

A dark blue vertical bar runs along the left edge of the page. A blue arrow-shaped banner points to the right from this bar, containing the text 'COS30041'. In the bottom-left corner, there are several thin, curved, light blue lines that sweep upwards and to the right.

COS30041

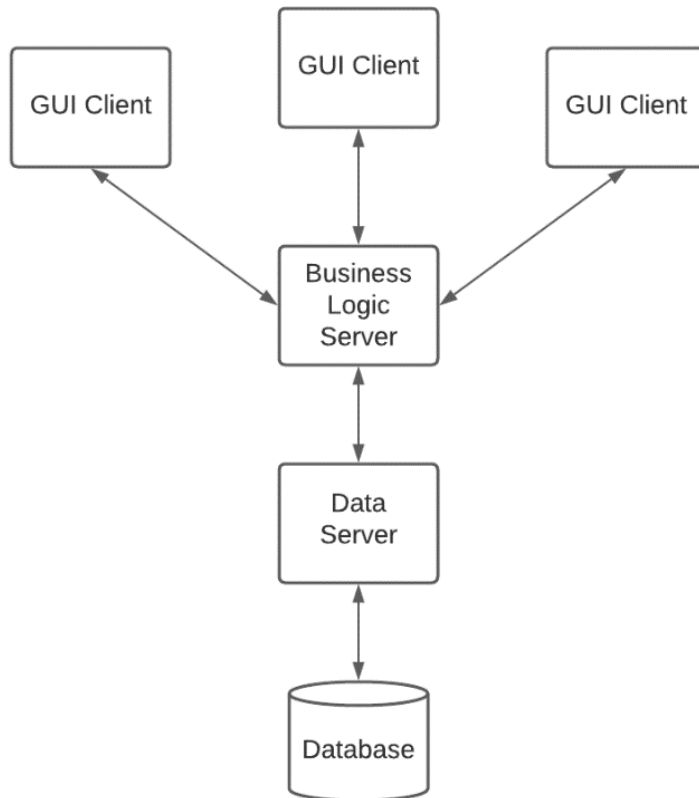
Creating Secure and Scalable Software

Pass task 05

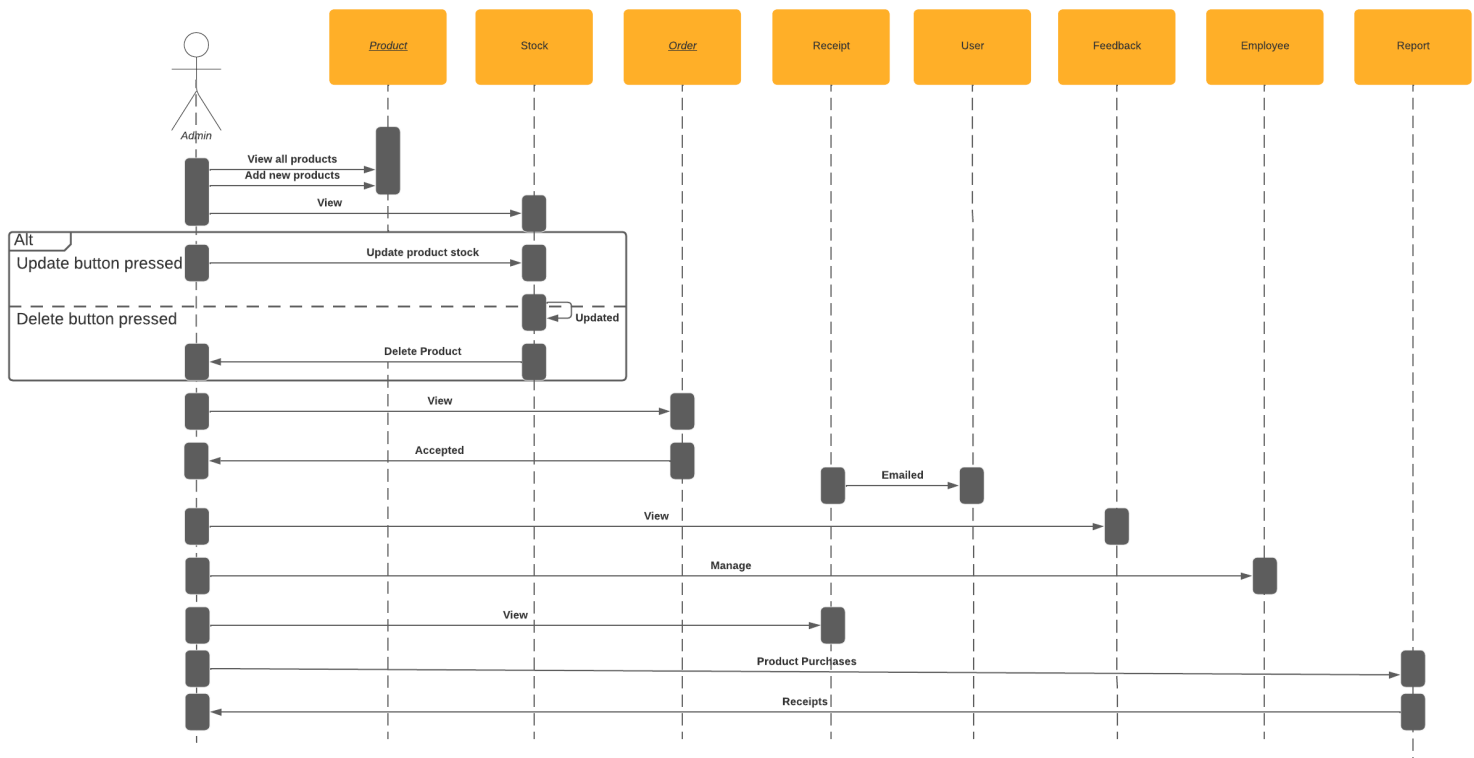
Noor Ul Ain Khurshid 102763334
SEMESTER 02, 2022

