# Functional testing

## Test 1: Create Order success scenario, mss

Name: Ros Finkov

Requirement: Order Management

Preconditions: microservice Ordering, Identity running and connected to the required RabbitMQ queues, database is set up with the required tables and entities.

Basket, Catalog and Payment microservices are Simulated.

Ordering-Background service is down.

Test group: Sanity, Functional

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Send Basket simulator message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. A new order entity was created with OrderStatusID =1 in DB 2. in order remove items form basket, ordering service send a message to RabbitMQ-> queue: Basket -> Routing key: OrderStartedIntegrationEvent |
| 2 | Turn On Ordering-Backgroudtasks service. | 1. Order entity updated Set OrderStatusID =2 in DB 2. To verify that item in stock, a message sent to RabbitMQ-> queue: Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent |
| 3 | Send Catalog simulator message to RabbitMQ-> queue: Ordering -> Routing key: OrderStockConfirmedIntegrationEvent | 1. Order entity updated Set OrderStatusID = 3 in DB 2. In Order Create payment, Ordering sends a message to RabbitMQ-> queue: Payment -> Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent |
| 4 | Send Payment simulator message to RabbitMQ-> queue: Ordering-> Routing key: OrderPaymentSucceededIntegrationEvent | Order entity updated Set OrderStatusID =4 in DB and process is done here |

## Test 2: Create Order with failed payment

Name: Ros Finkov

Requirement: Order Management, Payment Processing

Preconditions: microservice Ordering, Identity running and connected to the required RabbitMQ queues, database is set up with the required tables and entities.

Test group: Functional

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket API service sends a message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. A new order entity was created with OrderStatusID =1 in DB 2. in order remove items form basket, ordering service send a message to RabbitMQ-> queue: Basket -> Routing key: OrderStartedIntegrationEvent |
| 2 | Turn up Ordering-Backgroudtasks to find the record with OrderStatusID =1 | 1. Order entity updated Set OrderStatusID =2 in DB 2. To verify that item in stock, a message sent to RabbitMQ-> queue: Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent |
| 3 | Catalog Sends a message to RabbitMQ-> queue: Ordering -> Routing key: OrderStockConfirmedIntegrationEvent | 1. Order entity updated Set OrderStatusID = 3 in DB 2. In Order Create payment, Ordering sends a message to RabbitMQ-> queue: Payment -> Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent |
| 4 | Payment service sends a message to RabbitMQ-> queue: Ordering-> Routing key: OrderPaymentFailedIntegrationEvent | Order entity updated Set OrderStatusID =6 in DB and process is done here |

## Test 3: Create Order with out-of-stock items

Name: Ros Finkov

Requirement: Order Management, Inventory Management

Preconditions: microservice Ordering, Identity running and connected to the required RabbitMQ queues, database is set up with the required tables and entities.

Test group: Functional

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket API service sends a message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. A new order entity was created with OrderStatusID =1 in DB 2. in order remove items form basket, ordering service send a message to RabbitMQ-> queue: Basket -> Routing key: OrderStartedIntegrationEvent |
| 2 | Turn up Ordering-Backgroudtasks to find the record with OrderStatusID =1 | 1. Order entity updated Set OrderStatusID =2 in DB 2. To verify that item in stock, a message sent to RabbitMQ-> queue: Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent |
| 3 | Catalog Sends a message to RabbitMQ-> queue: Ordering -> Routing key: OrderStockRejectedIntegrationEvent | Order entity updated Set OrderStatusID = 6 in DB and the process is done here |

## Test 4: Canceling an order from status Submitted

Name: Ros Finkov

Requirement: Order Management, Order Tracking

Preconditions: microservice Ordering, Identity running and connected to the required RabbitMQ queues, database is set up with the required tables and entities.

