

# Nowakowski Sprawozdanie Lab6

## Spis treści

Część z EF: .....	1
Testowanie ProductService EF: .....	2
1. Dodanie produktu: .....	2
2. Aktywacja i dezaktywacja produktu: .....	3
3. Pobieranie listy produktów: .....	5
4. Pobieranie produktu po ID: .....	7
Testowanie Basket Service EF .....	8
1. Dodanie produktu do koszyka .....	8
2. Zmiana liczby produktów .....	9
3. Usunięcie produktu z koszyka .....	9
4. Generowanie zamówienia: .....	10
5. Opłacenie zamówienia: .....	11
Część z DB: .....	11
Testowanie ProductService DB: .....	13
1. Dodanie produktu: .....	13
2. Aktywacja i dezaktywacja produktu: .....	14
3. Pobieranie listy produktów: .....	15
4. Pobieranie produktu po ID: .....	17
Testowanie Basket Service DB .....	17
1. Dodanie produktu do koszyka .....	17
2. Zmiana liczby produktów .....	19
3. Usunięcie produktu z koszyka .....	19
4. Generowanie zamówienia: .....	20
5. Opłacenie zamówienia: .....	21
PODSUMOWANIE/WNIOSKI .....	22

## Część z EF:

Utworzyłem 2 interfejsy Product i Basket:

```

public interface IProductService
{
    Odwołania: 2
    IEnumerable<ProductResponseDTO> GetProducts(ProductFilterRequestDTO filter);
    Odwołania: 2
    ProductResponseDTO GetProductById(int id);
    Odwołania: 2
    ProductResponseDTO AddProduct(ProductRequestDTO productDTO);
    Odwołania: 2
    void DeactivateProduct(int productId);
    Odwołania: 2
    void ActivateProduct(int productId);
}

```

```

void AddToBasket(BasketItemRequestDTO basket);
Odwołania: 2
void ChangeNumberOfProducts(int id, int numberOfProducts);
Odwołania: 2
void RemoveProductFromBasket(int id);
Odwołania: 2
OrderResponseDTO GenerateOrder(int userId);
Odwołania: 2
void Pay(int userId, double value);

```

## Testowanie ProductService EF:

### 1. Dodanie produktu:

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** /api/Product
- Parameters:** No parameters
- Request body:** application/json
- Request body content:**

```

{
  "name": "mleko",
  "price": 4,
  "image": "link",
  "isActive": true,
  "groupId": 2,
  "basketPositions": null
}

```
- Buttons:** Execute, Clear, Cancel, Reset

Request URL  
http://localhost:5037/api/Product

Server response

Code	Details
201 Undocumented	<p>Response body</p> <pre>{   "id": 1007,   "name": "mleko",   "price": 4,   "image": "link",   "isActive": true,   "groupID": 2,   "groupName": "link",   "basketPositions": null }</pre> <p>Response headers</p> <pre>content-type: application/json; charset=utf-8 date: Wed, 27 Mar 2024 11:00:09 GMT location: http://localhost:5037/api/Product/1007 server: Kestrel transfer-encoding: chunked</pre>

Responses

Code	Description	Links
201	Created	

W bazie:

ID	Name	Price	Image	IsActive	GroupID
3	string	888	string	True	1
4	stri767676ng	88	stri767676ng	False	1
5	st66ng	888	st66ng	True	1
1002	str344354ing	99	str344354ing	True	1
1003	string	889	string	True	1
1004	stri89ng	98	stri89ng	True	1
1005	str6767ing	8	str6767ing	True	1
1006	mleko	3	mleko	True	2
1007	mleko	4	link	True	2
NULL	NULL	NULL	NULL	NULL	NULL

## 2. Aktywacja i dezaktywacja produktu:

PUT /api/Product/{id}/Deactivate

Parameters

Name Description

id \* required  
integer(\$int32)  
(path) 1007

Execute Clear

Responses

curl -X 'PUT' \

```
http://localhost:5037/api/Product/1007/Deactivate' \
```

-H 'accept: \*/\*'

Request URL

http://localhost:5037/api/Product/1007/Deactivate

Server response

Code	Details
204 Undocumented	<p>Response headers</p> <pre>date: Wed, 27 Mar 2024 11:02:25 GMT server: Kestrel</pre>

Responses

Code	Description	Links
200	Success	No links

	1006	mleko	3	mleko	True	2
	1007	mleko	4	link	False	2
	NULL	NULL	NULL	NULL	NULL	NULL

Aktywacja:

responses

Curl

```
curl -X 'PUT' \
'http://localhost:5037/api/Product/1007/Activate' \
-H 'accept: */*'
```

Request URL

```
http://localhost:5037/api/Product/1007/Activate
```

Server response

Code	Details
204	Response headers
Undocumented	<pre>date: Wed, 27 Mar 2024 11:03:03 GMT server: Kestrel</pre>
Responses	
Code	Description
200	Success

	1006	storing	3	storing	True	1
	1007	mleko	4	link	True	2
	NULL	NULL	NULL	NULL	NULL	NULL

### 3. Pobieranie listy produktów:

Name  
string  
(query)

GroupName  
string  
(query)

GroupID  
Integer(\$int32)  
(query)

IncludeInactive  
boolean  
(query)

SortBy  
string  
(query)

SortAsc  
boolean  
(query)

Execute

Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:5037/api/Product?Name=mleko&IncludeInactive=true&SortBy=Name&SortAsc=true' \
  -H 'accept: text/plain'
```

Request URL

```
http://localhost:5037/api/Product?Name=mleko&IncludeInactive=true&SortBy=Name&SortAsc=true
```

Server response

Code

Details

200

Response body

```
{
  {
    "id": 1000,
    "name": "mleko",
    "price": 0,
    "image": "Mabial",
    "inactive": true,
    "groupID": 1,
    "groupName": "mleko",
    "basketPositions": null
  },
  {
    "id": 1007,
    "name": "mleko",
    "price": 0,
    "image": "Mabial",
    "inactive": true,
    "groupID": 1,
    "groupName": "link",
    "basketPositions": null
  }
}
```

Download

Name	Description
Name string (query)	<input type="text" value="Name"/>
GroupName string (query)	<input type="text" value="GroupName"/>
GroupID integer(\$int32) (query)	<input type="text" value="1"/>
IncludeInactive boolean (query)	<input type="text" value="true"/>
SortBy string (query)	<input type="text" value="Name"/>
SortAsc boolean (query)	<input type="text" value="true"/>

Execute

Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:5837/api/Product?groupID=1&includeInactive=true&sortBy=Name&sortAsc=true' \
  -H 'accept: text/plain'
```

