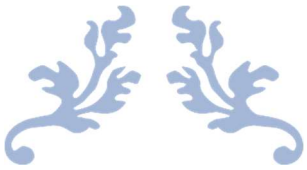


# DNIPRO

bookstore

JUNE 4, 2020

CRYPTOAWESOMETEAM
Kyiv

# Contents

## Intro

This application was created for the convenience of ordering books in the online bookstore "Dnipro". Shopping in our store is fast and convenient. Each person could register in this application. Please note that you grant the right to use your personal information:

- Name
- Age
- E-mail
- Address

Our application is easy in use.
You could find a specific publication, using the search, also you are able to view the availability of books (and their quantity) that are present in the store. There is an ability to add them to your cart, as well as remove, then complete order with books in your cart and get them.

We value our customers, so the online store can find a book for every taste.
Have any questions? Just contact with out support team, we always try to do our best!

Our project (to explore source code, follow linked repos): https://github.com/MlikeW?tab=projects

# Microservices

## Description

A description and the purpose of each service will be provided in this section.

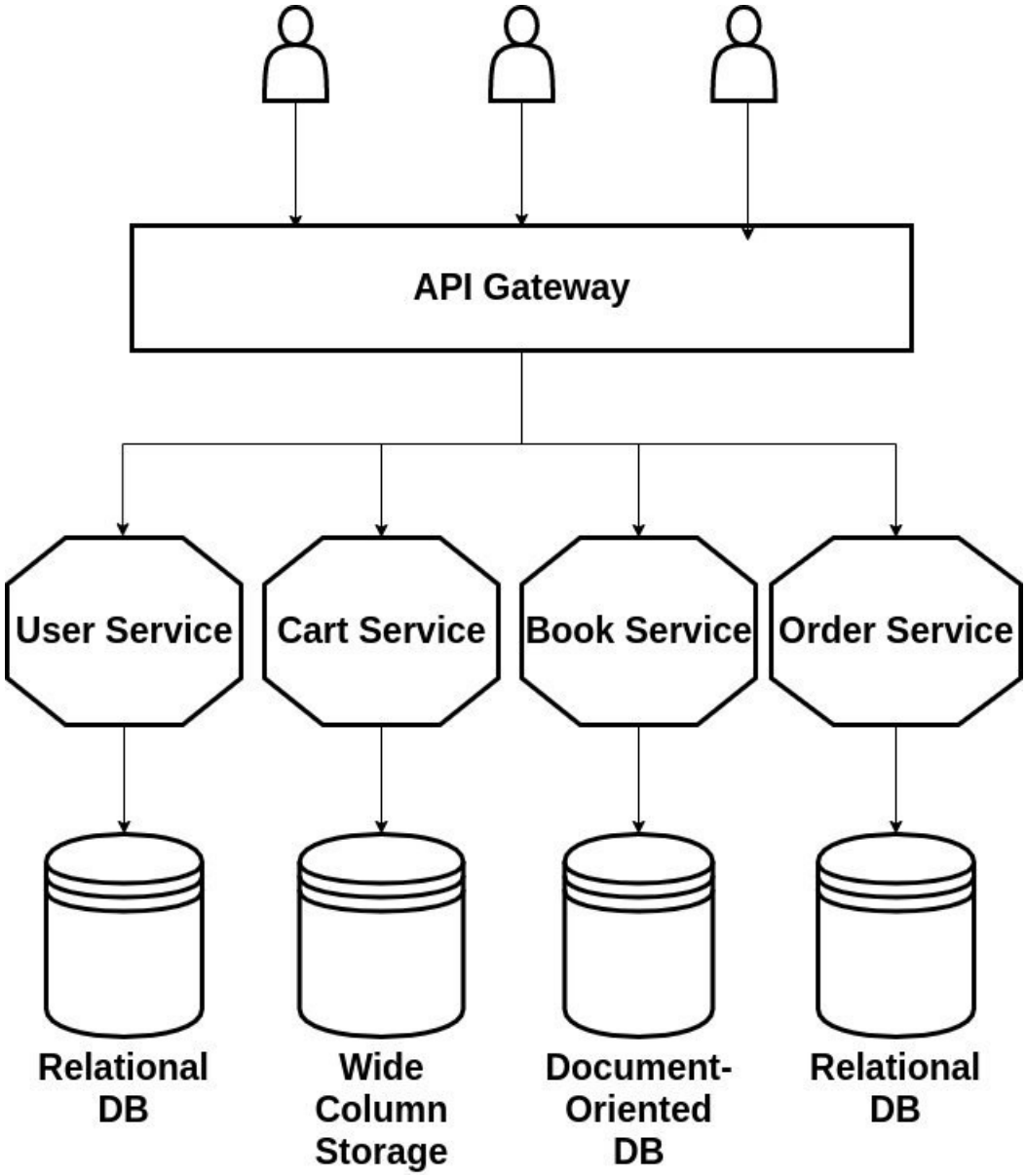
API Gateway – accepts all user requests and, at the beginning of the endpoint, distributes their execution to the corresponding microservices.

