Test Cases

**Test** **number**: 1

**Test** **name**: Create Order (MSS)

**Traceability**: Order Management

**Description**: Test the creation of a new order

**Preconditions**: Ordering is running, Identity is running, database is running, Ordering-Background is running, Basket and Catalog and Payment are Simulated.

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket simulator send message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. New order is created with “OrderStatusId” 1 in DB 2. Basket simulator receives message Routing key: OrderStartedIntegrationEvent 3. ordering-backgroundtasks update “OrderStatusId” to 2 4. Catalog simulator receives message Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent |
| 3 | Catalog simulator send message to RabbitMQ-> queue: Ordering -> Routing key: OrderStockConfirmedIntegrationEvent | 1. update “OrderStatusId” to 3 2. Payment simulator receives message Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent |
| 4 | Payment simulator send message to RabbitMQ-> queue: Ordering-> Routing key: OrderPaymentSucceededIntegrationEvent | 1. update “OrderStatusId” to 4 2. Catalog simulator receives message Routing key: OrderStatusChangedToPaidIntegrationEvent |

**Test** **number**: 2

**Test** **name**: Get all orders

**Traceability**: Order Management

**Description**: Test checks that the function returns a 200 status code

**Preconditions**: Ordering is running, Identity is running, database is running, Ordering-Background is running, Basket and Catalog and Payment are Simulated.

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | message get\_orders through api | Status code = 200 |

**Test** **number**: 3

**Test** **name**: Get order by id

**Traceability**: Order Management

**Description**: Test checks that the function returns a 200 status code

**Preconditions**: Ordering is running, Identity is running, database is running, Ordering-Background is running, Basket and Catalog and Payment are Simulated.

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | message get\_order by id through api | Status code = 200 |

**Test** **number**: 4

**Test** **name**: get card type

**Traceability**: Order Management

**Description**: Test checks that the function returns a 200 status code

**Preconditions**: Ordering is running, Identity is running, database is running, Ordering-Background is running, Basket and Catalog and Payment are Simulated.

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | message get\_card\_type through api | Status code = 200 |

**Test number**: 5

**Test name**: update status 1 to 5(shipped)

**Traceability**: Order Management

**Description**: Test checks that it is impossible to change from status 1 to 5

**Preconditions**: Ordering is running, Identity is running, database is running, Ordering-Background is running, Basket and Catalog and Payment are Simulated.

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket simulator send message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. New order is created with “OrderStatusId” 1 in DB 2. Basket simulator receives message Routing key: OrderStartedIntegrationEvent |
| 2 | Turn off the ordering.backgroundtasks container |  |
| 3 | message ship through api | Status code 400 and order status will remain 1 |

**Test** **number**: 6

**Test** **name**: update status 2 to 5(shipped)

**Traceability**: Order Management

**Description**: Test checks that it is impossible to change from status 2 to 5

**Preconditions**: Ordering is running, Identity is running, database is running, Ordering-Background is running, Basket and Catalog and Payment are Simulated.

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket simulator send message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. New order is created with “OrderStatusId” 1 in DB 2. Basket simulator receives message Routing key: OrderStartedIntegrationEvent 3. ordering-backgroundtasks update “OrderStatusId” to 2 4. Catalog simulator receives message Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent |
| 2 | message ship through api | Status code 400 and order status will remain 2 |

**Test** **number**: 7

**Test** **name**: update status 3 to 5(shipped)

**Traceability**: Order Management

**Description**: Test checks that it is impossible to change from status 3 to 5

**Preconditions**: Ordering is running, Identity is running, database is running, Ordering-Background is running, Basket and Catalog and Payment are Simulated.

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket simulator send message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. New order is created with “OrderStatusId” 1 in DB 2. Basket simulator receives message Routing key: OrderStartedIntegrationEvent 3. ordering-backgroundtasks update “OrderStatusId” to 2 4. Catalog simulator receives message Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent |
| 2 | Catalog simulator send message to RabbitMQ-> queue:Ordering -> Routing key: OrderStockConfirmedIntegrationEvent | 1. update “OrderStatusId” 3 in DB 2. Payment simulator receives message Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent |
| 3 | message ship through api | Status code 400 and order status will remain 3 |

**Test** **number**: 8

**Test** **name**: update status 4 to 5(shipped)

**Traceability**: Order Management

**Description**: Test checks that it is possible to change from status 4 to 5

**Preconditions**: Ordering is running, Identity is running, database is running, Ordering-Background is running, Basket and Catalog and Payment are Simulated.