Request URL

```
http://localhost:5837/api/Product?groupID=1&includeInactive=true&sortBy=Name&sortAsc=true
```

Server response

Code

Details

200

Response body

```
{
  "id": 0,
  "name": "string",
  "price": 0.0,
  "image": "None",
  "inactive": true,
  "groupID": 1,
  "groupName": "string",
  "basketPositions": null
},
{
  "id": 0,
  "name": "str1767676ng",
  "price": 0.0,
  "image": "None",
  "inactive": false,
  "groupID": 1,
  "groupName": "str1767676ng",
  "basketPositions": null
},
{
  "id": 0,
  "name": "st66ng",
  "price": 0.0,
  "image": "None",
  "inactive": true
}
```

Name	Description
Name string (query)	<input type="text" value="Name"/>
GroupName string (query)	<input type="text" value="GroupName"/>
GroupID integer(\$int32) (query)	<input type="text" value="1"/>
IncludeInactive boolean (query)	<input type="text" value="false"/>
SortBy string (query)	<input type="text" value="Name"/>
SortAsc boolean (query)	<input type="text" value="true"/>

[Execute](#) [Clear](#)

**Responses**

Curl

```
curl -X 'GET' \
  "http://localhost:5037/api/Product?GroupID=1&IncludeInactive=false&SortBy=Name&SortAsc=true" \
  -H 'accept: text/plain'
```

Request URL

```
http://localhost:5037/api/Product?GroupID=1&IncludeInactive=false&SortBy=Name&SortAsc=true
```

Server response

Code Details

200 Response body

```
{
  "id": 3,
  "name": "string",
  "price": 88,
  "image": "Nowa",
  "isActive": true,
  "groupID": 1,
  "groupName": "string",
  "basketPositions": null
},
{
  "id": 5,
  "name": "st66ng",
  "price": 889,
  "image": "Nowa",
  "isActive": true,
  "groupID": 1,
  "groupName": "st66ng",
  "basketPositions": null
},
{
  "id": 1007,
  "name": "str344354ing",
  "price": 89,
  "image": "Nowa",
  "isActive": true
}
```

#### 4. Pobieranie produktu po ID:

GET /api/Product/{id}

Parameters

Name Description

id \* required  
integer(\$int32)  
(path)

[Execute](#) [Clear](#)

**Responses**

Curl

```
curl -X 'GET' \
  "http://localhost:5037/api/Product/1007" \
  -H 'accept: */*'
```

Request URL

```
http://localhost:5037/api/Product/1007
```

Server response

Code Details

200 Response body

```
{
  "id": 1007,
  "name": "kiska",
  "price": 11,
  "image": "",
  "isActive": true,
  "groupID": 1,
  "groupName": "111",
  "basketPositions": null
}
```

Response headers

```
Content-Type: application/json; charset=utf-8
date: Wed, 27 Mar 2024 11:09:31 GMT
Server: Kestrel
transfer-encoding: chunked
```

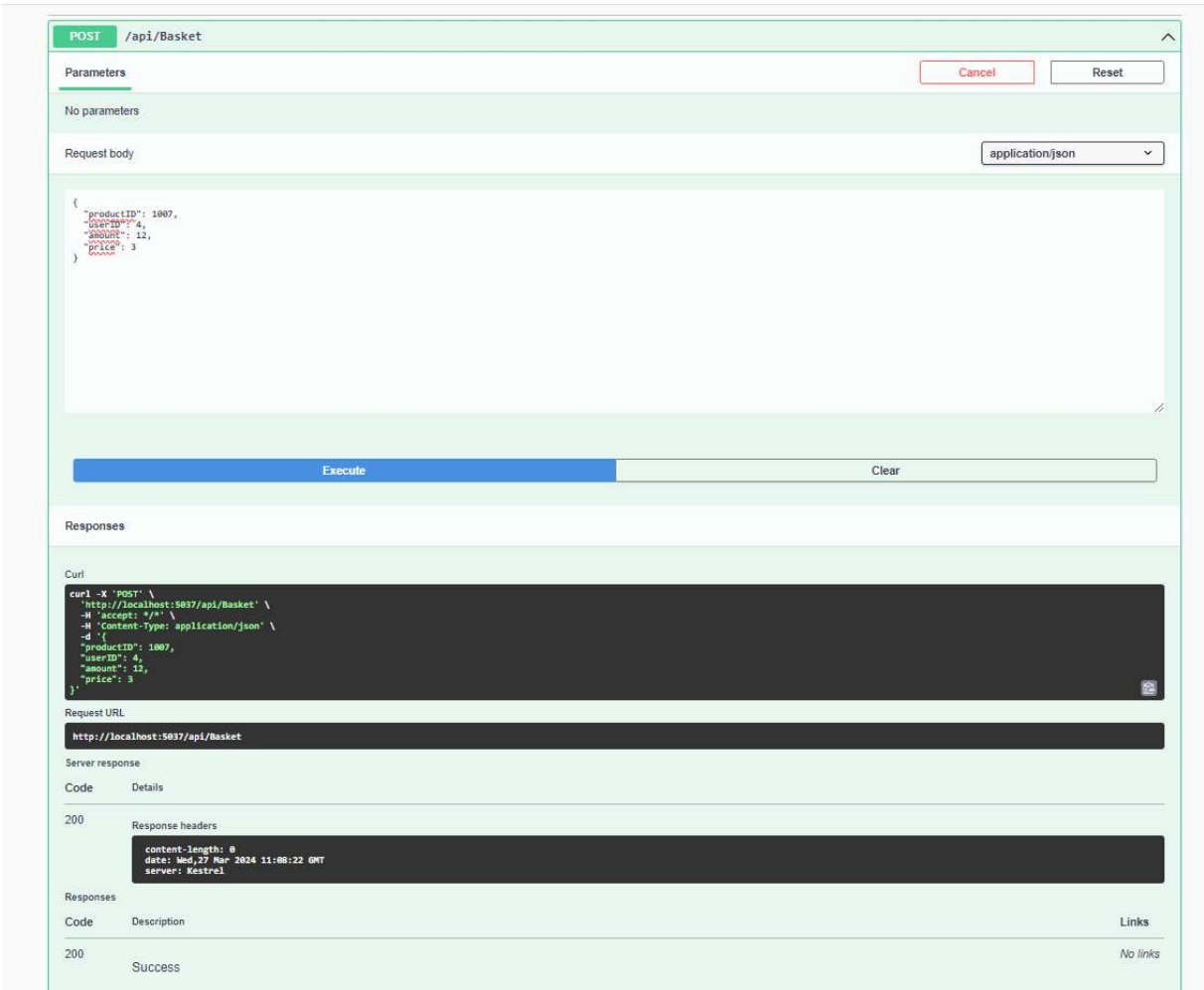
Responses

Code Description

[Links](#)