MICROSERVICE PROJECT

Architectural diagram representing the exchange of information between the e-commerce application and the user:



Sequence diagram representing the exchange of information between the e-commerce application and the user:



Integration Points of the application:

User: The client or user is the main endpoint of the program who handles the database using the developed API. They can interact with the products, the basket, and the cart.

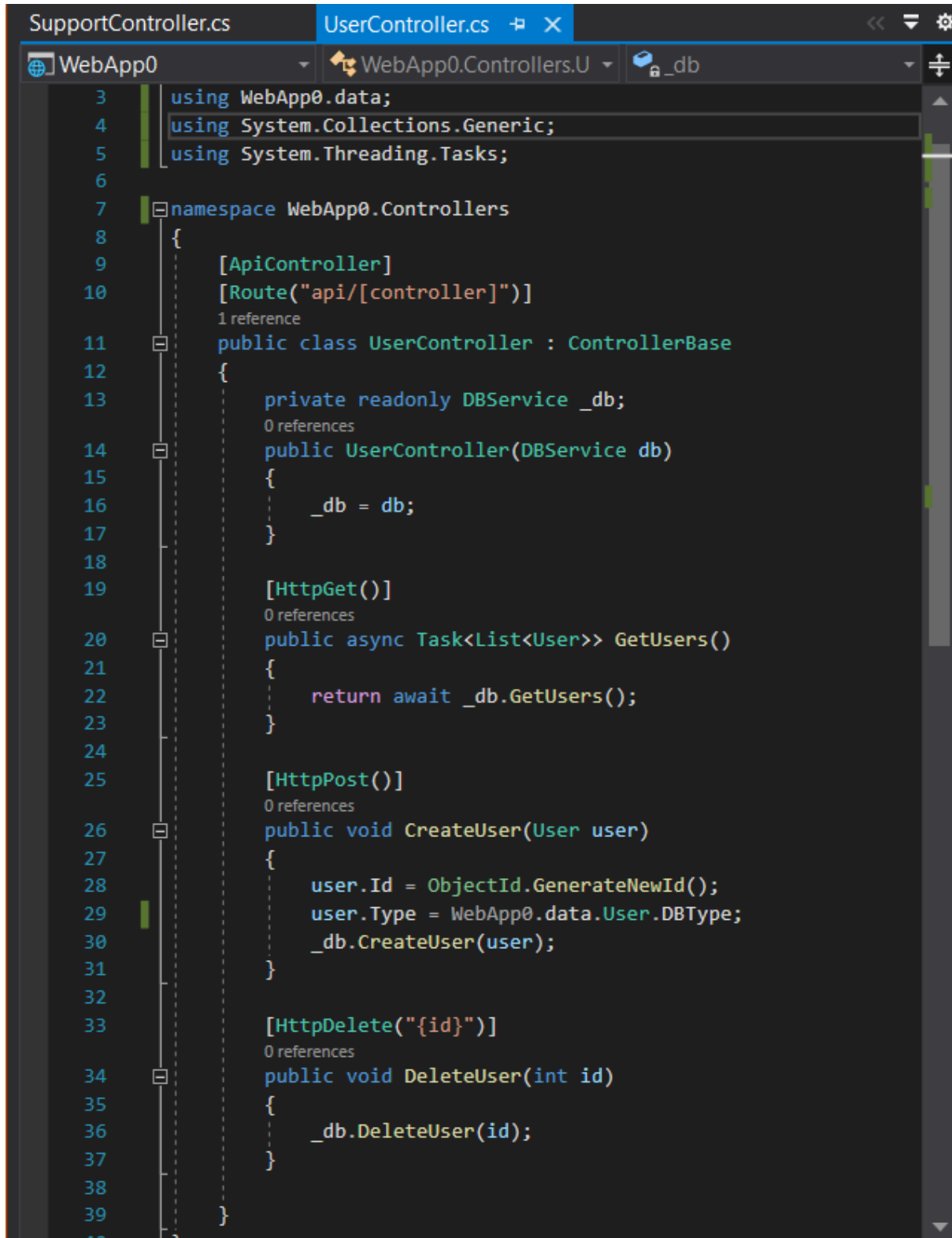
Product: The products in the application can be viewed and new products can be created and deleted from the database.

