Order Management API – Business Logic

Indhold

[Customers 2](#_Toc77088420)

[Create Customer 2](#_Toc77088421)

[Get Customer(s) 3](#_Toc77088422)

[Patch Customer 3](#_Toc77088423)

[Update Customer 3](#_Toc77088424)

[Deactivate Customer 3](#_Toc77088425)

[Delete Customer 4](#_Toc77088426)

[Orders 5](#_Toc77088427)

[Products 6](#_Toc77088428)

# Customers

* E-mail must be unique.
  + If the E-mail exists in the database, then the user is not created.
  + The response should give a generic error message for security reasons.
* Phone number must be unique
  + If the Phone number exists in the database, then the user is not created.
  + The response should give a generic error message for security reasons.
  + Only supports phone numbers in Denmark and should therefore be 8 digits.
* Required fields
  + FirstName, LastName, Address, Mail
* Customer Status
  + NoAccount
    - The user explicitly said they do NOT wish to create an account.
    - The user is still created, but with fillers on required fields.
  + Customer
    - The user signed up, but did not place an order yet
  + PurchasedOnce
    - The user is signed up and purchased an order once.
    - Eligible for discount code on next order
  + PurchasedMultiple
    - The user is signed up and purchased multiple orders.
    - Eligible for special discounts based purchased products.
  + Deactivated
    - The user has deactivated their account.
    - Not able to sign in as a user but is still registered in the system.

## Create Customer

When creating a customer in the system, the steps below shows the process. Each step should be unit tested.

1. The Customer object is received and validated
   1. First Name and Last Name is required and should be at least 2 characters long
      1. If not, return 400 BadRequest(customer)
   2. Address is required
      1. If not, return 400 BadRequest(customer)
   3. E-Mail is required and should be unique and validated as an email ([customer@domain.com](mailto:customer@domain.com) format)
      1. If not, return 400 BadRequest(customer)
2. The Customer object is inserted into the Database
   1. Once inserted into the database, the customer id is to be returned
      1. If the id is 0 or less, then the customer insertion failed
      2. If the id is higher than 0, then the customer was inserted successfully
3. The Customer object is returned with the id generated in the Database
   1. Return 201 Created(uri, customer)

## Get Customer(s)

Retrieving customers from the database can be done in multiple scenarios, depending on the user. It can be done so by getting all records, getting a single customer by id, or getting a list of customers based on their status. The steps are the same:

1. Retrieve records from database
   1. Check if there is parameter
2. Return the retrieved customer

## Patch Customer

The customer can change e-mail, address, name, or status which only requires one field to be updated. This should be done as a patch request and only handle the necessary information. The steps are the same:

1. Validate the information
   1. The e-mail should follow e-mail format
   2. The address needs to contain name of the street followed by a number
   3. The name should at least be 2 characters
   4. The id should be above 0
2. Check if the Customer exists
   1. Find customer by the given id
   2. If the customer was not found, return 404 NotFound(customer id)
3. Update the given information
4. Return Customer with new information

## Update Customer

The customer or admin can update multiple details. In this case the put request should be used to handle the entire customer object and return the updated customer. The steps are as follows:

1. Authenticate Customer or Admin
   1. If not the actual customer or an admin, return 401 Unauthorized
2. Validate customer details
   1. The e-mail should follow e-mail format
   2. The address needs to contain name of the street followed by a number
   3. The name should at least be 2 characters
   4. The id should be above 0
   5. Required fields should not be empty or whitespace
   6. Optional fields can be empty or whitespace
3. Check if the customer exists
   1. Find customer by the given id
   2. If the customer was not found, return 404 NotFound(customer id)
4. Update the given information
5. Return the updated customer object

## Deactivate Customer

The customer can deactivate the account, if they no longer wish to receive newsletter or any other interactions.

## Delete Customer

The admin can delete a customer from the system, if the customer no longer wishes to have their details stored.

# Orders

* Calculate subtotal
  + Each orderline on the order should include a subtotal (quantity \* product.price)
* Calculate total price
  + The total price of the order, including shipping and handling fees
  + The shipping and handling fee should be included above, separately.
* Validate ShippedTo has a value
  + The ShippedTo is the customer address, unless something else is specified
* Validate Order Information on Create
  + If the customer status is NoAccount, this part is skipped
  + Is the order null?
  + Does the order have orderlines?
* Validate Updating an Order
  + If the order status is received or processing the order can be updated, else return
  + If the order has no id, return
* Validate Order is Shipped
  + If the order status is Shipped and ShippedOn has value, else return
* The DateTime properties
  + **PlacedOn:** When the order is created by the customer, the date is set
  + **ProcessedOn:** When the order is going to be processed by the admin, the date is set
  + **ShippedOn:** When the order is being shipped by the admin, the date is set
* Order Status
  + Received
    - The order is received and the PlacedOn date is set
    - The order is not yet being processed and can still be cancelled
  + Processing
    - The order is now being processed and the ProcessedOn date is set
    - The order is not yet shipped and can still be cancelled
      * If cancelled, the shipping and handling fee is not returned
  + Shipped
    - The order is now being shipped and the ShippedOn date is set
    - The order has left the storage and can NOT be cancelled
    - The order can still be returned
  + Returned
    - The order has been sent and the customer returned the order.
  + Cancelled
    - The order is still in the warehouse and the customer cancelled the order.

## Create Order

## Get Order(s)

## Patch Order

## Update Order

## Delete Order

# Products

* Required Fields
  + Name, Brand, Description, stock, price
* Product Status
  + In Stock
    - The product is currently in stock and purchasable
  + Out of Stock
    - The product is sold out and has not been ordered to stock by admin
  + In Order
    - The product is out of stock, but has been ordered to stock by admin
  + Expired
    - The product is expired from the product catalogue
    - The product is sold until out of stock

## Create Product

The admin can create a new product and add it to the system. The steps are as follows:

1. Authenticate admin
   1. If the request is not made by an admin, return 401 Unauthorized
2. The product object is received and validated
   1. The name of the product can not be empty or whitespace and should be at least 2 characters long
   2. The brand of the product cannot be empty or whitespace
   3. The description of the product can not be empty or whitespace and should be at least 10 characters long
   4. The stock needs to be 0 or more. No negative numbers allowed
   5. The price needs to be greater than 0.
3. The product object is inserted into the database
   1. Once inserted, the product id should be returned
      1. If the id is 0 or less, insertion failed
      2. If the id is 1 or greater, insertion success
4. The product object is returned with the generated id
   1. Return 201 Created(uri, product)

## Get Product(s)

## Patch Product

## Update Product

## Delete Product