# Testowanie Basket Service EF

## 1. Dodanie produktu do koszyka



W bazie:

4	1007	4	12	3	NULL
NULL	NULL	NULL	NULL	NULL	NULL



## 2. Zmiana liczby produktów

**PUT** /api/Basket/{id}

**Parameters**

Name	Description
id * required	
integer(\$int32)	
(path)	
numberOfProducts	
integer(\$int32)	
(query)	

**Execute** **Clear**

**Responses**

**Curl**

```
curl -X 'PUT' \
  'http://localhost:5037/api/Basket/4?numberOfProducts=150' \
  -H 'accept: */*'
```

**Request URL**

```
http://localhost:5037/api/Basket/4?numberOfProducts=150
```

**Server response**

Code	Details
200	<b>Response headers</b> content-length: 0 date: Wed, 27 Mar 2024 11:09:51 GMT server: Kestrel

**Responses**

Code	Description	Links
200	Success	No links

4	1007	4	150	3	NULL
NULL	NULL	NULL	NULL	NULL	NULL

## 3. Usunięcie produktu z koszyka

**DELETE** /api/Basket/{id}

**Parameters**

Name	Description
id * required	
integer(\$int32)	
(path)	

**Execute** **Clear**

**Responses**

**Curl**

```
curl -X 'DELETE' \
  'http://localhost:5037/api/Basket/3' \
  -H 'accept: */*'
```

**Request URL**

```
http://localhost:5037/api/Basket/3
```

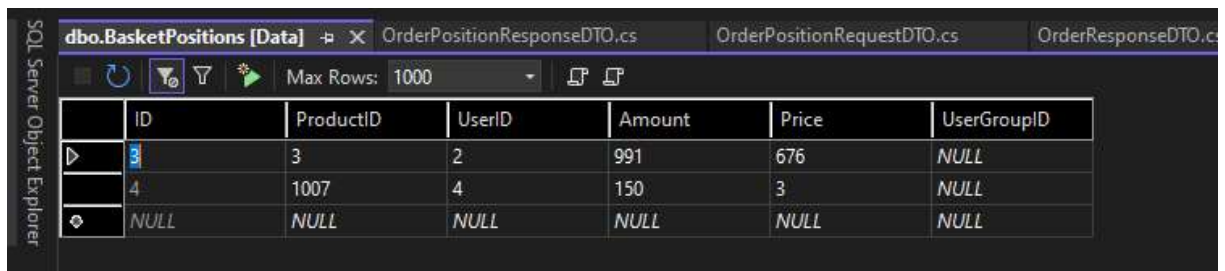
**Server response**

Code	Details
200	<b>Response headers</b> content-length: 0 date: Wed, 27 Mar 2024 11:11:06 GMT server: Kestrel

**Responses**

Code	Description	Links
200	Success	No links

Przed:



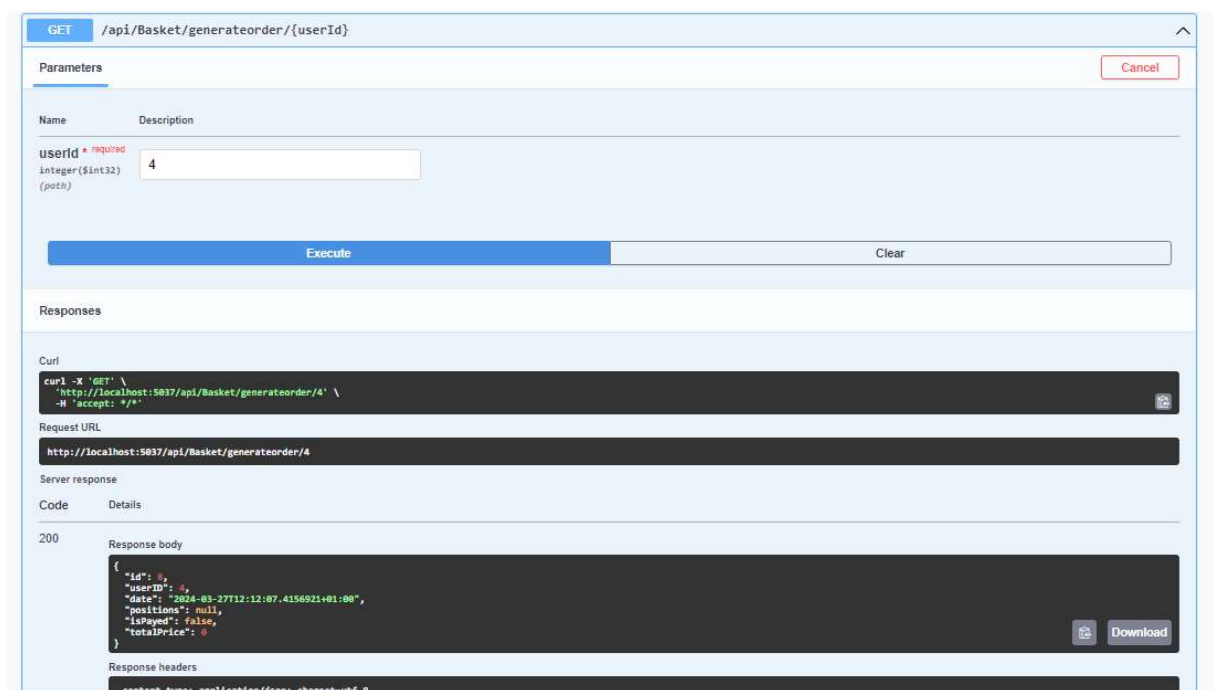
ID	ProductID	UserID	Amount	Price	UserGroupID
3	3	2	991	676	NULL
4	1007	4	150	3	NULL
NULL	NULL	NULL	NULL	NULL	NULL