User Service – stores all the users of this bookstore, all CRUD operations are implemented.

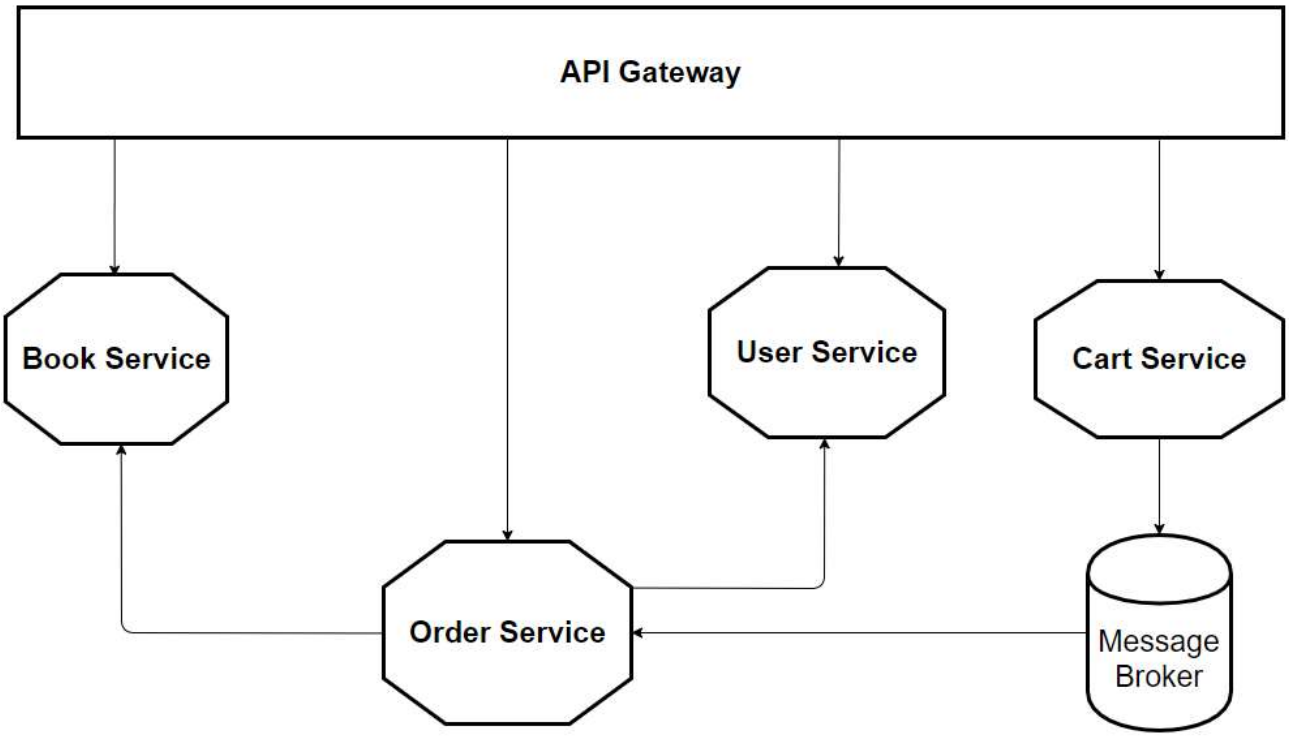
Book Service – stores all available books, their info and quantity, all CRUD operations are implemented.