Basket: The user can view products in the basket and add new products to the basket and delete them as well.

Support: The user can contact the support staff and give feedback to them and view feedbacks and delete them as well.


LT2.3. Develop the REST API.

User Controller



```
SupportController.cs  UserController.cs  WebApp0  WebApp0.Controllers.U  _db
3  using WebApp0.data;
4  using System.Collections.Generic;
5  using System.Threading.Tasks;
6
7  namespace WebApp0.Controllers
8  {
9      [ApiController]
10     [Route("api/[controller]")]
11     public class UserController : ControllerBase
12     {
13         private readonly DBService _db;
14         public UserController(DBService db)
15         {
16             _db = db;
17         }
18
19         [HttpGet()]
20         public async Task<List<User>> GetUsers()
21         {
22             return await _db.GetUsers();
23         }
24
25         [HttpPost()]
26         public void CreateUser(User user)
27         {
28             user.Id = ObjectId.GenerateNewId();
29             user.Type = WebApp0.data.User.DBType;
30             _db.CreateUser(user);
31         }
32
33         [HttpDelete("{id}")]
34         public void DeleteUser(int id)
35         {
36             _db.DeleteUser(id);
37         }
38     }
39 }
```

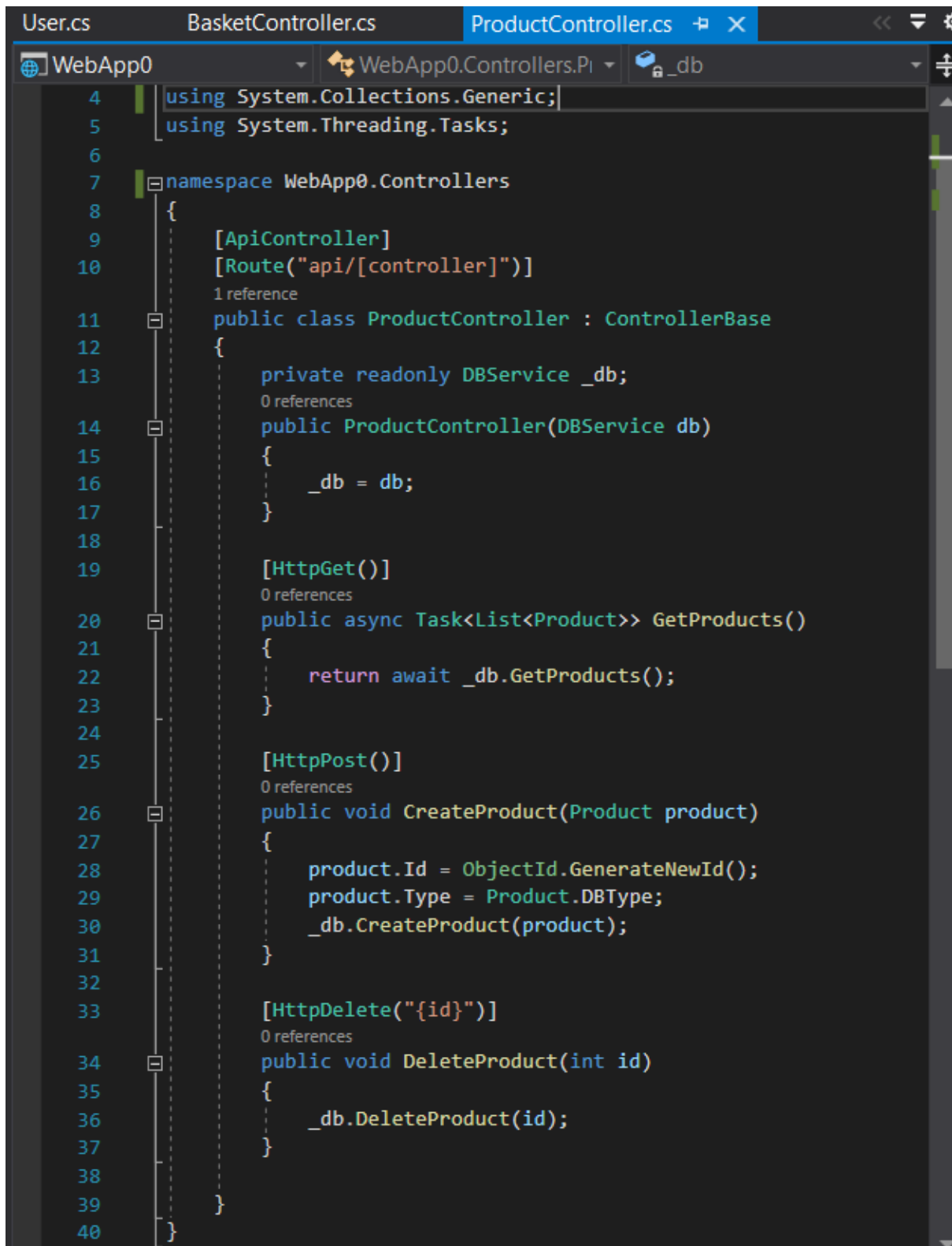
Basket Controller