Test group: Functional

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket API service sends a message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. A new order entity was created with OrderStatusID =1 in DB 2. in order remove items form basket, ordering service send a message to RabbitMQ-> queue: Basket -> Routing key: OrderStartedIntegrationEvent |
| 2 | Cancel order through api | Order entity updated Set OrderStatusID =6 in DB and process is done here |

## Test 5: Canceling an order from status Awaitingvalidation

Name: Ros Finkov

Requirement: Order Management, Order Tracking

Preconditions: microservice Ordering, Identity running and connected to the required RabbitMQ queues, database is set up with the required tables and entities.

Test group: Functional

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket API service sends a message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. A new order entity was created with OrderStatusID =1 in DB 2. in order remove items form basket, ordering service send a message to RabbitMQ-> queue: Basket -> Routing key: OrderStartedIntegrationEvent |
| 2 | Turn up Ordering-Backgroudtasks to find the record with OrderStatusID =1 | 1. Order entity updated Set OrderStatusID =2 in DB 2. To verify that item in stock, a message sent to RabbitMQ-> queue: Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent |
| 3 | Cancel order through api | Order entity updated Set OrderStatusID =6 in DB and process is done here |

## Test 6: Canceling an order from status Stockconfirmed

Name: Ros Finkov

Requirement: Order Management, Order Tracking

Preconditions: microservice Ordering, Identity running and connected to the required RabbitMQ queues, database is set up with the required tables and entities.

Test group: Functional

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket API service sends a message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. A new order entity was created with OrderStatusID =1 in DB 2. in order remove items form basket, ordering service send a message to RabbitMQ-> queue: Basket -> Routing key: OrderStartedIntegrationEvent |
| 2 | Turn up Ordering-Backgroudtasks to find the record with OrderStatusID =1 | 1. Order entity updated Set OrderStatusID =2 in DB 2. To verify that item in stock, a message sent to RabbitMQ-> queue: Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent |
| 3 | Catalog Sends a message to RabbitMQ-> queue: Ordering -> Routing key: OrderStockConfirmedIntegrationEvent | 1. Order entity updated Set OrderStatusID = 3 in DB 2. In Order Create payment, Ordering sends a message to RabbitMQ-> queue: Payment -> Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent |
| 4 | Cancel order through api | Order entity updated Set OrderStatusID =6 in DB and process is done here |

## Test 7: Get all orders

Name: Ros Finkov

Requirement: Order Management, Order Tracking

Preconditions: microservice Ordering, Identity running and connected to the required RabbitMQ queues, database is set up with the required tables and entities.

Test group: Functional

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | User click get all orders | ordering API accepted the request and returned all orders with status code 200 |

## Test 8: Get an order by ID

Name: Ros Finkov

Requirement: Order Management, Order Tracking

Preconditions: microservice Ordering, Identity running and connected to the required RabbitMQ queues, database is set up with the required tables and entities.

Test group: Functional

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | User click get order by id | ordering API accepted the request and returned the order with status code 200 |

## Test 9: Update order from status Paid

Name: Ros Finkov

Requirement: Order Management, Order Tracking

Preconditions: microservice Ordering, Identity running and connected to the required RabbitMQ queues, database is set up with the required tables and entities.

Test group: Functional

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket API service sends a message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. A new order entity was created with OrderStatusID =1 in DB 2. in order remove items form basket, ordering service send a message to RabbitMQ-> queue: Basket -> Routing key: OrderStartedIntegrationEvent |
| 2 | Turn up Ordering-Backgroudtasks to find the record with OrderStatusID =1 | 1. Order entity updated Set OrderStatusID =2 in DB 2. To verify that item in stock, a message sent to RabbitMQ-> queue: Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent |
| 3 | Catalog Sends a message to RabbitMQ-> queue: Ordering -> Routing key: OrderStockConfirmedIntegrationEvent | 1. Order entity updated Set OrderStatusID = 3 in DB 2. In Order Create payment, Ordering sends a message to RabbitMQ-> queue: Payment -> Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent |
| 4 | Payment service sends a message to RabbitMQ-> queue: Ordering-> Routing key: OrderPaymentSucceededIntegrationEvent | Order entity updated Set OrderStatusID =4 in DB |
| 5 | User clicks ship | Order entity updated Set OrderStatusID =5 in DB and process is done here |

# Non-functional tests

## Test 10: Canceling an order from status Paid

Name: Ros Finkov

Requirement: Order Management

Preconditions: microservice Ordering, Identity running and connected to the required RabbitMQ queues, database is set up with the required tables and entities.