Cart Service – stores current state of user cart (all books which user want to buy in one order), all CRUD operations are implemented.

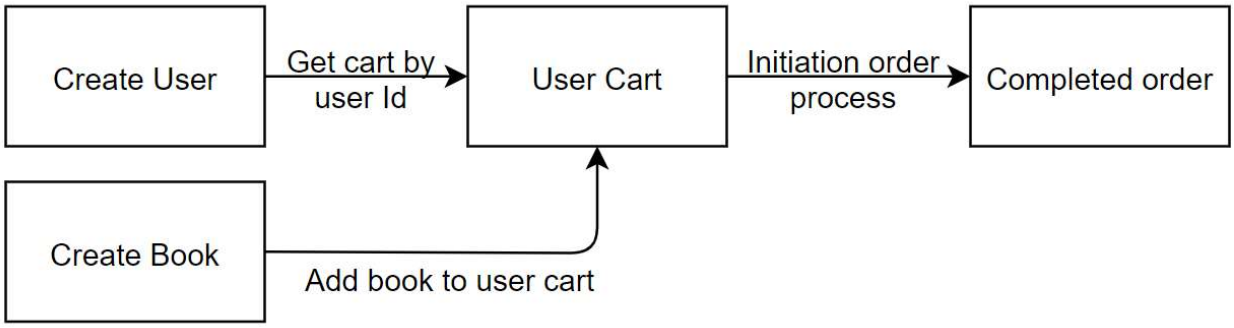
Order Service – stores all completed orders in bookstore, also there is ability to get all orders by existing user.



## Interactions diagram




## Main positive E2E Flow



# API Manual

| Endpoint | Comment |
|---|---|
| **/orders** | Collection of all processing orders in the store. |
| **/orders/{orderId}** | A specific order. |
| **/users** | Collection of all registered users in the store. |
| **/users/{userId}** | A specific user. |
| **/books** | Collection of all books in the store. |
| **/books/{bookId}** | A specific book. |
| **/cart/{userId}** | Collection of books in current user cart. |
| **/cart/{userId}/{bookId}** | A specific book in current user cart. |

## /orders

Method: **GET**

This method allows getting all\specific current available in store orders.

Request parameters:

| userId | page |
| --- | --- |
| A specific user Id. | A specific page number. |

Examples:

Request: **/orders**

Response: code – 200 OK


```
{
  "content": [
    {
      "id": "219856fd-3ddb-43aa-8392-37f0278a808a",
      "userId": "8",
      "addressLine": "user8Address",
      "status": "VERIFIED",
      "orderDetails": {
        "orderItems": [
          {
            "itemId": "3838",
            "quantity": 6
          }
        ]
      },
      "placedAt": "2020-05-24T10:22:13.161252"
    },
    {
      "id": "195856fd-3bab-43ab-6899-45f0278a688a",
      "userId": "1",
      "addressLine": "user1Address",
      "status": "VERIFIED",
      "orderDetails": {
        "orderItems": [
          {
            "itemId": "666",
            "quantity": 2
          }
        ]
      },
      "placedAt": "2020-05-24T10:25:13.172265"
    }
  ],
  "pageable": {
    "sort": {
      "sorted": false,
      "unsorted": true,
      "empty": true
    },
    "pageNumber": 0,
    "pageSize": 20,
    "offset": 0,
    "unpaged": false,
    "paged": true
  },
  "totalElements": 6,
  "last": true,
  "totalPages": 1,
  "sort": {
    "sorted": false,
    "unsorted": true,
    "empty": true
  },
  "numberOfElements": 2,
  "first": true,
  "size": 20,
  "number": 0,
  "empty": false
}
```


