrationEvent (Output Messages#1)  ordering service sends a message to RabbitMQ-> queue:Ordering.Signalrhub -> Routing key: OrderStatusChangedToSubmittedIntegrationEvent (Output Messages#2)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:   1. Update Order entity Set OrderStatusID =2 in DB 2. a message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) 3. same message has been sent to Ordering.Signalrhub queue. Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4)   message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) | Send message (Input Messages#1) into Basket Simulator :  RabbitMQ -> queue: Ordering -> Routing key : UserCheckoutAcceptedIntegrationEvent | 1 |
| Update Order entity Set OrderStatusID =3 in DB  Ordering sends a message to RabbitMQ-> queue:Payment -> Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent (Output Messages#3)  The same message has been sent to RabbitMQ-> queue: Signalrhub | Send message ( Input Messages#2) into Catalog Simulator :  RabbitMQ -> queue: Catalog-> Routing key : OrderStockConfirmedIntegrationEvent | 2 |
| Update Order entity Set OrderStatusID =4 in DB and process is done here | Send message (Input Messages#6 ) into Payment Simulator :  RabbitMQ -> queue: Payment -> Routing key :  OrderPaymentSucceededIntegrationEvent | 3 |
| Status code 200  Update Order entity Set OrderStatusID =5 in DB | Send API message into Postman  ( Postman message #1 (ship)) | 4 |
| Status code 200  Return Order details with OrderStatusID = shipped | Send API message into Postman  ( Postman message #3 (order)) | 5 |

## 4.6 Order tracking in status cancelled

Test name: Order Tracking in status cancelled (6)

Purpose test: check the online tracking of order in status 6.

Traceability: 1- Tracking order

Requirement:1

Test environment:RabbitMQ, identity ,UUT- Order, Simulator- basket, catalog, payment

Preconditions: all Simulators connected to RabbitMQ and Order connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
| **Expected Result** | **Description** | **Step** |
| Create new Order entity with OrderStatusID =1 in DB  ordering service send a message to RabbitMQ-> queue:Basket -> Routing key: OrderStartedIntegrationEvent (Output Messages#1)  ordering service sends a message to RabbitMQ-> queue:Ordering.Signalrhub -> Routing key: OrderStatusChangedToSubmittedIntegrationEvent (Output Messages#2)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:   1. Update Order entity Set OrderStatusID =2 in DB 2. a message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) 3. same message has been sent to Ordering.Signalrhub queue. Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4)   message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) | Send message (Input Messages#1) into Basket Simulator :  RabbitMQ -> queue: Ordering -> Routing key : UserCheckoutAcceptedIntegrationEvent | 1 |
| Update Order entity Set OrderStatusID =3 in DB  ordering service send a message to RabbitMQ-> queue:Ordering -> Routing key: OrderStockConfirmedIntegrationEvent (Input Messages#2)  message sent to RabbitMQ-> queue: Signalrhub -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (input Messages#5)  Ordering sends a message to RabbitMQ-> queue:Payment -> Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent (Output Messages#3)  The same message has been sent to RabbitMQ-> queue: Signalrhub | Send message ( Input Messages#2) into Catalog Simulator :  RabbitMQ -> queue: Catalog-> Routing key : OrderStockConfirmedIntegrationEvent | 2 |
| Update Order entity Set OrderStatusID =6 in DB  and the process is done here | Send API message into Postman  ( Postman message #2(canceled)) | 3 |
| Return Order details with OrderStatusID = shipped | Send API message into Postman  ( Postman message #3 (order)) | 4 |

# 5. Security test

This part of tests contain all the version and ramifications of security test scenario.