Test group: Non-functional

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket API service sends a message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. A new order entity was created with OrderStatusID =1 in DB 2. in order remove items form basket, ordering service send a message to RabbitMQ-> queue: Basket -> Routing key: OrderStartedIntegrationEvent |
| 2 | Turn up Ordering-Backgroudtasks to find the record with OrderStatusID =1 | 1. Order entity updated Set OrderStatusID =2 in DB 2. To verify that item in stock, a message sent to RabbitMQ-> queue: Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent |
| 3 | Catalog Sends a message to RabbitMQ-> queue: Ordering -> Routing key: OrderStockConfirmedIntegrationEvent | 1. Order entity updated Set OrderStatusID = 3 in DB 2. In Order Create payment, Ordering sends a message to RabbitMQ-> queue: Payment -> Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent |
| 4 | Payment service sends a message to RabbitMQ-> queue: Ordering-> Routing key: OrderPaymentSucceededIntegrationEvent | Order entity updated Set OrderStatusID =4 in DB |
| 5 | Cancel order through api | ordering API accepted the request and returned status code 400 |

## Test 11: Canceling an order from status Shipped

Name: Ros Finkov

Requirement: Order Management

Preconditions: microservice Ordering, Identity running and connected to the required RabbitMQ queues, database is set up with the required tables and entities.

Test group: Non-functional

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket API service sends a message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. A new order entity was created with OrderStatusID =1 in DB 2. in order remove items form basket, ordering service send a message to RabbitMQ-> queue: Basket -> Routing key: OrderStartedIntegrationEvent |
| 2 | Turn up Ordering-Backgroudtasks to find the record with OrderStatusID =1 | 1. Order entity updated Set OrderStatusID =2 in DB 2. To verify that item in stock, a message sent to RabbitMQ-> queue: Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent |
| 3 | Catalog Sends a message to RabbitMQ-> queue: Ordering -> Routing key: OrderStockConfirmedIntegrationEvent | 1. Order entity updated Set OrderStatusID = 3 in DB 2. In Order Create payment, Ordering sends a message to RabbitMQ-> queue: Payment -> Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent |
| 4 | Payment service sends a message to RabbitMQ-> queue: Ordering-> Routing key: OrderPaymentSucceededIntegrationEvent | Order entity updated Set OrderStatusID =4 in DB |
| 5 | Ship order through api | Order entity updated Set OrderStatusID =5 in DB |
| 6 | Cancel order through api | ordering API accepted the request and returned status code 400 |

## Test 12: Get order with a non-existent number

Name: Ros Finkov

Requirement: Order Management

Preconditions: microservice Ordering, Identity running and connected to the required RabbitMQ queues, database is set up with the required tables and entities.

Test group: Non-functional

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | User click get order by id with  non-existent id | ordering API accepted the request and returned status code 404 |

## Test 13: Update order from status Submitted

Name: Ros Finkov

Requirement: Order Management

Preconditions: microservice Ordering, Identity running and connected to the required RabbitMQ queues, database is set up with the required tables and entities.

Test group: Non-functional

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket API service sends a message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. A new order entity was created with OrderStatusID =1 in DB 2. in order remove items form basket, ordering service send a message to RabbitMQ-> queue: Basket -> Routing key: OrderStartedIntegrationEvent |
| 3 | Ship order through api | ordering API accepted the request and returned status code 400 |

## Test 14: Update an order from status Awaitingvalidation

Name: Ros Finkov

Requirement: Order Management

Preconditions: microservice Ordering, Identity running and connected to the required RabbitMQ queues, database is set up with the required tables and entities.

