ONLINE GAME STORE

Requirements – Epic 3 – Payment methods

Extend the functionality of the Game Store by adding the ability to buy a game.

The system should support the following features:

- Add game to cart.
- Get games from cart.
- Get orders.
- Get payment methods.
- Perform payment with the selected method.

Entities

Order

- Id: Guid, required, unique
- Date: DateTime, optional
- CustomerId: Guid, required
- Status: Enum, required

OrderGame

- OrderId: Guid, required
- ProductId: Guid, required
- Price: Double, required
- Quantity: Int, required
- Discount: Int, optional

Product-Order combinations are unique.

Additional Requirements

Payment

Create a single endpoint for payment independently from the selected method.

If the payment is processed successfully then the order must be marked as paid otherwise cancelled

Payment Methods

Allowed payment methods: “Bank”, “IBox terminal”, “Visa”.
Only one method can be applied to an order to get paid.
Each payment method contains a little picture (open-source pictures), title, short description.

Customer

Since we don’t have users yet, for Customer Id use a stub value.

Bank payment date of validity

The date of validity – how long the invoice is valid. The value should be configurable by application settings.

Order limitations

User cannot order more games than available in the stock.

Order statuses

Any order has the next possible statuses:

- Open – games are in the cart.
- Checkout – payment is started.
- Paid – payment is performed successfully.
- Cancelled – payment is performed with errors.

Endpoint specifications:

US1E3

Add game in the cart endpoint
Url: /games/{key}/buy
Type: POST
Limitation: if the endpoint is called for the game which is already in the cart, then just increment the quantity
Response: Success status code

Delete game from cart endpoint
Url: /orders/cart/{key}
Type: DELETE
Response: Success status code

US2E3

Get paid and cancelled orders endpoint.

Url: /orders

Type: GET

Response example:


```
[
  {
    "id": "5d8af81a-c146-4588-93bf-0e5c7e9e4a9e",
    "customerId": "5aa1c97e-e6b3-497c-8e00-270e96aa0b63",
    "date": "2023-11-20T11:03:26.0572863+02:00"
  },
  {
    "id": "bec0ffb1-fb80-47ff-8720-2f9a544a55a2",
    "customerId": "bfa70dfa-6b18-4f3f-b882-14af597396d4",
    "date": "2023-11-18T11:03:26.0575052+02:00"
  }
]
```


Get order by id endpoint.

Url: /orders/{id}

Type: GET

Response example:


```
{
  "id": "5d8af81a-c146-4588-93bf-0e5c7e9e4a9e",
  "customerId": "5aa1c97e-e6b3-497c-8e00-270e96aa0b63",
  "date": "2023-11-20T11:03:26.0572863+02:00"
}
```


US3E3

Get order details endpoint.

Url: /orders/{id}/details

Type: GET

Response Example:


```
[
  {
    "productId": "8ea595c2-765b-4190-b3da-da00540b2202",
    "price": 100,
    "quantity": 3,
    "discount": 0
  },
  {
    "productId": "c3a73173-fe0f-4771-a7eb-f2cd458d8303",
    "price": 10,
    "quantity": 5,
    "discount": 0
  },
  {
    "productId": "923a8bd8-b256-44f4-b972-a0d640d56ef4",
    "price": 100,
    "quantity": 1,
    "discount": 10
  }
]
```

US4E3

Get cart endpoint

Url: /orders/cart

Type: GET

Hint: Cart is an order in the status Open, if such an order is not present in the database this should be created automatically during adding the first game to the cart. As a result, if the last game is deleted from the cart then the open order should be deleted automatically.

Response example:


```
[
  {
    "productId": "f6f698ee-41df-4594-b90c-3862e7d2cbee",
    "price": 30,
    "quantity": 1,
    "discount": 0
  },
  {
    "productId": "75396383-c1fa-4cbd-81c9-c874ae3a3e67",
    "price": 100,
    "quantity": 1,
    "discount": 20
  }
]
```


US5E3

Get payment methods endpoint

Url: /orders/payment-methods

Type: GET

Response Example:


```
{
  "paymentMethods": [
    {
      "imageUrl": "image link1",
      "title": "Bank",
      "description": "Some text 1"
    },
    {
      "imageUrl": "image link2",
      "title": "IBox terminal",
      "description": "Some text 2"
    },
    {
      "imageUrl": "image link3",
      "title": "Visa",
      "description": "Some text 3"
    }
  ]
}
```

US6E3

''Bank'' payment
Url: /orders/payment
Type: POST
Request: {"method":"Bank"}
Flow: the system should return generated invoice file for download:
File type: PDF
The file content:

- User ID
- Order ID
- Creation date
- The date of validity – how long is the invoice is valid.
- Sum

US7E3

“IBox terminal” payment
Url: /orders/payment
Type: POST
Request: {"method":"IBox terminal"}
Integration: Integration with payment microservice is required
Flow: The system should handle requests with an IBox payment.
Response: should contain a user Id, invoice number (order ID), and sum.
Response Example:

```
{
  "userId": "24967e32-dec1-47b5-8ca6-478afa84c2be",
  "orderId": "7dce8347-4181-4316-9210-302361340975",
  “paymentDate”: “2023-11-18T11:03:26.0575052+02:00”,
  "sum": 100
}
```

US8E3

“Visa” payment
Url: /orders/payment
Type: POST
Request:

```
{
  "method": "Visa",
  "model": {
    "holder": "Vitalii",
    "cardNumber": "123321122344231",
    "monthExpire": 10,
    "yearExpire": 2030,
    "cvv2": 111
  }
}
```

Integration: Integration with payment microservice is required
Flow: The system should handle requests with card holder’s name, card number, Date of expiry (month and year), CVV2/CVC2
Response: Success status code.

NFR

NFR1E3

Microservice - an additional project that must be run locally and integrated with the game store api for Visa and IBox payments.
You can investigate possible requests and results of microservice by calling the swagger endpoint.

NFR2E3

Implement full tolerance payment acceptation.
As the microservice that accepts iBox and card payments can reject up to 10 % of the transactions.
A response validation must be implemented and an additional request must be sent in case of failure, the solution must be able to work with all responses of the Microservice.