## 5.1 Get order by id in status Unauthorized

Test name: get order by order number in status Unauthorized

Purpose test: check if it possibly to get data about order without logging into the system

Traceability: Security

Requirement:5

Test environment: Postman, RabbitMQ, identity , UUT- Order, Simulator- basket, catalog, payment

Preconditions: all Simulators connected to RabbitMQ and Order connected to RabbitMQ and DB contain order in Orders table

|  |  |  |
| --- | --- | --- |
| **Expected Result** | **Description** | **Step** |
| Status code 401 Unauthorized | Send API message into Postman  (Postman message #3) | 1 |

## 5.2 Cancel order by id in status Unauthorized

Test name: cancel order by order number in status Unauthorized

Purpose test: check if it possibly to cancel order without logging into the system

Traceability: Security

Requirement:5

Test environment: Postman, RabbitMQ, identity , UUT- Order, Simulator- basket, catalog, payment

Preconditions: all Simulators connected to RabbitMQ and Order connected to RabbitMQ and DB contain order is status 1/2/3 in Orders table

|  |  |  |
| --- | --- | --- |
| **Expected Result** | **Description** | **Step** |
| Status code 401 Unauthorized | Send API message into Postman  (Postman message #2) | 1 |

## 5.3 Update status order by id in status Unauthorized

Test name: update order by order number in status Unauthorized

Purpose test: check if it possibly to update order without logging into the system

Traceability: Security

Requirement:5

Test environment: Postman, RabbitMQ, identity , UUT- Order, Simulator- basket, catalog, payment

Preconditions: all Simulators connected to RabbitMQ and Order connected to RabbitMQ and DB contain order is status 4 in Orders table

|  |  |  |
| --- | --- | --- |
| **Expected Result** | **Description** | **Step** |
| Status code 401 Unauthorized | Send API message into Postman  (Postman message #1) | 1 |

## 5.4 Canceling another user's order

Test name: Canceling another user's order

Purpose test: Check if it is possible to cancel another user's order

Traceability: Security

Requirement:5

Test environment: Postman, RabbitMQ, identity , UUT- Order, Simulator- basket, catalog, payment

Preconditions: all Simulators connected to RabbitMQ and Order connected to RabbitMQ and DB contain order is status 1/2/3 in Orders table with buyerID=11

|  |  |  |  |
| --- | --- | --- | --- |
| **Actual Result** | **Expected Result** | **Description** | **Step** |
|  | Alice connected into the system | Login to the system with the following details : Username : alice  Password: Pass123$ | 1 |
| Status code 200 Update Order entity Set OrderStatusID =6 in DB | Status code 403 Forbidden   Order entity OrderStatusID =1/2/3 in DB | Send API message into Postman  (Postman message #2) {  "orderNumber": order id(bob)  } | 2 |