Test group: Non-functional

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket API service sends a message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. A new order entity was created with OrderStatusID =1 in DB 2. in order remove items form basket, ordering service send a message to RabbitMQ-> queue: Basket -> Routing key: OrderStartedIntegrationEvent |
| 2 | Turn up Ordering-Backgroudtasks to find the record with OrderStatusID =1 | 1. Order entity updated Set OrderStatusID =2 in DB 2. To verify that item in stock, a message sent to RabbitMQ-> queue: Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent |
| 3 | Ship order through api | ordering API accepted the request and returned status code 400 |

## Test 15: Update an order from status Stockconfirmed

Name: Ros Finkov

Requirement: Order Management

Preconditions: microservice Ordering, Identity running and connected to the required RabbitMQ queues, database is set up with the required tables and entities.

Test group: Non-functional

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket API service sends a message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. A new order entity was created with OrderStatusID =1 in DB 2. in order remove items form basket, ordering service send a message to RabbitMQ-> queue: Basket -> Routing key: OrderStartedIntegrationEvent |
| 2 | Turn up Ordering-Backgroudtasks to find the record with OrderStatusID =1 | 1. Order entity updated Set OrderStatusID =2 in DB 2. To verify that item in stock, a message sent to RabbitMQ-> queue: Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent |
| 3 | Catalog Sends a message to RabbitMQ-> queue: Ordering -> Routing key: OrderStockConfirmedIntegrationEvent | 1. Order entity updated Set OrderStatusID = 3 in DB 2. In Order Create payment, Ordering sends a message to RabbitMQ-> queue: Payment -> Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent |
| 4 | Ship order through api | ordering API accepted the request and returned status code 400 |

## Test 16: Update an order from status Cancelled

Name: Ros Finkov

Requirement: Order Management

Preconditions: microservice Ordering, Identity running and connected to the required RabbitMQ queues, database is set up with the required tables and entities.

Test group: Non-functional

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket API service sends a message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. A new order entity was created with OrderStatusID =1 in DB 2. in order remove items form basket, ordering service send a message to RabbitMQ-> queue: Basket -> Routing key: OrderStartedIntegrationEvent |
| 2 | Turn up Ordering-Backgroudtasks to find the record with OrderStatusID =1 | 1. Order entity updated Set OrderStatusID =2 in DB 2. To verify that item in stock, a message sent to RabbitMQ-> queue: Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent |
| 3 | Catalog Sends a message to RabbitMQ-> queue: Ordering -> Routing key: OrderStockConfirmedIntegrationEvent | 1. Order entity updated Set OrderStatusID = 3 in DB 2. In Order Create payment, Ordering sends a message to RabbitMQ-> queue: Payment -> Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent |
| 4 | Cancel order through api | Order entity updated Set OrderStatusID =6 |
| 5 | Ship order through api | ordering API accepted the request and returned status code 400 |

# Performance testing

## Test 17: Crash the server after any of the steps in mss

Name: Ros Finkov

Requirement: Order Management, Reliability

Preconditions: microservice Ordering, Identity running and connected to the required RabbitMQ queues, database is set up with the required tables and entities.

Test group: Performance

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket API service sends a message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. A new order entity was created with OrderStatusID =1 in DB 2. in order remove items form basket, ordering service send a message to RabbitMQ-> queue: Basket -> Routing key: OrderStartedIntegrationEvent |
| 2 | Crash and restart Ordering service | Order entity was updated to OrderStatusID = 1 in the DB |
| 3 | Turn up Ordering-Backgroudtasks to find the record with OrderStatusID =1 | 1. Order entity updated Set OrderStatusID =2 in DB 2. To verify that item in stock, a message sent to RabbitMQ-> queue: Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent |
| 4 | Crash and restart Ordering-Backgroudtasks service | Order entity was updated to OrderStatusID = 2 in the DB |
| 5 | Catalog Sends a message to RabbitMQ-> queue: Ordering -> Routing key: OrderStockConfirmedIntegrationEvent | 1. Order entity updated Set OrderStatusID = 3 in DB 2. In Order Create payment, Ordering sends a message to RabbitMQ-> queue: Payment -> Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent |
| 6 | Crash and restart Ordering service | Order entity was updated to OrderStatusID = 3 in the DB |
| 7 | Payment service send a message to RabbitMQ-> queue: Ordering-> Routing key: OrderPaymentSucceededIntegrationEvent | Order entity updated Set OrderStatusID =4 in DB |
| 8 | Crash and restart Ordering service | Order entity was updated to OrderStatusID = 4 in the DB |