The image shows a screenshot of a code editor with the file `BasketController.cs` open. The project is `WebApp0`, and the file is located in the `WebApp0.Controllers.B` namespace. The code defines a `BasketController` class that inherits from `ControllerBase`. It includes several attributes: `[ApiController]` and `[Route("api/[controller]")]`. The class has a private readonly `DBService _db` and a constructor that takes a `DBService db` parameter. It also has four methods: `GetBasket()` (an async method returning a `Task<List<Basket>>`), `CreateBasket(Basket basket)` (a void method), and `DeleteBasket(int id)` (a void method). The `GetBasket()` method calls `_db.GetBaskets()`. The `CreateBasket()` method calls `ObjectId.GenerateNewId()`, `Basket.DBType`, and `_db.CreateBasket(basket)`. The `DeleteBasket()` method calls `_db.DeleteBasket(id)`.

```
4 using System.Collections.Generic;
5 using System.Threading.Tasks;
6
7 namespace WebApp0.Controllers
8 {
9     [ApiController]
10    [Route("api/[controller]")]
11    public class BasketController : ControllerBase
12    {
13        private readonly DBService _db;
14        public BasketController(DBService db)
15        {
16            _db = db;
17        }
18
19        [HttpGet()]
20        public async Task<List<Basket>> GetBasket()
21        {
22            return await _db.GetBaskets();
23        }
24
25        [HttpPost()]
26        public void CreateBasket(Basket basket)
27        {
28            basket.Id = ObjectId.GenerateNewId();
29            basket.Type = Basket.DBType;
30            _db.CreateBasket(basket);
31        }
32
33        [HttpDelete("{id}")]
34        public void DeleteBasket(int id)
35        {
36            _db.DeleteBasket(id);
37        }
38    }
39 }
40 }
```