## 5.5 Update another user's order

Test name: update another user's order

Purpose test: Check if it is possible to update another user's order

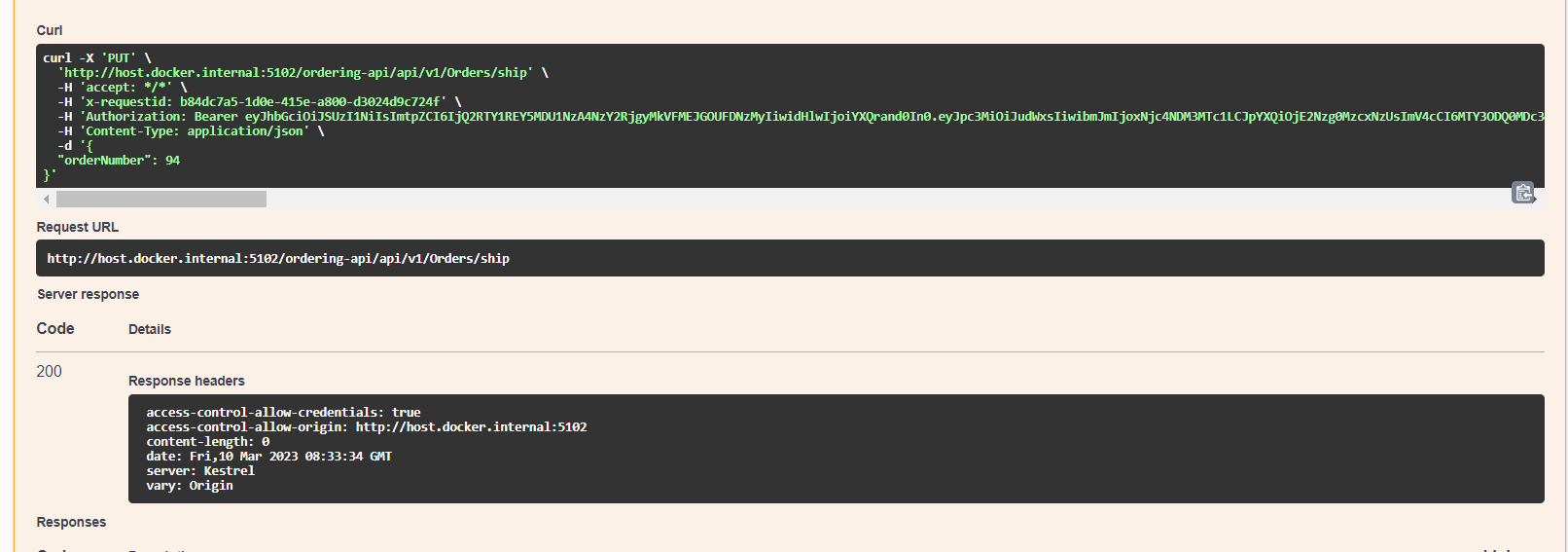
Traceability: Security

Requirement:5

Test environment: Postman, RabbitMQ, identity , UUT- Order, Simulator- basket, catalog, payment

Preconditions: all Simulators connected to RabbitMQ and Order connected to RabbitMQ and DB contain order is status 4 in Orders table with BuyerId=11

|  |  |  |  |
| --- | --- | --- | --- |
| **Actual Result** | **Expected Result** | **Description** | **Step** |
|  | Alice connected into the system | Login to the system with the following details : Username : alice  Password: Pass123$ | 1 |
| Status code 200 Update Order entity Set OrderStatusID =5 in DB | Status code 403 Forbidden   Order entity OrderStatusID =4 in DB | Send API message into Postman  (Postman message #1) {  "orderNumber": order id(bob)  } | 2 |



Status 200

## 5.6 Get order details of another user's order

Test name: get order details of another user's order

Purpose test: Check if it is possible to get order details of another user's order

Traceability: Security

Requirement:5

Test environment: Postman, RabbitMQ, identity , UUT- Order, Simulator- basket, catalog, payment

Preconditions: all Simulators connected to RabbitMQ and Order connected to RabbitMQ and DB contain order of user (userID=11) in Orders table

|  |  |  |  |
| --- | --- | --- | --- |
| **Actual Result** | **Expected Result** | **Description** | **Step** |
|  | Alice connected into the system | Login to the system with the following details : Username : alice  Password: Pass123$ | 1 |
| Status code 200 get order details. | Status code 403 Forbidden | Send API message into Postman  (Postman message #3) {  "orderNumber": order id(bob)  } | 2 |

## 5.7 create order with id of first user and name of second user

Status 200

Test name: create order with id of first user and name of second user

Purpose test: Check if it is possible to create order with id of first user and name of second user

Traceability: Security

Requirement:5

Test environment: Postman, RabbitMQ, identity , UUT- Order, Simulator- basket, catalog, payment

Preconditions: all Simulators connected to RabbitMQ and Order connected to RabbitMQ

|  |  |  |  |
| --- | --- | --- | --- |
| **Actual Result** | **Expected Result** | **Description** | **Step** |
| Create new Order entity in DB | Order not created in DB | Send message (Input Messages#1.3) into Basket Simulator :  RabbitMQ -> queue: Ordering -> Routing key : UserCheckoutAcceptedIntegrationEvent | 1 |





# 6. Scalability test

This part of tests contain all the version and ramifications of Scalability test scenario

## 6.1 Load of message

Test name: load 100 message

Purpose test: check the ability of system to contest with load of message.

