Inventory Management API Documentation

July 6, 2025

Contents

1 Introduction

This document provides comprehensive documentation for the Inventory Management API, hosted at https://formulario-q9c8.onrender.com/api/. The API, built with Node.js, Express, and MongoDB, manages categories, products, clients, catalogs, notifications, suppliers, and sales. Each endpoint is detailed with its HTTP method, URI, description, parameters, responses, and example requests/responses.

2 Categories

2.1 Get Categories Summary

Method and URI: GET `/api/categories/summary`

Description: Retrieves a summary of all categories with their product counts.

Parameters: None

Responses:
- 200: Success - Returns an array of category summaries (`[categoryId, name, productCount]`)

Example Request:
```
curl -X GET https://formulario-q9c8.onrender.com/api/categories
```

Example Response:
```
[
    {"categoryId": "507f1f77bcf86cd799439011", "name": "Electronics", "produ
    {"categoryId": "507f1f77bcf86cd799439012", "name": "Clothing", "productC
]
```

2.2 Merge Categories

Method and URI: POST `/api/categories/merge`

Description: Merges two categories, moving products from the source to the target category and deleting the source.

Parameters (Body): `{sourceCategoryId: String, targetCategoryId: String}`

Responses:
- 200: Success - Returns the updated target category
- 404: Not Found - Source or target category not found

Example Request:
```
curl -X POST https://formulario-q9c8.onrender.com/api/categorie
-H "Content-Type: application/json" \
-d '{"sourceCategoryId": "507f1f77bcf86cd799439011", "targetCategoryId": "50
```

Example Response:
- 200: {"$_id$" : "$507f1f77bcf86cd799439012$", "$name$" : "$Clothing$"}404 : `{"error": "Category not found"}`

2.3 Get All Categories

Method and URI: GET `/api/categories`

Description: Retrieves all categories.

Parameters: None

Responses:
- 200: Success - Returns an array of categories

Example Request:
```
curl -X GET https://formulario-q9c8.onrender.com/api/categories
```

Example Response:
```
[
      {"_id": "507f1f77bcf86cd799439011", "name": "Electronics"},
      {"_id": "507f1f77bcf86cd799439012", "name": "Clothing"}
  ]
```

2.4 Create Category

Method and URI: POST `/api/categories`

Description: Creates a new category.

Parameters (Body): `{name: String}`

Responses:
- 201: Created - Returns the created category
- 400: Bad Request - Invalid data

Example Request:
```
curl -X POST https://formulario-q9c8.onrender.com/api/categorie
  -H "Content-Type: application/json" \
  -d '{"name": "Electronics"}'
```

Example Response: {"$_id$" : "$507f1f77bcf86cd799439011$", "$name$" : "$Electronics$"}

2.5 Get Category by ID

Method and URI: GET `/api/categories/:id`

Description: Retrieves a specific category by ID.

Parameters (Path): `id: String (MongoDB ObjectId)`

Responses:
- 200: Success - Returns the category
- 404: Not Found - Category not found

Example Request:
```
curl -X GET https://formulario-q9c8.onrender.com/api/categories
```

Example Response: {"$_id$" : "$507f1f77bcf86cd799439011$", "$name$" : "$Electronics$"}

2.6 Assign Product to Category

Method and URI: POST `/api/categories/:id/assign-product`

Description: Assigns a product to a category.

Parameters (Path): `id: String (MongoDB ObjectId of category)`

Parameters (Body): `{productId: String}`

Responses:

- 200: Success - Returns the updated product
- 404: Not Found - Category or product not found

Example Request:

```
curl -X POST https://formulario-q9c8.onrender.com/api/categorie
  -H "Content-Type: application/json" \
  -d '{"productId": "507f1f77bcf86cd799439013"}'
```

Example Response: {"$_id$" : "507$f1f77bcf86cd$799439013", "$name$" : "$Laptop$", "$categoryId$" : "507$f1f77bcf86cd$799439011", ...}

2.7 Get Product Count by Category

Method and URI: GET `/api/categories/:id/product-count`

Description: Returns the number of products in a category.

Parameters (Path): `id: String (MongoDB ObjectId)`

Responses:

- 200: Success - Returns the product count
- 404: Not Found - Category not found

Example Request:

```
curl -X GET https://formulario-q9c8.onrender.com/api/categories
```

Example Response: `{"categoryId": "507f1f77bcf86cd799439011", "productCount": 10}`

2.8 Archive Category

Method and URI: PUT `/api/categories/:id/archive`

Description: Sets a category's status to "archived".