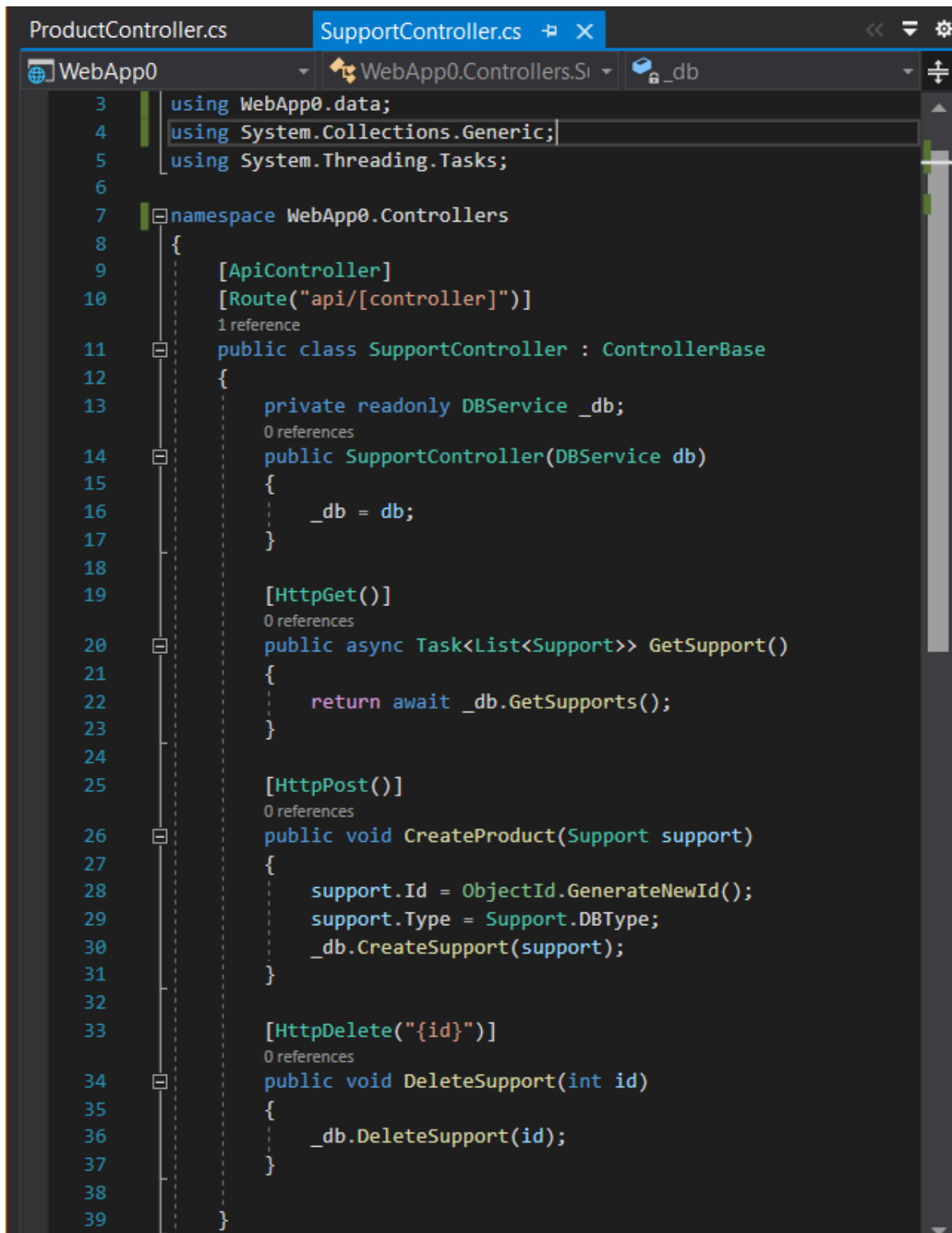
Product Controller



The screenshot shows the Visual Studio IDE with the 'ProductController.cs' file open. The file is part of the 'WebApp0' project, specifically within the 'WebApp0.Controllers.Pi' namespace. The code defines a 'ProductController' class that inherits from 'ControllerBase'. It includes attributes for API routing and a reference to 'DBService'. The class contains several methods: a constructor, a 'GetProducts' method using async/await, a 'CreateProduct' method, and a 'DeleteProduct' method. The code is as follows:

