**Order Management for E-Commerce Application**

**Contents of the Project File:**

Within the project file, the following are included:

**Source Codes:** The raw code files that make up the application.

**Compiled Project File:** The build or executable of the application.

**Postman Collection File:** A file containing all the requests and responses for API testing and documentation.

**Exported PostgreSQL Database File:** The exported version of the database used in the project, allowing for replication or backup.

Before using the application, ensure you have set up a valid PostgreSQL database.

Project serves at **8080** ports.

In the **.env** File;

POSTGRESUSER=postgres

POSTGRESPASSWORD=Parola.123

POSTGRESDBNAME=itm

POSTRESSSLMODE=disable

POSTGRESHOST=127.0.0.1

POSTGRESPORT=5432

These are for database connection. You can also configure for your database information.

LOGMODE=true

With this setting, you can determine if actions will be inserted in the "loggers" table.

For endpoints I have inserted some pre-determined tokens.(also customers for the tokens) For the future tokens, authentication service should be implemented.

For admin actions : "**admintoken**"

For user (customer) actions : "**usertoken1**" , "**usertoken2**", "**usertoken3**"

**Description:**

Order management is a fundamental function of an e-commerce application. This project involves developing a Go application for order management operations on an e-commerce platform.

**Main Features:**

Customer Management:

Retrieval of customer information.

Creation of a new customer.

Updating existing customer information.

**Product Management:**

Retrieval of product information.

Creation of a new product.

Updating existing product information.

Deletion of a product.

Ability to set variable discount rates and VAT values.

**Order Management:**

Retrieval of order information.

Creation of a new order.

Updating an order.

Cancellation of an order.

**Integrations:**

**Payment Service Integration:** The system listens to payment gateways and approves the payment if the recorded method and ID match the payment status.

**Cargo/Shipping Companies Integration:** The system listens to shipping companies and makes necessary updates automatically based on the received ID.

**Advanced Transaction Logging:** All operations are monitored via channels and are recorded in a logger table if set to "true".

**Security and Access Control:** Token-based validation for every request.

**Technical Details:**

**High Volume Request Management:** The system can process high-volume requests from users.

**Optimistic and Pessimistic Locking Methods:** Methods used to ensure the security of orders.

**Concurrent Order Processing:** The system can process multiple orders simultaneously.

**Dynamic Status Updates:** Automatic order status updates based on dynamic requests from payment and shipping companies.

**Benefits:**

**Effective Order Management:** Allows for the concurrent processing of orders in high volume.

**Integrated Solution:** The system offers order management on a single platform by integrating with external services such as payment gateways and shipping companies.

**Enhanced Security:** Provides secure access through token-based authorization.

**Comprehensive Monitoring:** All operations are tracked in a logging table, ensuring easy detection of potential issues or errors.

**Common Error Responses For Endpoints**

***Invalid Token***

**Code:** 401 Unauthorized

**Content:**

{

"data": {

"response": "Need Valid Token",

"status": "ERROR",

"type": "OK"

}

}

This can be caused by the wrong/empty/expired token for admin actions..

***SQL Error***

**Code:** 500 Internal Server Error

**Content:**

{

"data": {

"response": "SQL Error Text In Here",

"status": "ERROR",

"type": "OK"

}

}

***User Not Found***

**Code:** 401 Unauthorized

**Content:**

{

"data": {

"response": "User Not Found",

"status": "ERROR",

"type": "OK"

}

}

This can be caused by the wrong/empty/expired token for user actions.

***Json Error***

**Code:** 403 Bad Request

**Content:**

{

"data": {

"response": "String Json Search Error",

"status": "ERROR",

"type": "OK"

}

}

{

"data": {

"response": "Float or Integer Json Search Error",

"status": "ERROR",

"type": "OK"

}

}

{

"data": {

"response": "Boolean Json Search Error",

"status": "ERROR",

"type": "OK"

}

}

This can be caused by the wrong or missing type of json key.

***Not Valid Email & Phone***

**Code:** 403 Bad Request

**Content:**

{

"data": {

"response": "Need Valid Email Address / Phone Number",

"status": "ERROR",

"type": "OK"

}

}

This can be caused by the not valid input.  
Expected Phone Number : Must be only contains digits.  
Expected Email Address : “ **`^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6}$`**” must match with this regex pattern.

***Not Changable Status***

**Code:** 403 Bad Request

**Content:**

{

"data": {

"response": "Not Changable Status",

"status": "ERROR",

"type": "OK"

}

}

This error returns because of the current status.  
User cannot change status (cancel order) after item became “preparing”  
Admin cannot change status after the user cancels the status.

**Get Customers Endpoint**

URL : {{baseurl}}[/customers](http://localhost:8080/customers)

Method : GET

URL Parameters

Required: token=[string]

Success Response

Code: 200 OK

Content:

{

"data": {

"response": {

"customers": {

"1": {

"address": "Çankaya Kuğulu, Park, 06690 Çankaya/Ankara",

"email": "aliyilmaz@ali.com",

"firstName": "Ali",

"id": 1,

"lastName": "Yılmaz",

"phone": "90555111223311"

},

...