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket simulator send message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. New order is created with “OrderStatusId” 1 in DB 2. Basket simulator receives message Routing key: OrderStartedIntegrationEvent 3. ordering-backgroundtasks update “OrderStatusId” to 2 4. Catalog simulator receives message Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent |
| 2 | Catalog simulator send message to RabbitMQ-> queue:Ordering -> Routing key: OrderStockConfirmedIntegrationEvent | 1. update “OrderStatusId” 3 in DB 2. Payment simulator receives message Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent |
| 3 | Payment simulator send message to RabbitMQ-> queue: Ordering-> Routing key: OrderPaymentSucceededIntegrationEvent | 1. update “OrderStatusId” 4 in DB 2. Catalog simulator receives message Routing key: OrderStatusChangedToPaidIntegrationEvent |
| 4 | message ship through api | Status code 200 and order status update to 5 |

**Test** **number**: 9

**Test** **name**: update status 6 to 5(shipped)

**Traceability**: Order Management

**Description**: Test checks that it is impossible to change from status 6 to 5

**Preconditions**: Ordering is running, Identity is running, database is running, Ordering-Background is running, Basket and Catalog and Payment are Simulated.

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket simulator send message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. New order is created with “OrderStatusId” 1 in DB 2. Basket simulator receives message Routing key: OrderStartedIntegrationEvent |
| 2 | message cancel through api | Order status 6 |
| 3 | message ship through api | Status code 400 and order status will remain 6 |

**Test** **number**: 10

**Test** **name**: cancel status 1

**Traceability**: Order Management

**Description**: Test checks that an order with status 1 can be canceled

**Preconditions**: Ordering is running, Identity is running, database is running, Ordering-Background is running, Basket and Catalog and Payment are Simulated.

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket simulator send message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. New order is created with “OrderStatusId” 1 in DB 2. Basket simulator receives message Routing key: OrderStartedIntegrationEvent |
| 2 | Turn off the ordering.backgroundtasks container |  |
| 3 | message cancel through api | Order status 6 |

**Test** **number**: 11

**Test** **name**: cancel status 2

**Traceability**: Order Management

**Description**: Test checks that an order with status 2 can be canceled

**Preconditions**: Ordering is running, Identity is running, database is running, Ordering-Background is running, Basket and Catalog and Payment are Simulated.

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket simulator send message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. New order is created with “OrderStatusId” 1 in DB 2. Basket simulator receives message Routing key: OrderStartedIntegrationEvent 3. ordering-backgroundtasks update “OrderStatusId” to 2 4. Catalog simulator receives message Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent |
| 2 | message cancel through api | Order status 6 |

**Test** **number**: 12

**Test** **name**: cancel status 3

**Traceability**: Order Management

**Description**: Test checks that an order with status 3 can be canceled

**Preconditions**: Ordering is running, Identity is running, database is running, Ordering-Background is running, Basket and Catalog and Payment are Simulated.

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket simulator send message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. New order is created with “OrderStatusId” 1 in DB 2. Basket simulator receives message Routing key: OrderStartedIntegrationEvent 3. ordering-backgroundtasks update “OrderStatusId” to 2 4. Catalog simulator receives message Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent |
| 2 | Catalog simulator send message to RabbitMQ-> queue:Ordering -> Routing key: OrderStockConfirmedIntegrationEvent | 1. update “OrderStatusId” 3 in DB 2. Payment simulator receives message Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent |
| 3 | message cancel through api | Order status 6 |

**Test** **number**: 13

**Test** **name**: cancel status 4

**Traceability**: Order Management

**Description**: Test checks that it is impossible to cancel an order with status 4

**Preconditions**: Ordering is running, Identity is running, database is running, Ordering-Background is running, Basket and Catalog and Payment are Simulated.