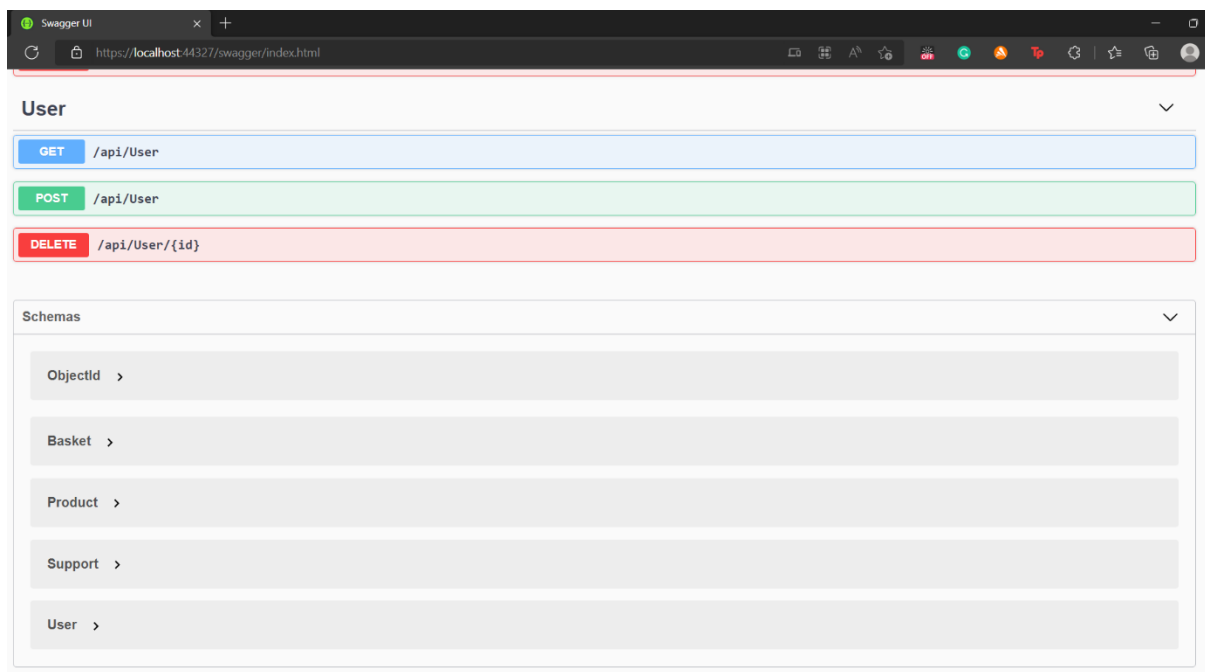
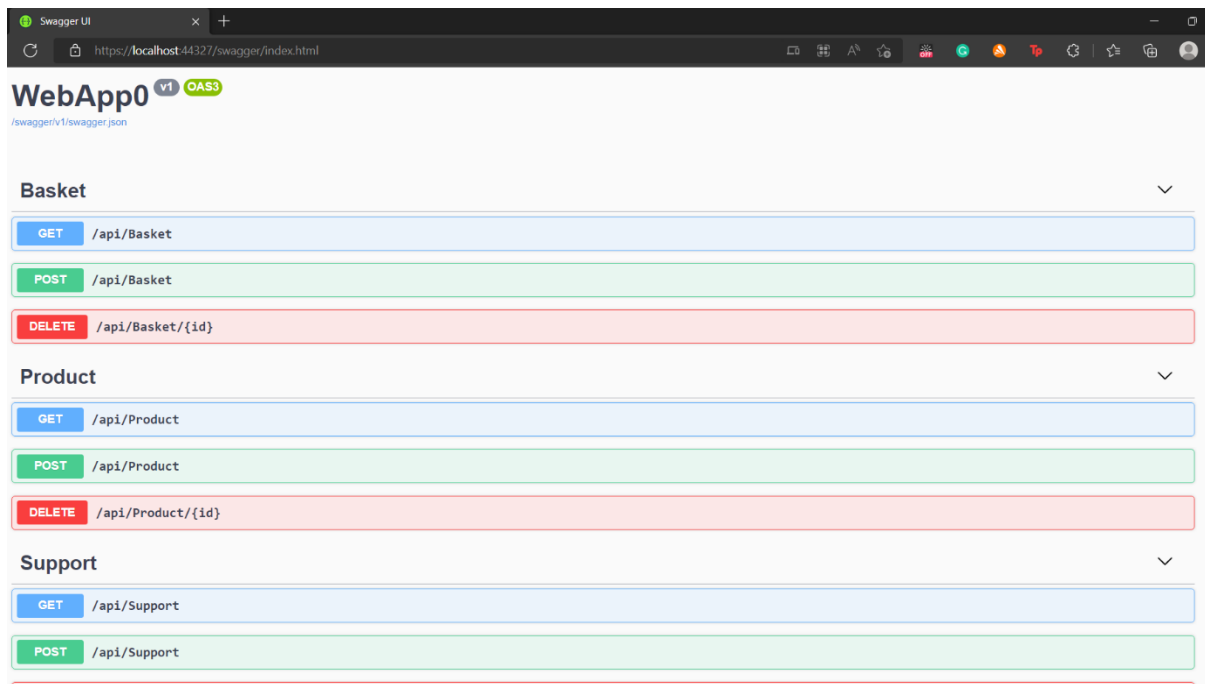
```
4  using System.Collections.Generic;
5  using System.Threading.Tasks;
6
7  namespace WebApp0.Controllers
8  {
9      [ApiController]
10     [Route("api/[controller]")]
11     public class ProductController : ControllerBase
12     {
13         private readonly DBService _db;
14         public ProductController(DBService db)
15         {
16             _db = db;
17         }
18
19         [HttpGet()]
20         public async Task<List<Product>> GetProducts()
21         {
22             return await _db.GetProducts();
23         }
24
25         [HttpPost()]
26         public void CreateProduct(Product product)
27         {
28             product.Id = ObjectId.GenerateNewId();
29             product.Type = Product.DBType;
30             _db.CreateProduct(product);
31         }
32
33         [HttpDelete("{id}")]
34         public void DeleteProduct(int id)
35         {
36             _db.DeleteProduct(id);
37         }
38     }
39 }
40
```