},

"totalCount": 5

},

"type": "OK"

}

}

**Create Customer Endpoint**

URL : {{baseurl}}/customers

Method : POST

Example Request

{

"data":{

"token":"admintoken",

"request":{

"customer":{

"firstName":"Mehmet",

"lastName":"Kesmez",

"phone":9055511122999,

"email":"mehmet@kesmez.com",

"address":"Çankaya, Atakule, 06690 Çankaya/Ankara"

}

}

}

}

Success Response

Code: 200 OK

Content:

{

"data": {

"response": "NEW CUSTOMER INSERTED",

"status": "OK",

"type": "OK"

}

}

Notes

This endpoint allows for the creation of a new customer in the database.

Admin token is required to access this endpoint.

**Update Customer Endpoint**

URL : {{baseurl}}/customers

Method : PUT

Example Request

{

"data":{

"token":"admintoken",

"request":{

"customer":{

"id":4,

"firstName":"Fatma",

"lastName":"Girik",

"phone":123456789,

"email":"fatma@girik.com",

"address":"Çankaya, Merkez, 06690 Çankaya/Ankara"

}

}

}

}

Success Response

Code: 200 OK

Content:

{

"data": {

"response": "CUSTOMER UPDATED",

"status": "OK",

"type": "OK"

}

}

Notes

This endpoint allows for updating an existing customer in the database using the provided customer ID.

Admin token is required to access this endpoint.

**Get Products Endpoint**

URL : {{baseurl}}/products

Method :GET

URL Parameters

Required:

token=[string]

Success Response

Code: 200 OK

Response Content:

{

"data": {

"response": {

"products": {

"4": {

"basePrice": 42000,

"category": "telefon",

"createdAt": "2023-10-30T09:39:15.254408Z",

"discount": 2,

"id": 4,

"isValid": true,

"name": "Apple iPhone 12 Pro",

"payPrice": 49392

},

...

},

"totalCount": 6

},

"type": "OK"

}

}

**Create Product Endpoint**

URL : {{baseurl}}/products

Method : POST

Example Request

{

"data":{

"token":"admintoken",

"request":{

"product":{

"name":"Apple iPhone 15 Pro Max",

"category":"telefon",

"isValid":true,

"basePrice":70000,

"discount":10,

"vat":0

}

}

}

}

Success Response

Code: 200 OK

Response Content:

{

"data": {

"response": "NEW PRODUCT INSERTED",

"status": "OK",

"type": "OK"

}

}

Notes

This endpoint allows for the creation of a new product in the database.

Admin token is required to access this endpoint.

**Update Product Endpoint**

URL : {{baseurl}}/products

Method : PUT

Example Request

{

"data":{

"token":"admintoken",

"request":{

"product":{

"id":9,

"name":"Apple iPhone 14 Pro Max",

"category":"telefon",

"isValid":true,

"basePrice":70000,

"discount":10,

"vat":10

}

}

}

}

Success Response

Code: 200 OK

Response Content:

{

"data": {

"response": "PRODUCT UPDATED",

"status": "OK",

"type": "OK"

}

}

Notes

This endpoint allows for updating an existing product in the database using the provided product ID.

Admin token is required to access this endpoint.

**Delete (Deactivate) Product Endpoint**

URL : {{baseurl}}/products

Method : DELETE

Example Request

{

"data":{

"token":"admintoken",

"request":{

"product":{

"id":3

}

}

}

}

Success Response

Code: 200 OK

Response Content:

{

"data": {

"response": "PRODUCT DELETED",

"status": "OK",

"type": "OK"

}

}

Notes

This endpoint does not physically delete the product from the database. Instead, it sets the isValid field of the product to false, effectively deactivating it.

Admin token is required to access this endpoint.

**Get Orders Endpoint**

URL : {{baseurl}}/orders

Method : GET

URL Parameters

Required:

token=[string]

p=[integer] (Page number for paginated results)

Success Response

Code: 200 OK

Content:

{

"data": {

"response": {

"maxPage": 3443,

"orders": {

"2a1262b08739267fc06c": {

"customer": {

"email": "mehmetkaya@mehmet.com",

"firstName": "Mehmet",

"lastName": "Kaya",

"phone": "90555111223411"

},

"details": {

"1191645": {

"productBasePrice": 42000,

"productDiscount": 2,

"productId": 4,

"productName": "Apple iPhone 12 Pro",

"productPayPrice": 49392,

"productVat": 20,

"quantity": 4

},

"id": 259000,

"insertDate": "2023-10-31T03:21:24.88409Z",

"orderId": "2a1262b08739267fc06c",

"status": "Waiting for payment",

"totalPrice": 49392,

"transactionMethod": "MethodX"  
 }

},

...