Request: **/orders?userId=8&page=1**

Response: code – 200 OK


```
{
  "content": [
    {
      "id": "219856fd-3ddb-43aa-8392-37f0278a808a",
      "userId": "8",
      "addressLine": "user8Address",
      "status": "VERIFIED",
      "orderDetails": {
        "orderItems": [
          {
            "itemId": "3838",
            "quantity": 6
          },
          {
            "itemId": "335",
            "quantity": 2
          }
        ]
      },
      "placedAt": "2020-05-24T10:22:13.161252"
    }
  ],
  "pageable": {
    "sort": {
      "sorted": false,
      "unsorted": true,
      "empty": true
    },
    "pageNumber": 0,
    "pageSize": 20,
    "offset": 0,
    "unpaged": false,
    "paged": true
  },
  "totalElements": 6,
  "last": true,
  "totalPages": 1,
  "sort": {
    "sorted": false,
    "unsorted": true,
    "empty": true
  },
  "numberOfElements": 1,
  "first": true,
  "size": 20,
  "number": 0,
  "empty": false
}
```

## /orders/{orderId}

Method: **GET**

This method allows getting a order from store.

Request parameters: none

Response: code – 200 OK


```
{
  "id": "386808ce-6632-42f5-ac35-3c8afbcd508f",
  "userId": "8",
  "addressLine": "Kyiv",
  "status": "VERIFIED",
  "orderDetails": {
    "orderItems": [
      {
        "itemId": "5ed6c6b3a1d9d451686d4664",
        "quantity": 1
      }
    ]
  },
  "placedAt": "2020-06-02T21:42:23.926053"
}
```

## /users

Method: **GET**
This method allows getting all\specific user from store.
Request parameters:

| Name |
| --- |
| A specific user Name. |

Response: code – 200 OK


```
{
  "ID": 8,
  "CreatedAt": "2020-06-02T21:35:59.766683Z",
  "UpdatedAt": "2020-06-02T21:35:59.766683Z",
  "DeletedAt": null,
  "Name": "Anakin Skywalker",
  "Age": 32,
  "Email": "mayThe4thBeWithYou@gmail.com",
  "Address": "Kyiv"
}
```


Method: **POST**
This method allows creating user in the store.
Request:


```
{
  "Name": "Anakin Skywalker",
  "Age": 32,
  "Email": "mayThe4thBeWithYou@gmail.com",
  "Address": "Kyiv"
}
```


Response: code – 200 OK


```
{
  "ID": 8,
  "CreatedAt": "2020-06-02T21:35:59.766683Z",
  "UpdatedAt": "2020-06-02T21:35:59.766683Z",
  "DeletedAt": null,
  "Name": "Anakin Skywalker",
  "Age": 32,
  "Email": "mayThe4thBeWithYou@gmail.com",
  "Address": "Kyiv"
}
```

## /users/{userId}

Method: **GET**

This method allows getting a specific user entity from store (ex. you might need user Id in creating cart).

Request parameters: none

Response: code – 200 OK


```
{
  "ID": 8,
  "CreatedAt": "2020-06-02T21:35:59.766683Z",
  "UpdatedAt": "2020-06-02T21:35:59.766683Z",
  "DeletedAt": null,
  "Name": "Anakin Skywalker",
  "Age": 32,
  "Email": "mayThe4thBeWithYou@gmail.com",
  "Address": "Kyiv"
}
```


Method: **DELETE**

This method allows deleting user from the store.

Request parameters: none

Response: code – 204 NoContent

## /books

Method: **POST**

This method allows adding new book to the store.

Request:


```
{
    "title": "Awesome Book I",
    "description": "book about awesomenest",
    "price": "15.21",
    "quantity": 13,
    "authors": [
        {
            "firstName": "Arthur",
            "lastName": "Pendragon"
        }
    ],
    "genres": [
        "life",
        "psychology"
    ],
    "tags": [
        "modern",
        "bestseller"
    ]
}
```


Response: code – 200 OK


```
{
    "id": "5ed6c6b3a1d9d451686d4664",
    "title": "Awesome Book I",
    "description": "book about awesomenest",
    "price": 15.21,
    "quantity": 13,
    "authors": [
        {
            "firstName": "Arthur",
            "lastName": "Pendragon"
        }
    ],
    "genres": [
        "life",
        "psychology"
    ],
    "tags": [
        "modern",
        "bestseller"
    ]
}
```

## /books/{bookId}

Method: **GET**

This method allows getting a specific book entity from store.