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket simulator send message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. New order is created with “OrderStatusId” 1 in DB 2. Basket simulator receives message Routing key: OrderStartedIntegrationEvent 3. ordering-backgroundtasks update “OrderStatusId” to 2 4. Catalog simulator receives message Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent |
| 2 | Catalog simulator send message to RabbitMQ-> queue:Ordering -> Routing key: OrderStockConfirmedIntegrationEvent | 1. update “OrderStatusId” 3 in DB 2. Payment simulator receives message Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent |
| 3 | Payment simulator send message to RabbitMQ-> queue: Ordering-> Routing key: OrderPaymentSucceededIntegrationEvent | 1. update “OrderStatusId” 4 in DB 2. Catalog simulator receives message Routing key: OrderStatusChangedToPaidIntegrationEvent |
| 4 | message cancel through api | Order status 6 |

**Test** **number**: 14

**Test** **name**: cancel status 5

**Traceability**: Order Management

**Description**: Test checks that it is impossible to cancel an order with status 5

**Preconditions**: Ordering is running, Identity is running, database is running, Ordering-Background is running, Basket and Catalog and Payment are Simulated.

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket simulator send message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. New order is created with “OrderStatusId” 1 in DB 2. Basket simulator receives message Routing key: OrderStartedIntegrationEvent 3. ordering-backgroundtasks update “OrderStatusId” to 2 4. Catalog simulator receives message Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent |
| 2 | Catalog simulator send message to RabbitMQ-> queue:Ordering -> Routing key: OrderStockConfirmedIntegrationEvent | 1. update “OrderStatusId” 3 in DB 2. Payment simulator receives message Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent |
| 3 | Payment simulator send message to RabbitMQ-> queue: Ordering-> Routing key: OrderPaymentSucceededIntegrationEvent | 1. update “OrderStatusId” 4 in DB 2. Catalog simulator receives message Routing key: OrderStatusChangedToPaidIntegrationEvent |
| 4 | message ship through api | status update to 5 |
| 5 | message cancel through api | Order status 6 |

**Test** **number**: 15

**Test** **name**: order tracking

**Traceability**: order tracking

**Description**: test checks that the statuses change according to the step where the user is at

**Preconditions**: Ordering is running, Identity is running, database is running, Ordering-Background is running, Basket and Catalog and Payment are Simulated.

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket simulator send message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. New order is created with “OrderStatusId” 1 in DB 2. Basket simulator receives message Routing key: OrderStartedIntegrationEvent |
|  | message get\_order\_id(last\_order\_id())['status'] through api | status = submitted |
|  | ordering-backgroundtasks update “OrderStatusId” to 2 | Catalog simulator receives message Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent |
|  | message get\_order\_id(last\_order\_id())['status'] through api | status = awaitingvalidation |
| 2 | Catalog simulator send message to RabbitMQ-> queue:Ordering -> Routing key: OrderStockConfirmedIntegrationEvent | 1. update “OrderStatusId” 3 in DB 2. Payment simulator receives message Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent |
|  | message get\_order\_id(last\_order\_id())['status'] through api | status = stockconfirmed |
| 3 | Payment simulator send message to RabbitMQ-> queue: Ordering-> Routing key: OrderPaymentSucceededIntegrationEvent | 1. update “OrderStatusId” 4 in DB 2. Catalog simulator receives message Routing key: OrderStatusChangedToPaidIntegrationEvent |
| 4 | message get\_order\_id(last\_order\_id())['status'] through api | status = paid |

**Test** **number**: 16

**Test** **name**: payment succeeded

**Traceability**: Payment Processing

**Description**: Test checks that the payment is correct

**Preconditions**: Ordering is running, Identity is running, database is running, Ordering-Background is running, Basket and Catalog and Payment are Simulated.

|  |  |  |
| --- | --- | --- |
| ב | Test steps | Expected result |
| 1 | Basket simulator send message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. New order is created with “OrderStatusId” 1 in DB 2. Basket simulator receives message Routing key: OrderStartedIntegrationEvent 3. ordering-backgroundtasks update “OrderStatusId” to 2 4. Catalog simulator receives message Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent |
| 2 | Catalog simulator send message to RabbitMQ-> queue:Ordering -> Routing key: OrderStockConfirmedIntegrationEvent | 1. update “OrderStatusId” 3 in DB 2. Payment simulator receives message Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent |
| 3 | Payment simulator send message to RabbitMQ-> queue: Ordering-> Routing key: OrderPaymentSucceededIntegrationEvent | 1. update “OrderStatusId” 4 in DB 2. Catalog simulator receives message Routing key: OrderStatusChangedToPaidIntegrationEvent |