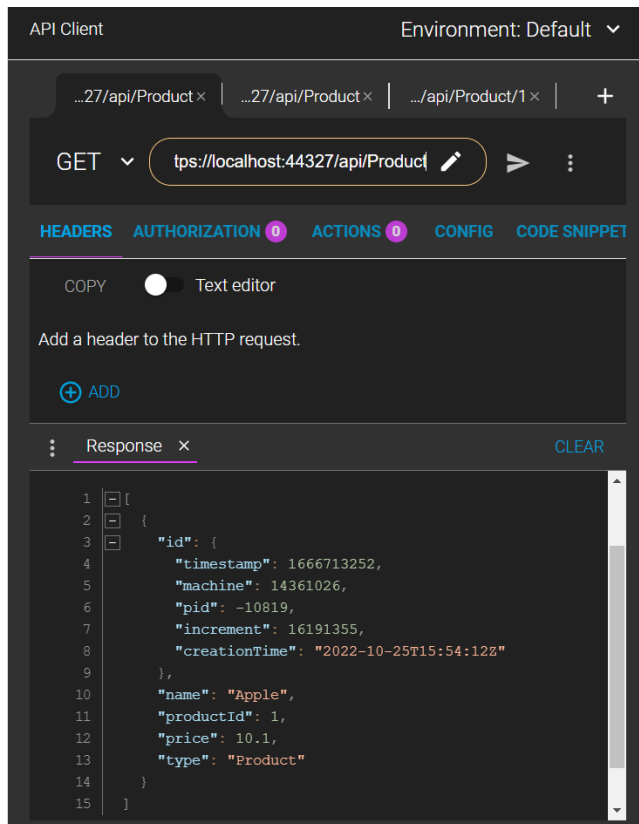
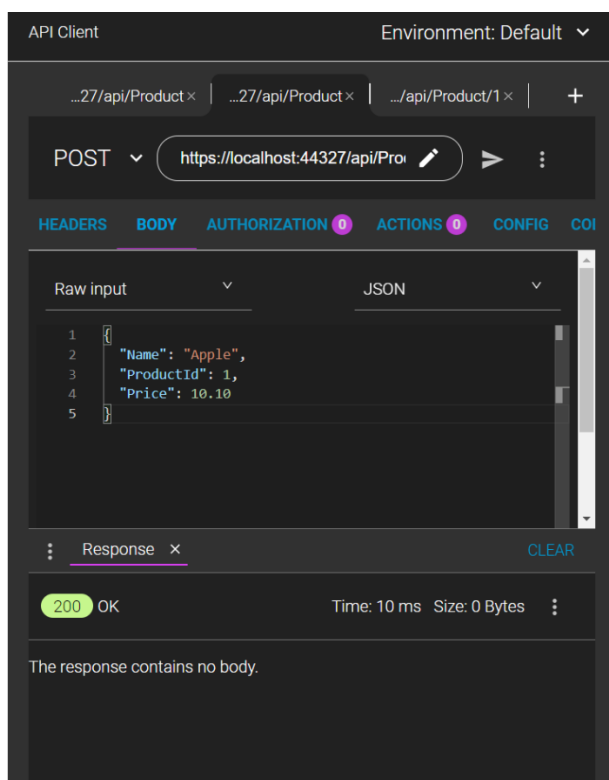
Support Controller