Parameters (Path): `id: String (MongoDB ObjectId)`

Responses:

- 200: Success - Returns the archived category
- 404: Not Found - Category not found

Example Request:

```
curl -X PUT https://formulario-q9c8.onrender.com/api/categories
```

Example Response: {"$_id$" : "507$f1f77bcf86cd$799439011", "$name$" : "$Electronics$", "$status$" : "$archived$"}

2.9 Update Category

Method and URI: PUT `/api/categories/:id`

Description: Updates a category's details.

Parameters (Path): `id: String (MongoDB ObjectId)`

Parameters (Body): `{name: String}`

Responses:

- 200: Success - Returns the updated category
- 404: Not Found - Category not found

Example Request:

```
curl -X PUT https://formulario-q9c8.onrender.com/api/categories
    -H "Content-Type: application/json" \
    -d '{"name": "Updated Electronics"}'
```

Example Response: {"$_{i}d$" : "$507f1f77bcf86cd799439011$", "$name$" : "$UpdatedElectronics$"}

2.10 Delete Category

Method and URI: DELETE `/api/categories/:id`

Description: Deletes a category.

Parameters (Path): `id: String (MongoDB ObjectId)`

Responses:

- 204: No Content - Category deleted
- 404: Not Found - Category not found

Example Request:

```
curl -X DELETE https://formulario-q9c8.onrender.com/api/categor
```

Example Response: (No content)

3 Products

3.1 Get Low Stock Products

Method and URI: GET `/api/products/low-stock`

Description: Retrieves products with stock below or equal to a threshold.

Parameters (Query): `threshold: Number (optional, default 10)`

Responses:

- 200: Success - Returns an array of low-stock products

Example Request:

```
curl -X GET https://formulario-q9c8.onrender.com/api/products/l
```

Example Response:
```
[
    {
        "_id": "507f1f77bcf86cd799439013",
        "name": "Laptop",
        "stock": 3,
        "categoryId": {"_id": "507f1f77bcf86cd799439011", "name": "Electroni
        "purchaseUnit": "unit",
        "quantityIncluded": 1,
        "saleUnit": "unit",
        "purchasePrice": 500,
        "salePrice": 1000
    }
]
```

3.2 Bulk Price Update

Method and URI: POST `/api/products/bulk-price-update`

Description: Increases prices of products in a category by a percentage.

Parameters (Body): `{categoryId: String, percentage: Number}`

Responses:

- 200: Success - Returns the number of modified products
- 400: Bad Request - Invalid percentage
- 404: Not Found - Category not found

Example Request:
```
curl -X POST https://formulario-q9c8.onrender.com/api/products/
-H "Content-Type: application/json" \
-d '{"categoryId": "507f1f77bcf86cd799439011", "percentage": 10}'
```

Example Response: `{"modifiedCount": 5}`

3.3 Get All Products

Method and URI: GET `/api/products`

Description: Retrieves all products.

Parameters: None

Responses:

- 200: Success - Returns an array of products

Example Request:
```
curl -X GET https://formulario-q9c8.onrender.com/api/products
```

Example Response:
```
[
    {
        "_id": "507f1f77bcf86cd799439013",
        "name": "Laptop",
        "categoryId": {"_id": "507f1f77bcf86cd799439011", "name": "Electroni
        "purchaseUnit": "unit",
        "quantityIncluded": 1,
        "saleUnit": "unit",
        "purchasePrice": 500,
        "salePrice": 1000,
        "stock": 100
    }
]
```

3.4 Create Product

Method and URI: POST `/api/products`

Description: Creates a new product.

Parameters (Body): `{name: String, categoryId: String, purchaseUnit: String, quantityIncluded: Number, saleUnit: String, purchasePrice: Number, salePrice: Number, stock: Number}`

Responses:

- 201: Created - Returns the created product
- 400: Bad Request - Invalid data

Example Request:

```
curl -X POST https://formulario-q9c8.onrender.com/api/products
-H "Content-Type: application/json" \
-d '{"name": "Laptop", "categoryId": "507f1f77bcf86cd799439011", "purchaseUn
```

Example Response: {"$_id$" : "$507f1f77bcf86cd799439013$", "$name$" : "$Laptop$", "$categoryId$" : "$507f1f77bcf86cd799439011$", ...}

3.5 Get Product by ID

Method and URI: GET `/api/products/:id`

Description: Retrieves a specific product by ID.