Po:



ID	ProductID	UserID	Amount	Price	UserGroupID
4	1007	4	150	3	NULL
NULL	NULL	NULL	NULL	NULL	NULL

#### 4. Generowanie zamówienia:



GET /api/Basket/generateorder/{userId}

Parameters

Name: userId \* required  
integer(\$int32)  
(path)  
Value: 4

Execute Clear

Responses

Curl

```
curl -X 'GET' \
'http://localhost:5037/api/Basket/generateorder/4' \
-H 'accept: */*'
```

Request URL

http://localhost:5037/api/Basket/generateorder/4

Server response

Code: 200


Response body

```
{
  "id": 4,
  "userId": 4,
  "date": "2024-03-27T12:12:07.4156921+01:00",
  "positions": null,
  "isPaid": false,
  "totalPrice": 0
}
```

Response headers

content-type: application/json; charset=utf-8

W bazie:



ID	OrderID	Amount	Price	ProductID
4	4	991	676	3
5	8	150	3	1007
NULL	NULL	NULL	NULL	NULL

Debugger window showing the **dbo.Orders [Data]** table. The table has columns: ID, UserID, Date, and isPaid. The data is as follows:

ID	UserID	Date	isPaid
4	2	27.03.2024 11:24:26	True
7	2	27.03.2024 11:37:27	False
8	4	27.03.2024 12:12:07	False
NULL	NULL	NULL	NULL

## 5.Opłacenie zamówienia:

```

82      UserID = basket.UserID,
83      };
84
85      }
86
87      }
88
89      public void Pay(int userID, double value)
90      {
91          var order = _dbContext.Orders?.FirstOrDefault(x => x.UserID == userID);
92          if (order != null && !order.IsPaid)
93          {
94              var priceToPay = _dbContext.OrderPositions.Where(x => x.OrderID == order.ID).Sum(x => x.Amount);
95              var amount = priceToPay * amount;
96              priceToPay = priceToPay * amount;
97              if (priceToPay == value)
98              {
99                  order.IsPaid = true;
100                  _dbContext.SaveChanges();
101              }
102              else throw new ArgumentException("Invalid price");
103          }
104      }
105
106      public void RemoveProductFromBasket(int id)
107      {
108          BasketPosition? basket = _dbContext.BasketPositions.FirstOrDefault(b => b.ID == id);
109          if (basket == null)
110              return;
111          _dbContext.BasketPositions.Remove(basket);
112          _dbContext.SaveChanges();
113      }
114
115      }

```

**NowakowskiLab2** 1.0 OAS3

http://localhost:5037/swagger/v1/swagger.json

**Basket**

- POST /api/Basket
- PUT /api/Basket/{id}
- DELETE /api/Basket/{id}
- GET /api/Basket/generateorder/{userId}
- PUT /api/Basket/pay/{userId}/{value}

**Parameters**

Name	Description
userId * required	Integer (int32) (path)
value * required	number (double) (path)

Example: userId = 4, value = 450

Debugger window showing the **dbo.Orders [Data]** table. The table has columns: ID, UserID, Date, and isPaid. The data is as follows:

ID	UserID	Date	isPaid
7	2	27.03.2024 11:37:27	False
8	4	27.03.2024 12:12:07	True
NULL	NULL	NULL	NULL

## Część z DB:

Utworzyłem 2 interfejsy ProductDB i BasketDB:

```

6 [using System.Threading.Tasks;
7
8 namespace BLL.ServiceInterfaces
9 {
10     Odwołania: 4
11     public interface IBasketServiceDB
12     {
13         Odwołania: 2
14         void AddToBasket(BasketItemRequestDTO basket);
15         Odwołania: 2
16         void ChangeNumberOfProducts(int id, int numberOfProducts);
17         Odwołania: 2
18         void RemoveProductFromBasket(int id);
19         Odwołania: 2
20         OrderResponseDTO GenerateOrder(int userId);
21         Odwołania: 2
22         void Pay(int userId, double value);
23     }
24 }

```

```

1 namespace BLL.ServiceInterfaces
2 {
3     Odwołania: 4
4     public interface IProductServiceDB
5     {
6         Odwołania: 2
7         IEnumerable<ProductResponseDTO> GetProducts(ProductFilterRequestDTO filter);
8         Odwołania: 2
9         ProductResponseDTO GetProductById(int id);
10        Odwołania: 2
11        ProductResponseDTO AddProduct(ProductRequestDTO productDTO);
12        Odwołania: 2
13        void DeactivateProduct(int productId);
14        Odwołania: 2
15        void ActivateProduct(int productId);
16    }
17 }

```

```

1 Odwołania: 0
2 public class Program
3 {
4     Odwołania: 0
5     public static void Main(string[] args)
6     {
7         var builder = WebApplication.CreateBuilder(args);
8
9         // Add services to the container.
10
11         builder.Services.AddControllers();
12         builder.Services.AddDbContext<WebshopContext>();
13         builder.Services.AddScoped<IBasketService, BasketService>();
14         builder.Services.AddScoped<IBasketServiceDB, BasketServiceDB>();
15         builder.Services.AddScoped<IProductService, ProductService>();
16         builder.Services.AddScoped<IProductServiceDB, ProductServiceDB>();
17         // Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
18         builder.Services.AddEndpointsApiExplorer();
19         builder.Services.AddSwaggerGen();
20     }
21 }

```

## Testowanie ProductService DB:

### 1. Dodanie produktu:

Response

Curl

```
curl -X 'POST' \
  'http://localhost:5037/api/ProductDB' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "name": "woda",
    "price": 4,
    "image": "string",
    "isActive": true,
    "groupID": 1,
    "basketPositions": null
  }'
```

Request URL

http://localhost:5037/api/ProductDB

Server response

Code	Details
201	Response body
undocumented	<pre>{   "id": 0,   "name": "woda",   "price": 4,   "image": null,   "isActive": true,   "groupID": 1,   "groupName": "string",   "basketPositions": null }</pre>
	Response headers
	<pre>content-type: application/json; charset=utf-8 date: Wed, 03 Apr 2024 11:32:31 GMT location: http://localhost:5037/api/ProductDB/0 server: Kestrel transfer-encoding: chunked</pre>
Responses	
Code	Description
200	Success