Traceability: Scalability

Requirement:

Test environment: RabbitMQ, identity ,UUT- Order, basket, catalog, payment , Order is off,

Preconditions: all Simulators connected to RabbitMQ and Order service is off, in DB orders table is empty

|  |  |  |
| --- | --- | --- |
| **Expected Result** | **Description** | **Step** |
| ordering service get 100 messages from RabbitMQ-> queue: ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent ()  the queue of ordering contains 100 message | Send 100 different message (Input Messages#1) into Basket Simulator :  RabbitMQ -> queue: Ordering -> Routing key : UserCheckoutAcceptedIntegrationEvent | 1 |
| Check in DB if 100 new orders have been created | Tern on the order server | 2 |

# 7. Reliability test

This part of tests contain all the version and ramifications of Reliability test scenario

## 7.1 Recovery Test in status awaitingvalidation

Test name: Recover uut in action in status awaitingvalidation (2)

Purpose test: check the ability of system to recover from fall.

Traceability: Recovery

Requirement:7

Test environment:RabbitMQ, identity ,UUT- Order, Simulator- basket, catalog, payment

Preconditions: all Simulators connected to RabbitMQ and Order service is off

|  |  |  |
| --- | --- | --- |
| **Expected Result** | **Description** | **Step** |
| Create new Order entity with OrderStatusID =1 in DB  ordering service send a message to RabbitMQ-> queue:Basket -> Routing key: OrderStartedIntegrationEvent (Output Messages#1)  ordering service sends a message to RabbitMQ-> queue:Ordering.Signalrhub -> Routing key: OrderStatusChangedToSubmittedIntegrationEvent (Output Messages#2)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:   1. Update Order entity Set OrderStatusID =2 in DB 2. a message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) 3. same message has been sent to Ordering.Signalrhub queue. Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4)   message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) | Send message (Input Messages#1) into Basket Simulator :  RabbitMQ -> queue: Ordering -> Routing key : UserCheckoutAcceptedIntegrationEvent | 1 |
| Order service is not connected | Tern off Order service | 2 |
| Order service connected and the process is continued | After 5 second Tern on Order service | 3 |
| Update Order entity Set OrderStatusID =3 in DB  Ordering sends a message to RabbitMQ-> queue:Payment -> Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent (Output Messages#3)  The same message has been sent to RabbitMQ-> queue: Signalrhub | Send message ( Input Messages#2) into Catalog Simulator :  RabbitMQ -> queue: Catalog-> Routing key : OrderStockConfirmedIntegrationEvent | 4 |
| Update Order entity Set OrderStatusID =4 in DB | Send message (Input Messages#6 ) into Payment Simulator :  RabbitMQ -> queue: Payment -> Routing key :  OrderPaymentSucceededIntegrationEvent | 5 |

## 7.2 Recovery Test in status stockconfirmed

Test name: Recover uut in action in status stockconfirmed (3)

Purpose test: check the ability of system to recover from fall.

Traceability: Recovery

Requirement:7

Test environment: RabbitMQ, identity ,UUT- Order, Simulator- basket, catalog, payment

Preconditions: all Simulators connected to RabbitMQ and Order service is off

|  |  |  |
| --- | --- | --- |
| **Expected Result** | **Description** | **Step** |
| Create new Order entity with OrderStatusID =1 in DB  ordering service send a message to RabbitMQ-> queue:Basket -> Routing key: OrderStartedIntegrationEvent (Output Messages#1)  ordering service sends a message to RabbitMQ-> queue:Ordering.Signalrhub -> Routing key: OrderStatusChangedToSubmittedIntegrationEvent (Output Messages#2)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:   1. Update Order entity Set OrderStatusID =2 in DB 2. a message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) 3. same message has been sent to Ordering.Signalrhub queue. Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4)   message sent to RabbitMQ-> queue:Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#4) | Send message (Input Messages#1) into Basket Simulator :  RabbitMQ -> queue: Ordering -> Routing key : UserCheckoutAcceptedIntegrationEvent | 1 |
| Update Order entity Set OrderStatusID =3 in DB  Ordering sends a message to RabbitMQ-> queue:Payment -> Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent (Output Messages#3)  The same message has been sent to RabbitMQ-> queue: Signalrhub | Send message ( Input Messages#2) into Catalog Simulator :  RabbitMQ -> queue: Catalog-> Routing key : OrderStockConfirmedIntegrationEvent | 2 |
| Order service is not connected | Tern off Order service | 3 |
| Order service connected and the process is continued | After 5 second Tern on Order service | 4 |
| Update Order entity Set OrderStatusID =4 in DB | Send message (Input Messages#6 ) into Payment Simulator :  RabbitMQ -> queue: Payment -> Routing key :  OrderPaymentSucceededIntegrationEvent | 5 |