Parameters (Path): `id: String (MongoDB ObjectId)`

Responses:

- 200: Success - Returns the product
- 404: Not Found - Product not found

Example Request:

```
curl -X GET https://formulario-q9c8.onrender.com/api/products/5
```

Example Response: {"$_id$" : "$507f1f77bcf86cd799439013$", "$name$" : "$Laptop$", "$categoryId$" : ..., ...}

3.6 Update Product Stock

Method and URI: PUT `/api/products/:id/stock`

Description: Updates the stock of a product.

Parameters (Path): `id: String (MongoDB ObjectId)`

Parameters (Body): `{stock: Number}`

Responses:

- 200: Success - Returns the updated product
- 400: Bad Request - Negative stock
- 404: Not Found - Product not found

Example Request:

```
curl -X PUT https://formulario-q9c8.onrender.com/api/products/5
-H "Content-Type: application/json" \
-d '{"stock": 50}'
```

Example Response: {"$_{i}d" : "507f1f77bcf86cd799439013", "name" : "Laptop", "stock" : 50, ...\}$

3.7 Archive Product

Method and URI: PUT `/api/products/:id/archive`

Description: Sets a product's status to "archived".

Parameters (Path): `id: String (MongoDB ObjectId)`

Responses:

- 200: Success - Returns the archived product
- 404: Not Found - Product not found

Example Request:

```
curl -X PUT https://formulario-q9c8.onrender.com/api/products/5
```

Example Response: {"$_{i}d" : "507f1f77bcf86cd799439013", "name" : "Laptop", "status" : "archived", ...\}$

3.8 Update Product

Method and URI: PUT `/api/products/:id`

Description: Updates a product's details.

Parameters (Path): `id: String (MongoDB ObjectId)`

Parameters (Body): `{name: String, categoryId: String, purchaseUnit: String, quantityIncluded: Number, saleUnit: String, purchasePrice: Number, salePrice: Number, stock: Number}`

Responses:

- 200: Success - Returns the updated product

- 404: Not Found - Product not found

Example Request:
```
curl -X PUT https://formulario-q9c8.onrender.com/api/products/5
-H "Content-Type: application/json" \
-d '{"name": "Updated Laptop", "stock": 75}'
```

Example Response: {"$_id" : "507f1f77bcf86cd799439013", "name" : "UpdatedLaptop", "stock" :$ $75, ...$}

3.9 Delete Product

Method and URI: DELETE `/api/products/:id`

Description: Deletes a product.

Parameters (Path): `id: String (MongoDB ObjectId)`

Responses:
- 204: No Content - Product deleted
- 404: Not Found - Product not found

Example Request:
```
curl -X DELETE https://formulario-q9c8.onrender.com/api/product
```

Example Response: (No content)

4 Clients

4.1 Get Top Clients by Sales

Method and URI: GET `/api/clients/top-sales`

Description: Retrieves the top 5 clients by number of sales.

Parameters: None

Responses:
- 200: Success - Returns an array of top clients

Example Request:
```
curl -X GET https://formulario-q9c8.onrender.com/api/clients/to
```

Example Response:
```
[
    {
        "clientId": "507f1f77bcf86cd799439014",
        "clientName": "John Doe",
        "totalSales": 10
    },
    {
        "clientId": "507f1f77bcf86cd799439015",
        "clientName": "Jane Smith",
        "totalSales": 8
        }
    ]
```

4.2 Get All Clients

Method and URI: GET `/api/clients`

Description: Retrieves all clients.

Parameters: None

Responses:
- 200: Success - Returns an array of clients

Example Request:
```
curl -X GET https://formulario-q9c8.onrender.com/api/clients
```

Example Response:
```
    [
        {
            "_id": "507f1f77bcf86cd799439014",
            "taxid": "123456789",
            "fullname": "John Doe",
            "address": "123 Main St",
            "references": "Ref1",
            "phone": "1234567890",
            "email": "john@example.com"
        }
    ]
```

4.3 Create Client

Method and URI: POST `/api/clients`

Description: Creates a new client.