**Test** **number**: 17

**Test** **name**: payment failed

**Traceability**: Payment Processing

**Description**: Test checks for a failed payment

**Preconditions**: Ordering is running, Identity is running, database is running, Ordering-Background is running, Basket and Catalog and Payment are Simulated.

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket simulator send message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. New order is created with “OrderStatusId” 1 in DB 2. Basket simulator receives message Routing key: OrderStartedIntegrationEvent 3. ordering-backgroundtasks update “OrderStatusId” to 2 4. Catalog simulator receives message Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent |
| 2 | Catalog simulator send message to RabbitMQ-> queue:Ordering -> Routing key: OrderStockConfirmedIntegrationEvent | 1. update “OrderStatusId” 3 in DB 2. Payment simulator receives message Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent |
| 3 | payment simulator send message to RabbitMQ-> queue: Ordering-> Routing key: OrderPaymentFailedIntegrationEvent | 1. update “OrderStatusId” 6 in DB 2. Ordering.signalrhub simulator receives message Routing key: OrderStatusChangedToCancelledIntegrationEvent |

**Test** **number**: 18

**Test** **name**: confirm stock

**Traceability**: Inventory Management

**Description**: Test checks an order with proper stock

**Preconditions**: Ordering is running, Identity is running, database is running, Ordering-Background is running, Basket and Catalog and Payment are Simulated.

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket simulator send message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. New order is created with “OrderStatusId” 1 in DB 2. Basket simulator receives message Routing key: OrderStartedIntegrationEvent 3. ordering-backgroundtasks update “OrderStatusId” to 2 4. Catalog simulator receives message Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent |
| 2 | Catalog simulator send message to RabbitMQ-> queue:Ordering -> Routing key: OrderStockConfirmedIntegrationEvent | 1. update “OrderStatusId” 3 in DB 2. Payment simulator receives message Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent |

**Test** **number**: 19

**Test** **name**: reject stock

**Traceability**: Inventory Management

**Description**: Test checks an order with incorrect inventory

**Preconditions**: Ordering is running, Identity is running, database is running, Ordering-Background is running, Basket and Catalog and Payment are Simulated.

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket simulator send message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. New order is created with “OrderStatusId” 1 in DB 2. Basket simulator receives message Routing key: OrderStartedIntegrationEvent 3. ordering-backgroundtasks update “OrderStatusId” to 2 4. Catalog simulator receives message Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent |
| 2 | message cancel through api | Order status 6 |

**Test** **number**: 20

**Test** **name**: cancel another user

**Traceability**: Security

**Description**: Test that checks that it is not possible to cancel an order of another user.

**Preconditions**: Ordering is running, Identity is running, database is running, Ordering-Background is running, Basket and Catalog and Payment are Simulated.

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket simulator send message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent  **Creating an order with another user** | 1. New order is created with “OrderStatusId” 1 in DB 2. Basket simulator receives message Routing key: OrderStartedIntegrationEvent |
| 3 | message cancel through api | Status code = 401 |

**Test** **number**: 21

**Test** **name**: payment another user

**Traceability**: Security

**Description**: Test that verifies that it's impossible to pay for an order of another user.

**Preconditions**: Ordering is running, Identity is running, database is running, Ordering-Background is running, Basket and Catalog and Payment are Simulated.