### W bazie:

SQL Server Object Explorer

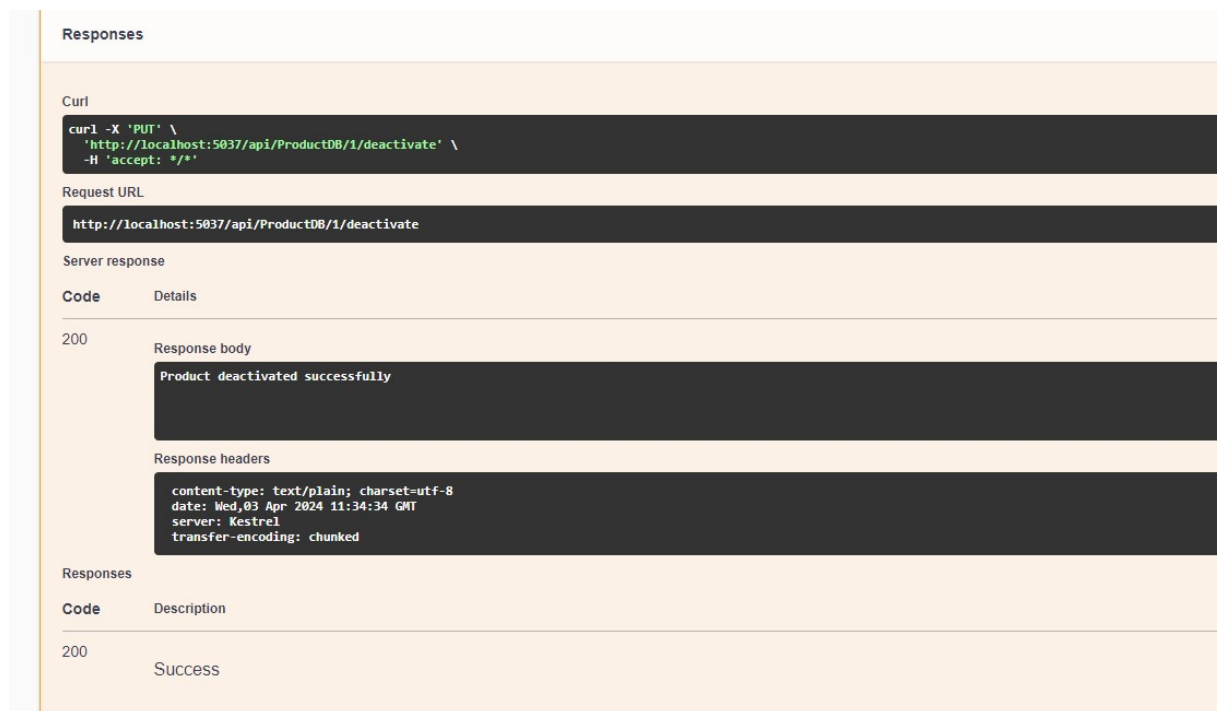
dbo.Products [Data]

Max Rows: 1000

	ID	Name	Price	Image	IsActive	GroupID
	1	woda	4	string	True	1
	NULL	NULL	NULL	NULL	NULL	NULL

## 2. Aktywacja i dezaktywacja produktu:

### Deaktywacja:



The screenshot shows a REST client interface with the following details:

- Responses** tab is selected.
- Curl**:

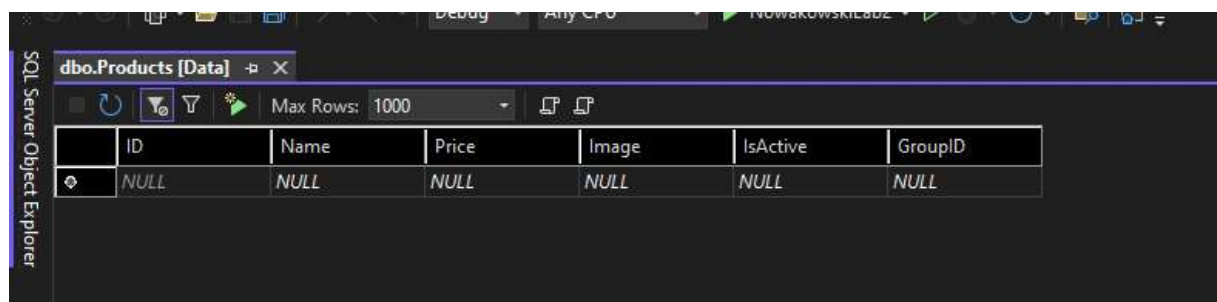
```
curl -X 'PUT' \
'http://localhost:5037/api/ProductDB/1/deactivate' \
-H 'accept: */*'
```
- Request URL**:

```
http://localhost:5037/api/ProductDB/1/deactivate
```
- Server response**:
  - Code**: 200
  - Details**:
    - Response body**:

```
Product deactivated successfully
```
    - Response headers**:

```
content-type: text/plain; charset=utf-8
date: Wed, 03 Apr 2024 11:34:34 GMT
server: Kestrel
transfer-encoding: chunked
```
- Responses** table:

Code	Description
200	Success



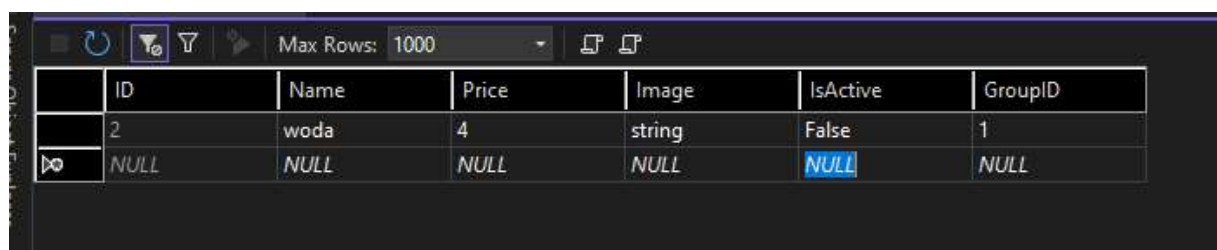
The screenshot shows the SQL Server Object Explorer with the **dbo.Products [Data]** table. The table has 7 columns: ID, Name, Price, Image, IsActive, and GroupID. The data is as follows:

ID	Name	Price	Image	IsActive	GroupID
NULL	NULL	NULL	NULL	NULL	NULL

Zgodnie z wytycznymi produkt został usunięty ponieważ nie należy do żadnego koszyka.