Parameters (Body): `{taxid: String, fullname: String, address: String, references: String, phone: String, email: String}`

Responses:
- 201: Created - Returns the created client
- 400: Bad Request - Invalid data

Example Request:
```
curl -X POST https://formulario-q9c8.onrender.com/api/clients \
    -H "Content-Type: application/json" \
    -d '{"taxid": "123456789", "fullname": "John Doe", "address": "123 Main St",
```

Example Response: {"$_id$" : "$507f1f77bcf86cd799439014$", "$fullname$" : "$JohnDoe$", ...}

4.4 Get Client by ID

Method and URI: GET `/api/clients/:id`

Description: Retrieves a specific client by ID.

Parameters (Path): `id: String (MongoDB ObjectId)`

Responses:

- 200: Success - Returns the client
- 404: Not Found - Client not found

Example Request: `curl -X GET https://formulario-q9c8.onrender.com/api/clients/50`

Example Response: {"$_id$" : "$507f1f77bcf86cd799439014$", "$fullname$" : "$JohnDoe$", ...}

4.5 Get Client Purchase History

Method and URI: GET `/api/clients/:id/purchase-history`

Description: Retrieves the sales associated with a client.

Parameters (Path): `id: String (MongoDB ObjectId)`

Responses:

- 200: Success - Returns an array of sales
- 404: Not Found - No purchases found for the client

Example Request: `curl -X GET https://formulario-q9c8.onrender.com/api/clients/50`

Example Response:

```
[
    {
        "_id": "507f1f77bcf86cd799439016",
        "clientId": "507f1f77bcf86cd799439014",
        "items": [...],
        "total": 100
    }
]
```

4.6 Assign Sale to Client

Method and URI: POST `/api/clients/:id/assign-sale`

Description: Assigns a sale to a client.

Parameters (Path): `id: String (MongoDB ObjectId of client)`

Parameters (Body): `{saleId: String}`

Responses:

- 200: Success - Returns the updated sale
- 404: Not Found - Client or sale not found

Example Request:
```
curl -X POST https://formulario-q9c8.onrender.com/api/clients/5
-H "Content-Type: application/json" \
-d '{"saleId": "507f1f77bcf86cd799439016"}'
```

Example Response: {"$_{i}d$" : "507f1f77bcf86cd799439016", "$clientId$" : "507f1f77bcf86cd799439014", ...}

4.7 Update Client Credit

Method and URI: PUT `/api/clients/:id/credit`

Description: Updates a client's credit.

Parameters (Path): `id: String (MongoDB ObjectId)`

Parameters (Body): `{credit: Number}`

Responses:
- 200: Success - Returns the updated client
- 400: Bad Request - Negative credit
- 404: Not Found - Client not found

Example Request:
```
curl -X PUT https://formulario-q9c8.onrender.com/api/clients/50
-H "Content-Type: application/json" \
-d '{"credit": 1000}'
```

Example Response: {"$_{i}d$" : "507f1f77bcf86cd799439014", "$fullname$" : "$JohnDoe$", "$credit$" : 1000, ...}

4.8 Deactivate Client

Method and URI: PUT `/api/clients/:id/deactivate`

Description: Sets a client's status to "inactive".

Parameters (Path): `id: String (MongoDB ObjectId)`

Responses:
- 200: Success - Returns the deactivated client
- 404: Not Found - Client not found

Example Request:
```
curl -X PUT https://formulario-q9c8.onrender.com/api/clients/50
```

Example Response: {"$_{i}d$" : "507f1f77bcf86cd799439014", "$fullname$" : "$JohnDoe$", "$status$" : "$inactive$", ...}

4.9 Update Client

Method and URI: PUT `/api/clients/:id`

Description: Updates a client’s details.

Parameters (Path): `id: String (MongoDB ObjectId)`

Parameters (Body): `{taxid: String, fullname: String, address: String, references: String, phone: String, email: String}`

Responses:
- 200: Success - Returns the updated client
- 404: Not Found - Client not found

Example Request:
```
curl -X PUT https://formulario-q9c8.onrender.com/api/clients/50
  -H "Content-Type: application/json" \
  -d '{"fullname": "Updated John Doe"}'
```

Example Response: {"$_{i}d$" : ”507f1f77bcf86cd799439014”, ”fullname” : ”UpdatedJohnDoe”, ...}

4.10 Delete Client

Method and URI: DELETE `/api/clients/:id`

Description: Deletes a client.