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket simulator send message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent  **Creating an order with another user** | 1. New order is created with “OrderStatusId” 1 in DB 2. Basket simulator receives message Routing key: OrderStartedIntegrationEvent 3. ordering-backgroundtasks update “OrderStatusId” to 2 4. Catalog simulator receives message Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent |
| 2 | Catalog simulator send message to RabbitMQ-> queue:Ordering -> Routing key: OrderStockConfirmedIntegrationEvent | 1. update “OrderStatusId” 3 in DB 2. Payment simulator receives message Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent |
| 3 | Payment simulator send message to RabbitMQ-> queue: Ordering-> Routing key: OrderPaymentSucceededIntegrationEvent | Status order != 4 |

**Test** **number**: 22

**Test** **name**: ship another user

**Traceability**: Security

**Description**: Test that checks that it is not possible to send a shipment for another users order.

**Preconditions**: Ordering is running, Identity is running, database is running, Ordering-Background is running, Basket and Catalog and Payment are Simulated.

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket simulator send message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. New order is created with “OrderStatusId” 1 in DB 2. Basket simulator receives message Routing key: OrderStartedIntegrationEvent 3. ordering-backgroundtasks update “OrderStatusId” to 2 4. Catalog simulator receives message Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent |
| 2 | Catalog simulator send message to RabbitMQ-> queue:Ordering -> Routing key: OrderStockConfirmedIntegrationEvent | 1. update “OrderStatusId” 3 in DB 2. Payment simulator receives message Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent |
| 3 | Payment simulator send message to RabbitMQ-> queue: Ordering-> Routing key: OrderPaymentSucceededIntegrationEvent | 1. update “OrderStatusId” 4 in DB 2. Catalog simulator receives message Routing key: OrderStatusChangedToPaidIntegrationEvent |
| 4 | message ship through api | Status code 401 |

**Test** **number**: 23

**Test** **name**: hundred orders per hour

**Traceability**: Scalability

**Description**: Test checks whether the system is capable of closing 100 orders within an hour.

**Preconditions**: Ordering is running, Identity is running, database is running, Ordering-Background is running, Basket and Catalog and Payment are Simulated

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Create a loop of 100 repetitions | One hundred orders |
| 2 | Basket simulator send message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. New order is created with “OrderStatusId” 1 in DB 2. Basket simulator receives message Routing key: OrderStartedIntegrationEvent 3. ordering-backgroundtasks update “OrderStatusId” to 2 4. Catalog simulator receives message Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent |
| 3 | Catalog simulator send message to RabbitMQ-> queue: Ordering -> Routing key: OrderStockConfirmedIntegrationEvent | 1. update “OrderStatusId” to 3 2. Payment simulator receives message Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent |
| 4 | Payment simulator send message to RabbitMQ-> queue: Ordering-> Routing key: OrderPaymentSucceededIntegrationEvent | 1. update “OrderStatusId” to 4 2. Catalog simulator receives message Routing key: OrderStatusChangedToPaidIntegrationEvent |
| 5 | Start time minus end time and compare to an hour | The result should be less than an hour |

**Test** **number**: 24

**Test** **name**: crash ordering service

**Traceability**: Reliability

**Description**: Test that checks if the system returns to the same place it stopped after a crash.

**Preconditions**: Ordering is running, Identity is running, database is running, Ordering-Background is running, Basket and Catalog and Payment are Simulated

|  |  |  |
| --- | --- | --- |
|  | Test steps | Expected result |
| 1 | Basket simulator send message to RabbitMQ-> queue: Ordering -> Routing key: UserCheckoutAcceptedIntegrationEvent | 1. New order is created with “OrderStatusId” 1 in DB 2. Basket simulator receives message Routing key: OrderStartedIntegrationEvent 3. ordering-backgroundtasks update “OrderStatusId” to 2 4. Catalog simulator receives message Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent |
|  | Turn off the ordering api container  And then Turn on the ordering api container | The container turns off and after that turns on again |
| 3 | Catalog simulator send message to RabbitMQ-> queue: Ordering -> Routing key: OrderStockConfirmedIntegrationEvent | 1. update “OrderStatusId” to 3 2. Payment simulator receives message Routing key: OrderStatusChangedToStockConfirmedIntegrationEvent |
|  | Turn off the ordering api container  And then Turn on the ordering api container | The container turns off and after that turns on again |
| 4 | Payment simulator send message to RabbitMQ-> queue: Ordering-> Routing key: OrderPaymentSucceededIntegrationEvent | 1. update “OrderStatusId” to 4 2. Catalog simulator receives message Routing key: OrderStatusChangedToPaidIntegrationEvent |