5. Message Dictionary

# Rabbit Messages

## Input Messages

|  |  |  |
| --- | --- | --- |
| # | Message Example | More Information |
| 1 | {  "UserId": "b9e5dcdd-dae2-4b1c-a991-f74aae042814",  "UserName": "alice",  "OrderNumber": 0,  "City": "Redmond",  "Street": "15703 NE 61st Ct",  "State": "WA",  "Country": "U.S.",  "ZipCode": "98052",  "CardNumber": "4012888888881881",  "CardHolderName": "Alice Smith",  "CardExpiration": "2024-12-31T22:00:00Z",  "CardSecurityNumber": "123",  "CardTypeId": 1,  "Buyer": null,  "RequestId": "**097d108a-18e6-4644-b160-acdd2ef87c14**",  "Basket": {  "BuyerId": "b9e5dcdd-dae2-4b1c-a991-f74aae042814",  "Items": [  {  "Id": "c1f98125-a109-4840-a751-c12a77f58dff",  "ProductId": 1,  "ProductName": ".NET Bot Black Hoodie",  "UnitPrice": 19.5,  "OldUnitPrice": 0,  "Quantity": 1,  "PictureUrl": "http://host.docker.internal:5202/c/api/v1/catalog/items/1/pic/"  }  ]  },  "Id": "16c5ddbc-229e-4c19-a4bd-d4148417529c",  "CreationDate": "2023-03-04T14:20:24.4730559Z" } | * Exchange: eshop\_event\_bus * Routing key: UserCheckoutAcceptedIntegrationEvent   Data:   * RequestId: Just Unique ID * Items: List of Items (see items schema in Schemas Table) |
|  | 1.1{  "UserId": "b9e5dcdd-dae2-4b1c-a991-f74aae042814",  "UserName": "alice",  "OrderNumber": 0,  "City": "Redmond",  "Street": "15703 NE 61st Ct",  "State": "WA",  "Country": "U.S.",  "ZipCode": "98052",  "CardNumber": "4012888888881881",  "CardHolderName": "Alice Smith",  "CardExpiration": "2024-12-31T22:00:00Z",  "CardSecurityNumber": "123",  "CardTypeId": 1,  "Buyer": null,  "RequestId": "**097d108a-18e6-4644-b160-acdd2ef87c14**",  "Basket": {  "BuyerId": "b9e5dcdd-dae2-4b1c-a991-f74aae042814",  "Items": [  {  "Id": "c1f98125-a109-4840-a751-c12a77f58dff",  "ProductId": 1,  "ProductName": ".NET Bot Black Hoodie",  "UnitPrice": 19.5,  "OldUnitPrice": 0,  "Quantity": 0,  "PictureUrl": "http://host.docker.internal:5202/c/api/v1/catalog/items/1/pic/"  }  ]  },  "Id": "16c5ddbc-229e-4c19-a4bd-d4148417529c",  "CreationDate": "2023-03-04T14:20:24.4730559Z" } | * Exchange: eshop\_event\_bus * Routing key: UserCheckoutAcceptedIntegrationEvent   Data:   * RequestId: Just Unique ID * Items: List of Items (see items schema in Schemas Table) |
|  | 1.2{  "UserId": "b9e5dcdd-dae2-4b1c-a991-f74aae042814",  "UserName": "alice",  "OrderNumber": 0,  "City": "Redmond",  "Street": "15703 NE 61st Ct",  "State": "WA",  "Country": "U.S.",  "ZipCode": "98052",  "CardNumber": "4012888888881881",  "CardHolderName": "Alice Smith",  "CardExpiration": "2024-12-31T22:00:00Z",  "CardSecurityNumber": "123",  "CardTypeId": 1,  "Buyer": null,  "RequestId": "**097d108a-18e6-4644-b160-acdd2ef87c14**",  "Basket": {  "BuyerId": "b9e5dcdd-dae2-4b1c-a991-f74aae042814",  "Items": [  { }  ]  },  "Id": "16c5ddbc-229e-4c19-a4bd-d4148417529c",  "CreationDate": "2023-03-04T14:20:24.4730559Z" } | * Exchange: eshop\_event\_bus * Routing key: UserCheckoutAcceptedIntegrationEvent   Data:   * RequestId: Just Unique ID * Items: List of Items (see items schema in Schemas Table) |