Parameters (Path): `id: String (MongoDB ObjectId)`

Responses:
- 204: No Content - Client deleted
- 404: Not Found - Client not found

Example Request:
```
curl -X DELETE https://formulario-q9c8.onrender.com/api/clients
```

Example Response: (No content)

5 Catalogs

5.1 Get All Catalogs

Method and URI: GET `/api/catalogs`

Description: Retrieves all catalogs.

Parameters: None

Responses:

5.2 Create Catalog

Method and URI: POST `/api/catalogs`

Description: Creates a new catalog.

Parameters (Body): `{name: String, filePath: String}`

Responses:

5.3 Get Catalog by ID

Method and URI: GET `/api/catalogs/:id`
Description: Retrieves a specific catalog by ID.
Parameters (Path): `id: String (MongoDB ObjectId)`

Responses:

5.4 Get Catalog Products

Method and URI: GET `/api/catalogs/:id/products`
Description: Retrieves the products assigned to a catalog.
Parameters (Path): `id: String (MongoDB ObjectId)`

Responses:

5.5 Assign Product to Catalog

Method and URI: POST `/api/catalogs/:id/assign-product`
Description: Assigns a product to a catalog.
Parameters (Path): `id: String (MongoDB ObjectId of catalog)`
Parameters (Body): `{productId: String}`

Responses:

5.6 Remove Product from Catalog

Method and URI: POST `/api/catalogs/:id/remove-product`
Description: Removes a product from a catalog.
Parameters (Path): `id: String (MongoDB ObjectId of catalog)`
Parameters (Body): `{productId: String}`

Responses:

5.7 Duplicate Catalog

Method and URI: POST `/api/catalogs/:id/duplicate`
Description: Creates a copy of a catalog.
Parameters (Path): `id: String (MongoDB ObjectId)`

Responses:

5.8 Update Catalog Status

Method and URI: PUT `/api/catalogs/:id/status`
Description: Updates a catalog's status (active/inactive).
Parameters (Path): `id: String (MongoDB ObjectId)`
Parameters (Body): `{status: String ("active" or "inactive")}`

Responses:

5.9 Update Catalog

Method and URI: PUT `/api/catalogs/:id`
Description: Updates a catalog's details.
Parameters (Path): `id: String (MongoDB ObjectId)`
Parameters (Body): `{name: String, filePath: String}`

Responses: 5.10 Delete Catalog

Method and URI: DELETE `/api/catalogs/:id`
Description: Deletes a catalog.
Parameters (Path): `id: String (MongoDB ObjectId)`

Responses: 6 Notifications

6.1 Get All Notifications

Method and URI: GET `/api/notifications`
Description: Retrieves all notifications.
Parameters: None

Responses: 6.2 Create Low Stock Notification

Method and URI: POST `/api/notifications/low-stock`
Description: Creates notifications for products with low stock.
Parameters (Body): `{threshold: Number (optional, default 10)}`

Responses: 6.3 Create Notification

Method and URI: POST `/api/notifications`
Description: Creates a new notification.
Parameters (Body): `{icon: String, message: String, read: Boolean (optional, default false)}`

Responses: 6.4 Get Notification by ID

Method and URI: GET `/api/notifications/:id`
Description: Retrieves a specific notification by ID.
Parameters (Path): `id: String (MongoDB ObjectId)`

Responses: 6.5 Mark Notification as Read

Method and URI: PUT `/api/notifications/:id/read`
Description: Marks a notification as read.
Parameters (Path): `id: String (MongoDB ObjectId)`

Responses: 6.6 Update Notification

Method and URI: PUT `/api/notifications/:id`

Description: Updates a notification's details.

Parameters (Path): `id: String (MongoDB ObjectId)`

Parameters (Body): `{icon: String, message: String, read: Boolean}`

Responses: 6.7 Delete Notification

Method and URI: DELETE `/api/notifications/`

Description: Deletes a notification.

Parameters (Path): `id: String (MongoDB ObjectId)`

Responses: 7 Suppliers

7.1 Get Suppliers Summary

Method and URI: GET `/api/suppliers/summar`

Description: Retrieves a summary of suppliers with their product counts.