},

"responseCount": 30,

"totalCount": 103275

},

"type": "OK"

}

}

**Create Orders Endpoint**

URL : {{baseurl}}/orders

Method : POST

Example Request

{

"data":{

"token":"usertoken1",

"request":{

"order":{

"transactionMethod":"MethodX",

"productlist":"1,2,2,3,3,3,4,4,4,4,5,5,5,5,5,6,6,6,6,6,6,7,7,7,7,7,7,7"

}

}

}

}

Success Response

Code: 200 OK

Content:

{

"data": {

"response": "NEW ORDER INSERTED",

"status": "OK",

"type": "OK"

}

}

Notes

This endpoint allows for the creation of a new order in the database.

A user token is required to access this endpoint.

The productlist field contains a comma-separated list of product IDs representing the products in the order. The quantity of each product is determined by the number of times its ID appears in the list.

**Update Orders Endpoint**

URL : {{baseurl}}/orders

Method : PUT

Data Parameters

Example Request

{

"data":{

"token":"admintoken",

"request":{

"order":{

"id":259114,

"status":5

}

}

}

}

Success Response

Code: 200 OK

Content:

{

"data": {

"response": "ORDER UPDATED",

"status": "OK",

"type": "OK"

}

}

Notes

This endpoint allows for updating the status of an existing order in the database.

Admin token is required to access this endpoint. Only admins can update order status.

The status field represents the new status to be set for the order.

**Get User’s Orders Endpoint**

URL : {{baseurl}}/orders

Method : GET

URL Parameters

Required:

token=[string]

p=[integer] (Page number for paginated results)

Success Response

Code: 200 OK

Content:

{

"data": {

"response": {

"maxPage": 3443,

"orders": {

"2a1262b08739267fc06c": {

"customer": {

"email": "mehmetkaya@mehmet.com",

"firstName": "Mehmet",

"lastName": "Kaya",

"phone": "90555111223411"

},

"details": {

"1191645": {

"productBasePrice": 42000,

"productDiscount": 2,

"productId": 4,

"productName": "Apple iPhone 12 Pro",

"productPayPrice": 49392,

"productVat": 20,

"quantity": 4

},

"id": 259000,

"insertDate": "2023-10-31T03:21:24.88409Z",

"orderId": "2a1262b08739267fc06c",

"status": "Waiting for payment",

"totalPrice": 49392,

"transactionMethod": "MethodX"  
 }

},

...

},

"responseCount": 30,

"totalCount": 103275

},

"type": "OK"

}

}

**Update Users Orders Endpoint**

URL : {{baseurl}}/myorders

Method : PUT

Data Parameters

Example Request

{

"data":{

"token":"usertoken2",

"request":{

"order":{

"id":259114,

"status":5

}

}

}

}

Success Response

Code: 200 OK

Content:

{

"data": {

"response": "ORDER UPDATED",

"status": "OK",

"type": "OK"

}

}

Notes

This endpoint allows for updating the status of an existing order in the database.

Any valid user token is required to access this endpoint. Users only can update their order status.

The status field represents the new status to be set for the order.

**MethodX / MethodY Payment Listener Endpoint**

URL : {{baseurl}}/methodx-payment

Method : POST

Example Request

{

"data":{

"status":"OK",

"transactionid":"03adad98511dbce17a44"

}

}

Success Response

Code: 200 OK

Content: "OK"

Error Response

Code: Relevant error status (e.g., 400 Bad Request, 500 Internal Server Error, etc.)

Content: "ERROR"

Notes

This endpoint acts as a listener for the MethodX payment API.

It expects a confirmation from the MethodX payment API, indicating whether the payment was successful or not.

If the payment is successful, the status "OK" is returned. If there's an error or issue with the payment, the status "ERROR" is returned.

**ShipperA Shipping Listener Endpoint**

URL : {{baseurl}}/companya-shipping

Method : POST

Example Request

{

"data":{

"status":"OK",

"order":"03adad98511dbce17a44"

}

}

Success Response

Code: 200 OK

Content: "OK"

Error Response

Code: Relevant error status (e.g., 400 Bad Request, 500 Internal Server Error, etc.)

Content: "ERROR"

Notes

This endpoint acts as a listener for the ShipperA shipping API.

It expects a notification from the ShipperA shipping API, indicating whether the order has been taken for shipping or not.

If the shipping process starts successfully, the status "OK" is returned. If there's an error or issue with the shipping process, the status "ERROR" is returned.

**ShipperA Delivery Confirmation Listener Endpoint**

URL : {{baseurl}}/companya-delivered

Method :POST

Example Request

{

"data":{

"status":"OK",

"order":"03adad98511dbce17a44"

}

}

Success Response

Code: 200 OK

Content: "OK"

Error Response

Code: Relevant error status (e.g., 400 Bad Request, 500 Internal Server Error, etc.)

Content: "ERROR"

Notes

This endpoint acts as a listener for the ShipperA delivery API.

It expects a notification from the ShipperA delivery API, indicating whether the order has been successfully delivered or not.

If the delivery is successful, the status "OK" is returned. If there's an error or issue with the delivery process, the status "ERROR" is returned.