|  | 1.3 {  "UserId": "b9e5dcdd-dae2-4b1c-a991-f74aae042814",  "UserName": "**bob**",  "OrderNumber": 0,  "City": "Redmond",  "Street": "15703 NE 61st Ct",  "State": "WA",  "Country": "U.S.",  "ZipCode": "98052",  "CardNumber": "4012888888881881",  "CardHolderName": "Alice Smith",  "CardExpiration": "2024-12-31T22:00:00Z",  "CardSecurityNumber": "123",  "CardTypeId": 1,  "Buyer": null,  "RequestId": "**097d108a-18e6-4644-b160-acdd2ef87c14**",  "Basket": {  "BuyerId": "b9e5dcdd-dae2-4b1c-a991-f74aae042814",  "Items": [  {  "Id": "c1f98125-a109-4840-a751-c12a77f58dff",  "ProductId": 1,  "ProductName": ".NET Bot Black Hoodie",  "UnitPrice": 19.5,  "OldUnitPrice": 0,  "Quantity": 1,  "PictureUrl": "http://host.docker.internal:5202/c/api/v1/catalog/items/1/pic/"  }  ]  },  "Id": "16c5ddbc-229e-4c19-a4bd-d4148417529c",  "CreationDate": "2023-03-04T14:20:24.4730559Z" } | * Exchange: eshop\_event\_bus * Routing key: UserCheckoutAcceptedIntegrationEvent   Data:   * RequestId: Just Unique ID * Items: List of Items (see items schema in Schemas Table) |
| 2 | {  "OrderId": 51,  "Id": " 7157389f-3ab8-4d4b-a8b5-57e459caa6ff",  "CreationDate": "2023-03-05T14:52:24.705823Z" } | * Exchange: eshop\_event\_bus * Routing key: OrderStockConfirmedIntegrationEvent |
| 3 | {  "OrderId": 51,  "OrderStockItems": [  {  "ProductId": 1,  "HasStock": false  }  ],  "Id": "99c3f974-c6ed-41a4-8e01-5cb00f9e6335",  "CreationDate": "2023-03-05T15:51:11.5458796Z" } | * Exchange: eshop\_event\_bus * Routing key: OrderStockRejectedIntegrationEvent |
| 4 | {  "OrderId": 41,  "Id": "b84dc7a5-1d0e-429e-a800-d3024d9c724f",  "CreationDate": "2023-03-05T15:33:18.1376971Z" } | * Exchange: eshop\_event\_bus * Routing key: Routing key: OrderPaymentSucceededIntegrationEven |
| 5 | {  "OrderId": 31,  "OrderStatus": "awaitingvalidation",  "BuyerName": "alice",  "OrderStockItems": [  {  "ProductId": 1,  "Units": 1  }  ],  "Id": "1f07939c-8ad2-404b-963f-efe02cb6615a",  "CreationDate": "2023-03-05T14:27:28.8042812Z" } | * Exchange: eshop\_event\_bus * Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent   Data:   * id: Just Unique ID |
| 6 | {  "OrderId": 51,  "Id": "7ebe80ad-31be-4244-a501-bcc67290ec7e",  "CreationDate": "2023-03-05T16:07:11.8297341Z" } | * Exchange: eshop\_event\_bus * Routing key: OrderPaymentSucceededIntegrationEvent |
| 7 | {  "Ordered": 51,  "OrderStatus": "stockconfirmed",  "BuyerName": "alice",  "Id": "cca155c0-4480-4c93-a763-910e54218040",  "CreationDate": "2023-03-05T17:07:35.6306122Z" } | * Exchange: eshop\_event\_bus * Routing key: OrderPaymentFailedIntegrationEvent |