Parameters: None

Responses: 7.2 Get All Suppliers

Method and URI: GET `/api/suppliers`

Description: Retrieves all suppliers.

Parameters: None

Responses: 7.3 Create Supplier

Method and URI: POST `/api/suppliers`

Description: Creates a new supplier.

Parameters (Body): `{idNumber: String, company: String, contactName: String, phone: String, bankAccount: String, bankName: String, catalogId: String}`

Responses:
7.4 Get Supplier by ID

Method and URI: GET `/api/suppliers/`

Description: Retrieves a specific supplier by ID.

Parameters (Path): `id: String (MongoDB ObjectId)`

Responses:
7.5 Get Supplier Product Count

Method and URI: GET `/api/supplier`

Description: Returns the number of products in a supplier's catalog.

Parameters (Path): `id: String (MongoDB ObjectId)`

Responses:
7.6 Assign Catalog to Supplier

Method and URI: POST `/api/suppliers/:id/assign-catalo`

Description: Assigns a catalog to a supplier.

Parameters (Path): `id: String (MongoDB ObjectId of supplier)`

Parameters (Body): `{catalogId: String}`

Responses:
7.7 Update Supplier Status

Method and URI: PUT `/api/suppliers/:id/status`

Description: Updates a supplier's status (active/inactive).

Parameters (Path): `id: String (MongoDB ObjectId)`

Parameters (Body): `{status: String ("active" or "inactive")}`

Responses:

7.8 Update Supplier

Method and URI: PUT `/api/suppliers/:id`

Description: Updates a supplier's details.

Parameters (Path): `id: String (MongoDB ObjectId)`

Parameters (Body): `{idNumber: String, company: String, contactName: String, phone: String, bankAccount: String, bankName: String, catalogId: String}`

Responses:

7.9 Delete Supplier

Method and URI: DELETE `/api/suppliers/:id`

Description: Deletes a supplier.

Parameters (Path): `id: String (MongoDB ObjectId)`

Responses:

8 Sales

8.1 Get All Sales

Method and URI: GET `/api/sales`

Description: Retrieves all sales.

Parameters: None

Responses:

8.2 Create Sale with Validation

Method and URI: POST `/api/sales/valida`

Description: Creates a sale with stock and client validation.

Parameters (Body):

```
    "clientId": Str
    "items": [
        {
            "produc
            "name"
            "quant
            "unitP
            "total
        }
    ],
    "total": Numbe
}
```

Responses:

8.3 Get Sales by Date Range

Method and URI: POST `/api/sales/date`

Description: Retrieves sales within a date range with totals.

Parameters (Body): `{startDate: String (ISO8601), endDate: String (ISO8601)}`

Responses:

8.4 Create Sale

Method and URI: POST `/api/sales`

Description: Creates a new sale without stock validation.

Parameters (Body):

```
        "clientId":
        "items": [
            {
                "pr
                "na
                "qu
                "ur
                "to
            }
        ],
        "total": Nu
    }
```

Responses:

8.5 Get Sale by ID

Method and URI: GET `/api/sales/:`

Description: Retrieves a specific sale by ID.

Parameters (Path): `id: String (MongoDB ObjectId)`

Responses:

8.6 Get Sale Profit

Method and URI GET `/api/sales/:id/`

Description: Calculates the profit for a sale.

Parameters (Path `id:`

`String (MongoDB ObjectId)`

Responses:

8.7 Apply Discount to Sale

Method and U
PUT `/api/sales/:i`

Description:
Applies a percentage discount to a sale.

Parameters (Pa
`id: String (MongoDB ObjectId)`

Parameters (Bo
`{discount: Number (0-100)}`

Responses:

8.8 Upda Sale

Method and
PUT `/api/sales/`

Description:
Updates a sale's

de-
tails.

Parameters (

id:
String
(MongoDB
ObjectId)

Parameters (

```
        "cl
        "it



            ],
        "to
    }
```

Responses:

8.9 Can Sale

Method an
DELETE
/api/sale

Descriptio
Cancels a sale and re-stores prod-uct stock.

Parameters
id:
String
(MongoDB
ObjectId)

Responses

8.10 D
Sale

Method a
DELET
`/api/sa`

Descripti
Deletes a sale with-out restor-ing stock.

Paramete
`id: String (MongoD ObjectI`

Response