### Aktywacja:

#### Przed:



The screenshot shows the SQL Server Object Explorer with the **dbo.Products [Data]** table. The table has 7 columns: ID, Name, Price, Image, IsActive, and GroupID. The data is as follows:

ID	Name	Price	Image	IsActive	GroupID
2	woda	4	string	False	1
NULL	NULL	NULL	NULL	NULL	NULL

**id** \* required  
integer(\$int32)  
(path)

2

Execute Clear

**Responses**

Curl

```
curl -X 'POST' \
  'http://localhost:5037/api/Product08/2/activate' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:5037/api/Product08/2/activate
```

Server response

Code	Details
200	<p>Response body</p> <pre>Product activated successfully</pre> <p>Response headers</p> <pre>content-type: text/plain; charset=utf-8 date: Wed, 03 Apr 2024 11:36:07 GMT server: Kestrel transfer-encoding: chunked</pre>

Responses

Code	Description	Links
200	Success	No links

Po

ID	Name	Price	Image	IsActive	GroupID
2	woda	4	string	True	1
NULL	NULL	NULL	NULL	NULL	NULL

### 3. Pobieranie listy produktów:

**Name** **Description**

Name  
string  
(query)  
mleko

GroupName  
string  
(query)  
GroupName

GroupID  
integer(\$int32)  
(query)  
GroupID

IncludeInactive  
boolean  
(query)  
true

SortBy  
string  
(query)  
Name

SortAsc  
boolean  
(query)  
false

Execute Clear

**Responses**

Curl

```
curl -H 'GET' \
  'http://localhost:5037/api/Product08?Name=mleko&IncludeInactive=true&SortBy=Name&SortAsc=false' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:5037/api/Product08?Name=mleko&IncludeInactive=true&SortBy=Name&SortAsc=false
```

Server response

Code	Details
200	<p>Response body</p> <pre>{   "id": 1,   "name": "mleko",   "price": 1,   "image": "",   "isActive": true,   "groupId": 1,   "groupName": "mleko",   "basketPositions": null }</pre>



(query)

GroupName  
string  
(query)

GroupID  
integer(\$int32)  
(query)

IncludeInactive  
boolean  
(query)

SortBy  
string  
(query)

SortAsc  
boolean  
(query)

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:5037/api/ProductDB?GroupID=1&IncludeInactive=true&SortBy=Name&SortAsc=false' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:5037/api/ProductDB?GroupID=1&IncludeInactive=true&SortBy=Name&SortAsc=false
```

Server response

Code Details

200

Response body

```
{
  "id": 1,
  "name": "woda",
  "price": 1,
  "image": "",
  "isActive": true,
  "groupID": 1,
  "groupName": "string",
  "basketPositions": null
},
{
  "id": 1,
  "name": "wisko",
  "price": 1,
  "image": "",
  "isActive": true,
  "groupID": 1,
  "groupName": "string",
  "basketPositions": null
},
{
  "id": 1,
  "name": "kawa"
}
```

GroupName  
string  
(query)

GroupID  
integer(\$int32)  
(query)

IncludeInactive  
boolean  
(query)

SortBy  
string  
(query)

SortAsc  
boolean  
(query)

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:5037/api/ProductDB?GroupID=1&IncludeInactive=true&SortBy=Name&SortAsc=true' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:5037/api/ProductDB?GroupID=1&IncludeInactive=true&SortBy=Name&SortAsc=true
```

Server response

Code Details

200

Response body

```
{
  "id": 1,
  "name": "cola",
  "price": 1,
  "image": "",
  "isActive": true,
  "groupID": 1,
  "groupName": "string",
  "basketPositions": null
},
{
  "id": 1,
  "name": "kawa",
  "price": 1,
  "image": "",
  "isActive": true,
  "groupID": 1,
  "groupName": "string",
  "basketPositions": null
},
{
  "id": 1,
  "name": "wisko"
}
```



#### 4. Pobieranie produktu po ID:

The screenshot shows a REST client interface with the following sections:

- Parameters:** A table with columns 'Name' and 'Description'. It contains one parameter: 

Name	Description
id * required	integer(\$int32) (path)
- Execute:** A blue button to execute the request.
- Responses:** A section containing:
  - Curl:**

```
curl -X 'GET' \
  'http://localhost:5037/api/ProductDB/3' \
  -H 'accept: */*'
```
  - Request URL:** `http://localhost:5037/api/ProductDB/3`
  - Server response:** A table with columns 'Code' and 'Details'. It shows a response with status code 200 and a JSON body: 

```
{
  "id": 3,
  "name": "mleko",
  "price": 5,
  "image": null,
  "isActive": true,
  "groupID": 1,
  "groupName": "string",
  "basketPositions": null
}
```
  - Response headers:**

```
content-type: application/json; charset=utf-8
date: Wed, 03 Apr 2024 11:38:27 GMT
server: Kestrel
transfer-encoding: chunked
```
- Responses Table:** A table with columns 'Code', 'Description', and 'Links'. It contains one row: 

Code	Description	Links
200	Success	No links

## Testowanie Basket Service DB

### 1. Dodanie produktu do koszyka

```
{
  "productID": 2,
  "userID": 1,
  "amount": 10,
  "price": 20
}
```

ExecuteClear

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:5037/api/BasketDB/add' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "productID": 2,
    "userID": 1,
    "amount": 10,
    "price": 20
  }'
```

Request URL

http://localhost:5037/api/BasketDB/add

Server response

CodeDetails

200

Response body

Product added to basket successfully

Download

Response headers

content-type: text/plain; charset=utf-8  
date: Wed, 03 Apr 2024 11:39:11 GMT  
server: Kestrel  
transfer-encoding: chunked

Responses

CodeDescriptionLinks

200SuccessNo links

W bazie:

ID	ProductID	UserID	Amount	Price	UserGroupID
	2	1	10	20	NULL
NULL	NULL	NULL	NULL	NULL	NULL

## 2. Zmiana liczby produktów

PUT /api/BasketDB/{id}/changeamount/{amount}

Parameters

Name	Description
id * required integer(\$int32) (path)	1
amount * required integer(\$int32) (path)	40

Execute Clear

Responses

Curl