## Test 18: Crashed the server after step 1 in mss

Name: Ros Finkov

Requirement: Order Management, Reliability

Preconditions: microservice Ordering, Identity running and connected to the required RabbitMQ queues, database is set up with the required tables and entities.

Test group: Performance

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket API service sends a message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. A new order entity was created with OrderStatusID =1 in DB 2. in order remove items form basket, ordering service send a message to RabbitMQ-> queue: Basket -> Routing key: OrderStartedIntegrationEvent |
| 2 | Crash and restart Ordering service | Order entity was updated to OrderStatusID = 1 in the DB |
| 3 | Turn up Ordering-Backgroudtasks to find the record with OrderStatusID =1 | 1. Order entity updated Set OrderStatusID =2 in DB 2. To verify that item in stock, a message sent to RabbitMQ-> queue: Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent |
| 4 | Catalog Sends a message to RabbitMQ-> queue: Ordering -> Routing key: OrderStockConfirmedIntegrationEvent | 1. Order entity updated Set OrderStatusID = 3 in DB 2. In Order Create payment, Ordering sends a message to RabbitMQ-> queue: Payment -> Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent |
| 5 | Payment service send a message to RabbitMQ-> queue: Ordering-> Routing key: OrderPaymentSucceededIntegrationEvent | Order entity updated Set OrderStatusID =4 in DB |

## Test 19: Crashed the server after step 2 in mss

Name: Ros Finkov

Requirement: Order Management, Reliability

Preconditions: microservice Ordering, Identity running and connected to the required RabbitMQ queues, database is set up with the required tables and entities.

Test group: Performance

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket API service sends a message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. A new order entity was created with OrderStatusID =1 in DB 2. in order remove items form basket, ordering service send a message to RabbitMQ-> queue: Basket -> Routing key: OrderStartedIntegrationEvent |
| 2 | Turn up Ordering-Backgroudtasks to find the record with OrderStatusID =1 | 1. Order entity updated Set OrderStatusID =2 in DB 2. To verify that item in stock, a message sent to RabbitMQ-> queue: Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent |
| 3 | Crash and restart Ordering service | Order entity was updated to OrderStatusID = 2 in the DB |
| 4 | Catalog Sends a message to RabbitMQ-> queue: Ordering -> Routing key: OrderStockConfirmedIntegrationEvent | 1. Order entity updated Set OrderStatusID = 3 in DB 2. In Order Create payment, Ordering sends a message to RabbitMQ-> queue: Payment -> Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent |
| 5 | Payment service send a message to RabbitMQ-> queue: Ordering-> Routing key: OrderPaymentSucceededIntegrationEvent | Order entity updated Set OrderStatusID =4 in DB |

## Test 20: Crashed the server after step 3 in mss

Name: Ros Finkov

Requirement: Order Management, Reliability

Preconditions: microservice Ordering, Identity running and connected to the required RabbitMQ queues, database is set up with the required tables and entities.

Test group: Performance

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket API service sends a message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. A new order entity was created with OrderStatusID =1 in DB 2. in order remove items form basket, ordering service send a message to RabbitMQ-> queue: Basket -> Routing key: OrderStartedIntegrationEvent |
| 2 | Turn up Ordering-Backgroudtasks to find the record with OrderStatusID =1 | 1. Order entity updated Set OrderStatusID =2 in DB 2. To verify that item in stock, a message sent to RabbitMQ-> queue: Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent |
| 3 | Catalog Sends a message to RabbitMQ-> queue: Ordering -> Routing key: OrderStockConfirmedIntegrationEvent | 1. Order entity updated Set OrderStatusID = 3 in DB 2. In Order Create payment, Ordering sends a message to RabbitMQ-> queue: Payment -> Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent |
| 4 | Crash and restart Ordering service | Order entity was updated to OrderStatusID = 3 in the DB |
| 5 | Payment service send a message to RabbitMQ-> queue: Ordering-> Routing key: OrderPaymentSucceededIntegrationEvent | Order entity updated Set OrderStatusID =4 in DB |