Request parameters: none

Response: code – 200 OK


```
{
    "id": "5ed6c6b3a1d9d451686d4664",
    "title": "Awesome Book I",
    "description": "book about awesomenest",
    "price": 15.21,
    "quantity": 13,
    "authors": [
        {
            "firstName": "Arthur",
            "lastName": "Pendragon"
        }
    ],
    "genres": [
        "life",
        "psychology"
    ],
    "tags": [
        "modern",
        "bestseller"
    ]
}
```

## /cart/{userId}

Method: **GET**
This method allows getting list of books in current user cart.
Request parameters: none
Response: code – 200 OK


```
{
  "userId": 0,
  "items": [
    {
      "itemId": "string",
      "number": 0
    }
  ]
}
```


Method: **POST**
Complete order (send to store orders) from current user card.
Request parameters: none
Response: code – 200 OK

Method: **DELETE**
This method allows deleting all books from current user cart.
Request parameters: none
Response: code – 200 OK

## /cart/{userId}/{bookId}

Method: **GET**
This method allows getting an entity of book and its quantity in current user cart.
Request parameters: none
Response: code – 200 OK


```
{
  "itemId": "string",
  "number": 0
}
```


Method: **PUT**
Add book (with different quantity) to current user cart.
Request parameters:

| number |
| --- |
| Number of added books (1 by default). |

Response: code – 200 OK

Method: **POST**
Change quantity of the book in current user cart.
Request:


```
{
  "operation": "INCREMENT",
  "number": 0
}
```


Response: code – 200 OK


```
{
  "itemId": "string",
  "number": 0
}
```


Method: **DELETE**
This method allows deleting book from current user cart.
Request parameters: none
Response: code – 200 OK


```
true
```

# Use Cases

In this section will be presented scripting of Gherkin scenarios of usage of the application (by default we will add 5 books to the store).

```
Feature: Flow
        As a user of Dnipro store
        I want to be able to create account and make an order

Background:
        Given I create user in store
                | Name          | Age | Email                | Address |
                | Lord Voldemort | 45  | avadakedavra@gmail.com | London  |
        And I add 'Pandemia' book to store

Scenario: Easy positive flow
        When I add 'Pandemia' book to Lord Voldemort user cart
        And I place order from Lord Voldemort user cart
        Then I check Lord Voldemort user order present in store

Scenario: Delete user before process order
        When I add 'Pandemia' book to Lord Voldemort user cart
        And I delete Lord Voldemort user
        And I place order from Lord Voldemort user cart
        Then I check Lord Voldemort user order absent in store

Scenario: Delete book before process order
        When I add 'Pandemia' book to Lord Voldemort user cart
        And I delete 'Pandemia' book from store
        And I place order from Lord Voldemort user cart
        Then I check Lord Voldemort user order absent in store

Scenario: Order more books then available in store
        When I add 'Pandemia' book to Lord Voldemort user cart 6 times
        And I place order from Lord Voldemort user cart
        Then I check Lord Voldemort user order absent in store

Scenario: Delete from store one of two book in cart
        When I add 'Pandemia' book to Lord Voldemort user cart
        And I add 'World War' book to store
        And I add 'World War' book to Lord Voldemort user cart
        And I delete 'Pandemia' book from store
        And I place order from Lord Voldemort user cart
        Then I check Lord Voldemort user order absent in store

Scenario: Check absence of deleted book from cart in order
        When I add 'Pandemia' book to Lord Voldemort user cart
        And I add 'World War' book to store
        And I add 'World War' book to Lord Voldemort user cart
        And I delete 'Pandemia' book from Lord Voldemort user cart
        And I place order from Lord Voldemort user cart
        Then I check Lord Voldemort user order present in store
        And Lord Voldemort user order contain books
                | Books     |
                | World War |

Scenario Outline: Check correctness of different quantity of books in order
        When I add 'Pandemia' book to Lord Voldemort user cart <Quantity 1> times
        And I add 'World War' book to store
        And I add 'World War' book to Lord Voldemort user cart <Quantity 2> times
        And I place order from Lord Voldemort user cart
        Then I check Lord Voldemort user order present in store
        And Lord Voldemort user order contain books
                | Books     | Quantity     |
                | Pandemia  | <Quantity 1> |
                | World War | <Quantity 2> |

        Examples:
                | Quantity 1 | Quantity 2 |
                | 2          | 3          |
                | 1          | 5          |
```