```
curl -X 'PUT' \
  http://localhost:5037/api/BasketDB/1/changeamount/40 \
  -H 'accept: */*'
```

Request URL

http://localhost:5037/api/BasketDB/1/changeamount/40

Server response

Code Details

200

Response body

Number of products changed successfully

Response headers

```
content-type: text/plain; charset=utf-8
date: Wed, 03 Apr 2024 11:39:53 GMT
server: Kestrel
transfer-encoding: chunked
```

SQL Server Object Explorer

dbo.BasketPositions [Data] | X | dbo.Products [Data]

Max Rows: 1000

	ID	ProductID	UserID	Amount	Price	UserGroupID
▶	1	2	1	40	20	NULL
⬇	NULL	NULL	NULL	NULL	NULL	NULL

## 3. Usunięcie produktu z koszyka

Przed:

SQL Server Object Explorer

dbo.BasketPositions [Data] | X | dbo.Products [Data]

Max Rows: 1000

	ID	ProductID	UserID	Amount	Price	UserGroupID
▶	1	2	1	40	20	NULL
	2	3	2	10	2000	NULL
⬇	NULL	NULL	NULL	NULL	NULL	NULL

Po:

PUT /api/BasketDB/pay/{userId}/{value}

DELETE /api/BasketDB/{id}/remove

Parameters

NameDescription

id \* requiredinteger(\$int32)(path)2

ExecuteClear

Responses

Curl

```
curl -X 'DELETE' \
'http://localhost:5037/api/BasketDB/2/remove' \
-H 'accept: */*'
```

Request URL

http://localhost:5037/api/BasketDB/2/remove

Server response

CodeDetails

200

Response body

Product removed from basket successfully

Download

Response headers

```
content-type: text/plain; charset=utf-8
date: Wed, 03 Apr 2024 11:41:06 GMT
server: Kestrel
transfer-encoding: chunked
```

	ID	ProductID	UserID	Amount	Price	UserGroupID
	1	2	1	40	20	NULL
	NULL	NULL	NULL	NULL	NULL	NULL

## 4.Generowanie zamówienia:

POST /api/BasketDB/generateorder/{userId}

Parameters

NameDescription

userId \* requiredinteger(\$int32)(path)1

ExecuteClear

Responses

Curl

```
curl -X 'POST' \
'http://localhost:5037/api/BasketDB/generateorder/1' \
-H 'accept: */*' \
-d ''
```

Request URL

http://localhost:5037/api/BasketDB/generateorder/1

Server response

CodeDetails

200

Response body

```
{
  "id": 1,
  "userId": 1,
  "date": "2024-04-03T13:41:37.4533333",
  "positions": null,
  "isPaid": false,
  "totalPrice": 0
}
```

Download

Response headers

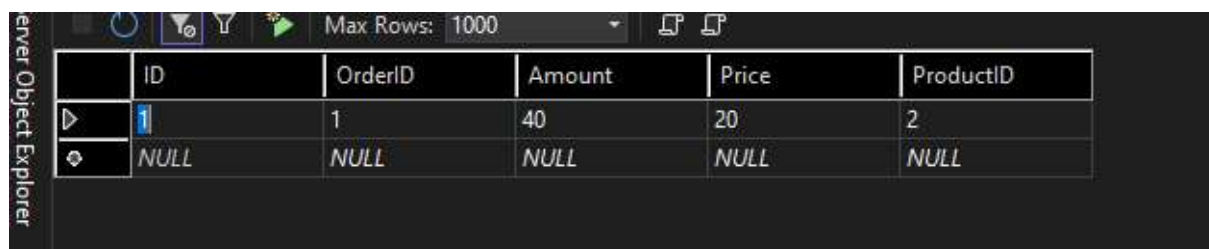
```
content-type: application/json; charset=utf-8
date: Wed, 03 Apr 2024 11:41:37 GMT
server: Kestrel
transfer-encoding: chunked
```

Responses

CodeDescriptionLinks

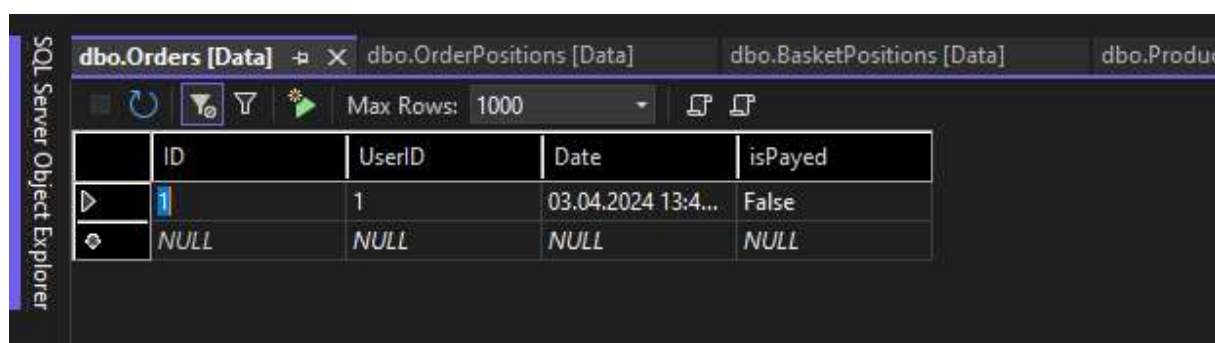
200SuccessNo links

W bazie:



Max Rows: 1000

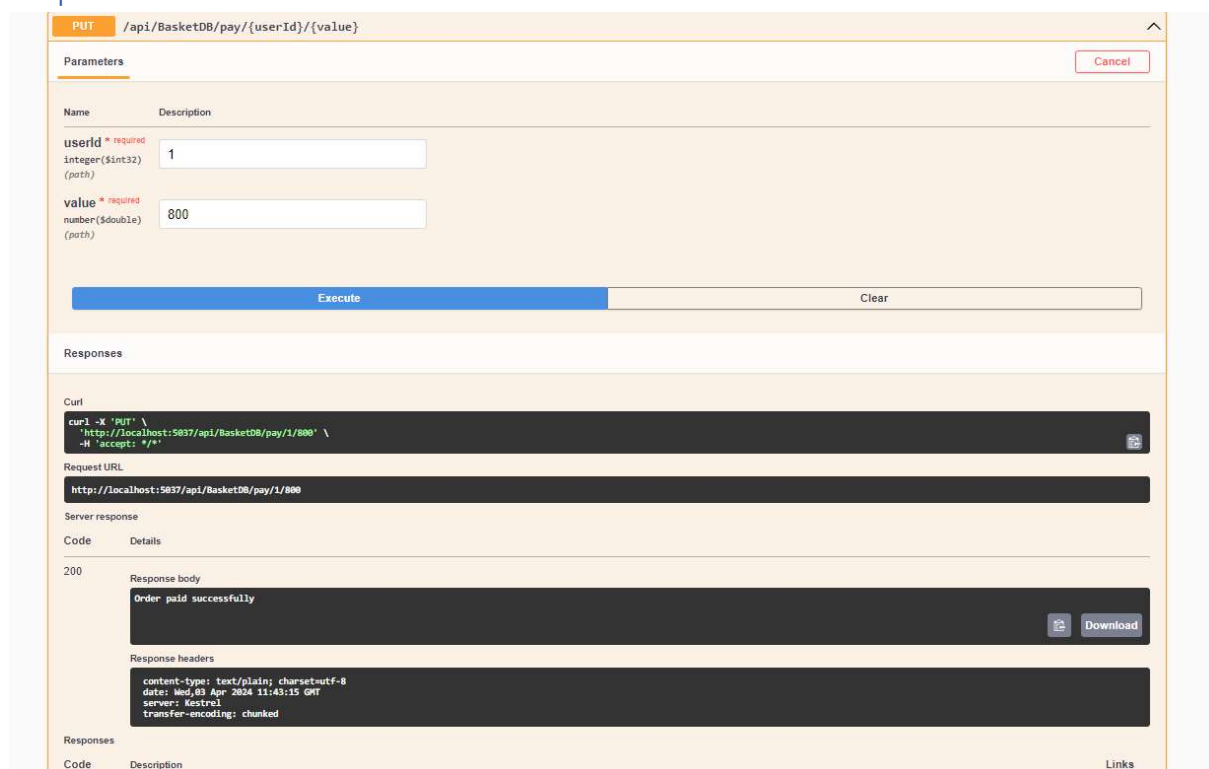
	ID	OrderID	Amount	Price	ProductID
▶	1	1	40	20	2
◆	NULL	NULL	NULL	NULL	NULL



Max Rows: 1000

	ID	UserID	Date	isPaid
▶	1	1	03.04.2024 13:4...	False
◆	NULL	NULL	NULL	NULL

## 5. Opłacenie zamówienia:



PUT /api/BasketDB/pay/{userId}/{value}

Parameters

Cancel

Name	Description
userid * required integer(\$int32) (path)	1
value * required number(\$double) (path)	800

Execute Clear

Responses

Curl