## Output Messages

|  |  |  |
| --- | --- | --- |
| # | Message Example | More Information |
| 1 | {  "UserId": "b9e5dcdd-dae2-4b1c-a991-f74aae042814",  "Id": "a2fbc6c0-07cf-4536-a1f5-fc9ede9fec07",  "CreationDate": "2023-03-05T13:43:13.8898923Z" } | * Exchange: eshop\_event\_bus * Routing key: Routing key: OrderStartedIntegrationEvent   Data:   * id: Just Unique ID |
| 2 | {  "OrderId": 31,  "OrderStatus": "submitted",  "BuyerName": "alice",  "Id": "cca359ec-15b0-4a9d-b5fd-44f896f49570",  "CreationDate": "2023-03-05T13:43:15.7845974Z" } | * Exchange: eshop\_event\_bus * Routing key: OrderStatusChangedToSubmittedIntegrationEvent   Data:   * id: Just Unique ID |
| 3 | {  "OrderId": 51,  "OrderStatus": "stockconfirmed",  "BuyerName": "alice",  "Id": "cca155c0-4480-4c93-a763-910e54218040",  "CreationDate": "2023-03-05T17:07:35.6306122Z" } | * Exchange: eshop\_event\_bus * Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent |
|  | 4 {  "OrderId": 317,  "OrderStatus": "paid",  "BuyerName": "alice",  "OrderStockItems": [  {  "ProductId": 1,  "Units": 1  }  ],  "Id": "383a647b-5886-4f3b-a75f-78838e8bf043",  "CreationDate": "2023-03-12T15:23:43.3070772Z" } | * Exchange: eshop\_event\_bus * Routing key:  |  | | --- | | OrderStatusChangedToPaidIntegrationEvent | |  | |

# API

## Schema Tables

|  |  |  |
| --- | --- | --- |
| # |  |  |
| 1 | Basket Item | {  Id:string nullable: true  productid:integer($int32)  productName:string  nullable: true  unitPrice:number($double)  oldUnitPrice:number($double)  quantity:integer($int32)  pictureUrl:string  nullable: true  } |
| 2 | Order | {  ordernumber:integer($int32)  date:string($date-time)  status:string  nullable:true  description:string  nullable:true  street:string  nullable: true  city:string  nullable:true  zipcode:string  nullable:true  country:string  nullable:true  orderitems[...]  totalnumber($double)  } |
|  |  |  |
|  |  |  |
|  |  |  |

# Database

|  |  |
| --- | --- |
| ID | Name |
| 1 | Submitted |
| 2 | Awaitingvalidation |
| 3 | Stockconfirmed |
| 4 | Paid |
| 5 | Shipped |
| 6 | Cancelled |

# 5.API Message

## 1.Ship Order

|  |  |  |
| --- | --- | --- |
|  | Method | Put |
| Request | URL | http://host.docker.internal:5102/api/v1/Orders/ship |
| Bearer Token |  |
| x-requestid | Unique Guide |
| Request Body | {  "orderNumber": 0  } |
| response | 200 | Success |
| 400 | Bad Request |
| 401 | Unauthorized |
| 403 | Forbidden |

## 2.Cancel Order

Order can be cancelled only with statuses (submitted, awaitingvalidation, stockconfirmed)

|  |  |  |
| --- | --- | --- |
|  | Method | Put |
| Request | URL | http://host.docker.internal:5102/api/v1/Orders/cancel |
| Bearer Token |  |
| x-requestid | Unique Guide |
| Request Body | {  "orderNumber": 0  } |
| response | 200 | Success |
| 400 | Bad Request |
| 401 | Unauthorized |
| 403 | Forbidden |

## 3.Get Order (online data)

|  |  |  |
| --- | --- | --- |
|  | Method | Get |
| Request | URL | http://host.docker.internal:5102/api/v1/Orders/{OrderId} |
| Bearer Token |  |
| x-requestid | Unique Guide |
| response | 200 | Success |
| See Order Schema |
| 400 | Bad Request |
| 401 | Unauthorized |
| 403 | Forbidden |