**Parcel Delivery Application**

This application allows us to manage user accounts and support order creation and delivery. Three services used for this task.

Run this application just with the **docker-compose** command in the project base folder. **10** containers will be created for support all functionalities of application. **Docker Volumes** used for persist all data for logs, sql server, kafka topic states from one docker run to another.

**UserManagementService** - for managing account available in project. There are three roles.( **Admin**, **Courier**, **User**). "**/api**" endpoints used for external client. "/**api/ex**" routes used for communication between services. "**api/ex**" routes require "**ApiKey**" authentication. Swagger url ([Swagger UI](http://localhost:8003/swagger/index.html))

**OrderManagementService** - This service manage creation process of Order and give to user some functionalities like CancelOrder,ChangeOrderAddress and viewing own orders. Admin accounts can assign order to Courier and after that process will continue in the DeliveryManagementService. Swagger URL ( [Swagger UI](http://localhost:8001/swagger/index.html) )

**OrderManagementService –** This service works as BackgroundService and main purpose of this service provide reliable messaging between services. Use OutBox pattern for implementation.

**DeliveryManagementService** - This service manages the process of shipment from one status to another status. It stores all **history of shipment** and expose endpoints for retrieving order statuses and changing order status to next stage. Swagger URL ( [Swagger UI](http://localhost:8004/swagger/index.html) )

**Zookeper and Kafka** – Use these technologies for asynchronous messaging between services

**Kafdrop -** Used for monitoring kafka topics and messages. URL ( [Kafdrop: Broker List](http://localhost:9000/) )

**ApiGateWay –** Used for only one entry point for external clients like Front End. Accessing to “**api/ex**” resources prohibited by gateway. It also used for **RateLimiting.**

**SQL Service –** Used for persist data to database. SQL data stores in volumes for storing all data when container deletion.

**Health Check App –** Used for checking services states for production environment. URL ([Health Checks UI](http://localhost:8005/healthchecks-ui#/healthchecks))

**Execution Flow**

Used **ApiKey for “api/ex” routes is “**6b0f6be35a6e73d6f0e0f55a6fa1a01b**”**

1. You need sign in to project like admin. [Swagger UI](http://localhost:8003/swagger/index.html)
   1. Request to <http://localhost:8003/api/Account/signin>
   2. Body

{

"userName": "admin",

"password": "Admin1234"

}

* 1. Get JWT token from response

1. Create user account. [Swagger UI](http://localhost:8003/swagger/index.html)
   1. Request to <http://localhost:8003/api/Account/createAccount>
   2. Body

{

"firstName": "user",

"lastName": "user",

"email": "user@gmail.com",

"userName": "user1999",

"password": "User1999",

"role": "User"

}

* 1. Role should be one of them ( Admin , User , Courier )
  2. Only Admin account can create another account

1. Create courier account. [Swagger UI](http://localhost:8003/swagger/index.html)
   1. Request to <http://localhost:8003/api/Account/createAccount>
   2. Body

{

"firstName": "courier",

"lastName": "courier",

"email": "courier@gmail.com",

"userName": "courier1999",

"password": "Courier1999",

"role": "Courier"

}

* 1. Only Admin account can create another account

1. SignIn like a user and add that JWT token to swagger authorization part and Create an Order. [Swagger UI](http://localhost:8001/swagger/index.html)
   1. Request to <http://localhost:8001/api/OrderRegistry/createOrder>
   2. Body

{

"billingAddress": "Server Bedelbeyli 78",

"shippingAddress": "Sulh kucesi 45",

"orderItems": [

{

"productNumber": "1551DFHR",

"quantity": 5,

"price": 2

},

{

"productNumber": "6236GHDR",

"quantity": 7,

"price": 4

}

]

}

* 1. OrderNumber will be create automatically
  2. UserId will be get from user credential
  3. InvoiceAmount will be calculated and saved
  4. Only **User** and **Admin** can create order
  5. Order will be created in **Pending** status
  6. Response body

{

"data": {

"id": 1,

"orderNumber": "20230424125734-D19BE47D",

"customerFirstName": "user",

"customerLastName": "user",

"billingAddress": "Server Bedelbeyli 78",

"shippingAddress": "Sulh kucesi 45",

"invoiceAmount": 38,

"orderStatus": "Pending",

"assignedBy": null,

"orderItems": [

{

"productNumber": "1551DFHR",

"quantity": 5,

"price": 2

},

{

"productNumber": "6236GHDR",

"quantity": 7,

"price": 4

}

]

},

"succeeded": true,

"errorMessage": null,

"validationErrors": {}

}

1. User can update Billing and Shipping addresses before order will be assigned to courier. [Swagger UI](http://localhost:8001/swagger/index.html)
   1. Request to <http://localhost:8001/api/OrderRegistry/changeOrderAddresses>
   2. Body

{

"orderId": 1,

"billingAddress": "Tbilisi street 23",

"shippingAddress": null

}

* 1. Only “BillingAddress” will be updated in this situation
  2. Order should be in “Pending” status
  3. Only **owner** of order and **Admin** can update the addresses of order
  4. Success response

{

"succeeded": true,

"errorMessage": null,

"validationErrors": {}

}

1. Assign order to courier. [Swagger UI](http://localhost:8001/swagger/index.html)
   1. Request to <http://localhost:8001/api/OrderRegistry/assignToCurier>
   2. Body

{

"orderId": 1,

"courierId": 3

}

* 1. Order should be in “Pending” status
  2. After execution status of Order changed to “InProgress” and OrderMessage will be sent to DeliveryManagementService with Kafka
  3. Response

{

"succeeded": true,

"errorMessage": null,

"validationErrors": {}

}

d. Only user in **Admin** can assign order to courier

1. After order will be created in DeliveryyManagementService user can Cancel order. [Swagger UI](http://localhost:8001/swagger/index.html)
   1. Request to <http://localhost:8001/api/OrderRegistry/cancelOrder>
   2. Body

{

"orderId": 1

}

* 1. Order should be in one of these statuses for Canceling ( Pending, InProgress, Shipped )
  2. Cancelation message will be sent to DeliveryManagementService with kafka after checking status
  3. After cancelation OrderStatus will be changed to “Canceled” status
  4. Only Admin and Owner of order can cancel it

1. After order assigned to any courier it start processing in the DeliveryManagementService. Courier should change order to next stage. [Swagger UI](http://localhost:8004/swagger/index.html)
   1. Request to <http://localhost:8004/api/DeliveryHistory/changeOrderStatus>
   2. Body

{

"id": 1,

"orderStatus": "Shipped"

}

* 1. After each stage “OrderStatus” synchronize with OrderManagementService over Kafka
  2. OrderStatus checked each time only appropriate transitions succeeded
     1. Pending => InProcess => Shipped => Delivered
  3. Only Admin and assigned Courier can change status to next stage

1. Accounts can get orders. [Swagger UI](http://localhost:8004/swagger/index.html)
   1. Request to <http://localhost:8004/api/DeliveryHistory/getOrders>
   2. Only Admins and Courier can access to this endpoint. Couriers will view only order which assigned to them
2. Accounts can get detailed delivery history. [Swagger UI](http://localhost:8004/swagger/index.html)
   1. Request to [http://localhost:8004/api/DeliveryHistory/getOrderDeliveryHistory?OrderNumber={OrderNumber}](http://localhost:8004/api/DeliveryHistory/getOrderDeliveryHistory?OrderNumber=%7bOrderNumber%7d)
   2. Admin role can view all order histories
   3. Couriers can view only order which assigned to them
   4. Users can view orders which they created