```
curl -X 'PUT' \
'http://localhost:5037/api/BasketDB/pay/1/800' \
-H 'accept: */*'
```

Request URL

http://localhost:5037/api/BasketDB/pay/1/800

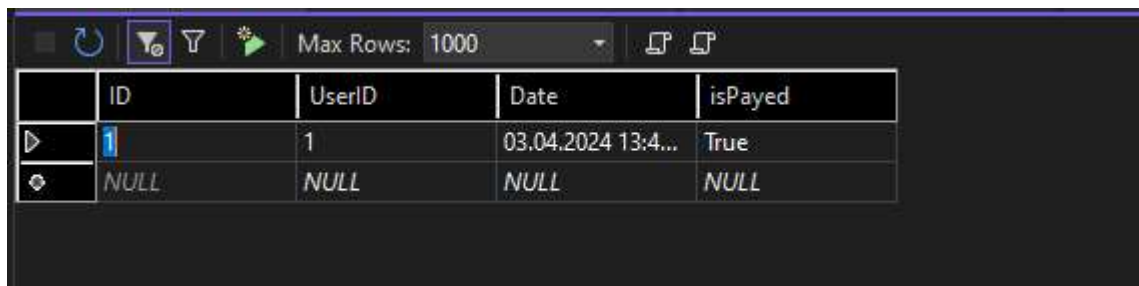
Server response

Code	Details
200	<p>Response body</p> <p>Order paid successfully</p> <p>Response headers</p> <p>content-type: text/plain; charset=utf-8 date: Wed, 03 Apr 2024 11:43:15 GMT server: Kestrel transfer-encoding: chunked</p>

Responses

Code	Description
------	-------------

Links



	ID	UserID	Date	isPaid
▶	1	1	03.04.2024 13:4...	True
⚙	NULL	NULL	NULL	NULL

## PODSUMOWANIE/WNIOSKI

W ramach niniejszego sprawozdania przeprowadziłem porównanie podejścia ORM (Object-Relational Mapping) do osadzenia logiki aplikacji w bazie danych. Podczas wykonywania ćwiczeń 4 i 5 pracowałem zarówno z Entity Framework (EF), który wykorzystuje podejście ORM, jak i bezpośrednim podejściem do bazy danych opartym na procedurach składowanych i triggerach.

Podczas moich studiów inżynierskich miałem głównie do czynienia z Entity Framework oraz innymi narzędziami ORM, dlatego podejście oparte na ORM było mi dobrze znane i stosunkowo łatwe do pracy. Jednak kiedy przyszło mi pracować z podejściem opartym na procedurach i triggerach, natknąłem się na nową dla mnie dziedzinę, z którą nie miałem wcześniej do czynienia i wiele rzeczy musiałem szukać w internecie co zajęło mnóstwo czasu. Na ten moment posiadam niewielkie umiejętności w pisaniu procedur i triggerów.

Z moich doświadczeń wynika, że podejście ORM oferuje wiele korzyści, takich jak ułatwienie pracy z bazą danych poprzez mapowanie obiektów na struktury relacyjne, czy też automatyczne generowanie zapytań SQL. Dodatkowo, ORM pozwala na stosowanie obiektowych technik programowania, co ułatwia zrozumienie i utrzymanie kodu.

Z drugiej strony, podejście oparte na procedurach i triggerach może być bardziej skuteczne w przypadku aplikacji wymagających wysokiej wydajności lub obsługujących dużą ilość danych. Procedury składowane mogą być zoptymalizowane na poziomie bazy danych, co może przynieść korzyści w przypadku złożonych operacji czy zapytań.

Warto zauważyć, że wybór między podejściem ORM a podejściem opartym na bazie danych zależy od indywidualnych potrzeb i charakterystyki projektu. Każda z tych metod ma swoje zalety i wady, które należy uwzględnić podczas projektowania aplikacji.