## Test 21: Crashed the server after step 4 in mss

Name: Ros Finkov

Requirement: Order Management, Reliability

Preconditions: microservice Ordering, Identity running and connected to the required RabbitMQ queues, database is set up with the required tables and entities.

Test group: Performance

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket API service sends a message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. A new order entity was created with OrderStatusID =1 in DB 2. in order remove items form basket, ordering service send a message to RabbitMQ-> queue: Basket -> Routing key: OrderStartedIntegrationEvent |
| 2 | Turn up Ordering-Backgroudtasks to find the record with OrderStatusID =1 | 1. Order entity updated Set OrderStatusID =2 in DB 2. To verify that item in stock, a message sent to RabbitMQ-> queue: Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent |
| 3 | Catalog Sends a message to RabbitMQ-> queue: Ordering -> Routing key: OrderStockConfirmedIntegrationEvent | 1. Order entity updated Set OrderStatusID = 3 in DB 2. In Order Create payment, Ordering sends a message to RabbitMQ-> queue: Payment -> Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent |
| 4 | Payment service send a message to RabbitMQ-> queue: Ordering-> Routing key: OrderPaymentSucceededIntegrationEvent | Order entity updated Set OrderStatusID =4 in DB |
| 5 | Crash and restart Ordering service | Order entity was updated to OrderStatusID = 4 in the DB |

# Load Tests

## Test 22: Check loads on creating an order

Name: Ros Finkov

Requirement: Order Management, Scalability

Preconditions: microservice Ordering, Identity running and connected to the required RabbitMQ queues, database is set up with the required tables and entities.

Test group: Load

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket API service sends a message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. A new order entity was created with OrderStatusID =1 in DB 2. in order remove items form basket, ordering service send a message to RabbitMQ-> queue: Basket -> Routing key: OrderStartedIntegrationEvent |
| 2 | Turn up Ordering-Backgroudtasks to find the record with OrderStatusID =1 | 1. Order entity updated Set OrderStatusID =2 in DB 2. To verify that item in stock, a message sent to RabbitMQ-> queue: Catalog -> Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent |
| 3 | Catalog Sends a message to RabbitMQ-> queue: Ordering -> Routing key: OrderStockConfirmedIntegrationEvent | 1. Order entity updated Set OrderStatusID = 3 in DB 2. In Order Create payment, Ordering sends a message to RabbitMQ-> queue: Payment -> Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent |
| 4 | Payment service send a message to RabbitMQ-> queue: Ordering-> Routing key: OrderPaymentSucceededIntegrationEvent | Order entity updated Set OrderStatusID =4 in DB |
| 5 | Repeat steps 1-4 | 1. The microservice should be able to handle at least 100 orders within one hour without any errors or delays. 2. All the order processing steps should be completed successfully for each incoming order message. 3. The Order entity status should be updated correctly in the database after each order processing step.   All the messages sent to the RabbitMQ queues with the corresponding routing keys should be received and processed successfully by the microservice. |

# Security Tests

## Test 23: cancel order from another user

Name: Ros Finkov

Requirement: Security

Preconditions: microservice Ordering, Identity running and connected to the required RabbitMQ queues, database is set up with the required tables and entities.

Test group: Security

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | User click Cancel Order by ID with ID from another user | ordering API accepted the request and returned status code 400 |

## Test 24: ship order from another user

Name: Ros Finkov

Requirement: Security

Preconditions: microservice Ordering, Identity running and connected to the required RabbitMQ queues, database is set up with the required tables and entities.

Test group: Security

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | User click Ship Order by ID with ID from another user | ordering API accepted the request and returned status code 400 |