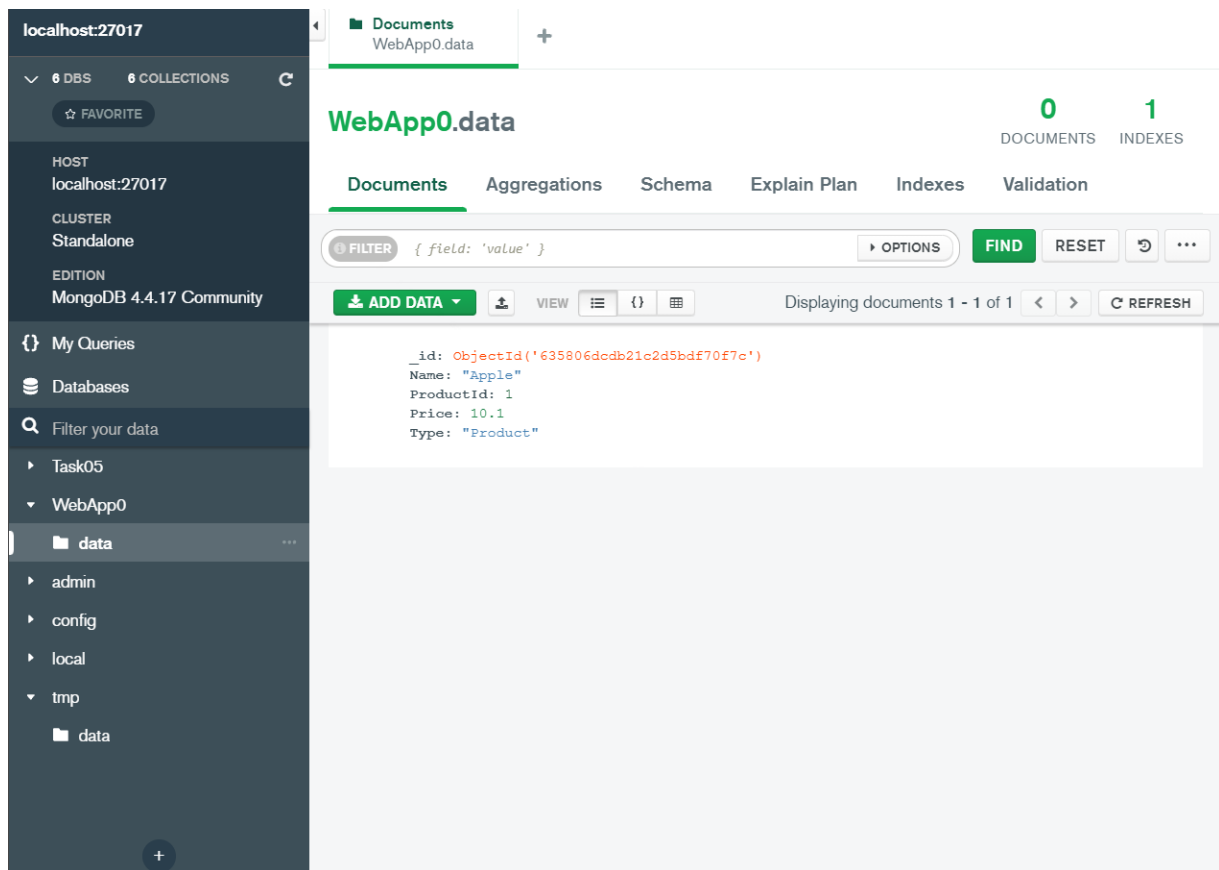


```
ProductController.cs | SupportController.cs |
WebApp0 | WebApp0.Controllers.Si | _db
3 using WebApp0.data;
4 using System.Collections.Generic;
5 using System.Threading.Tasks;
6
7 namespace WebApp0.Controllers
8 {
9     [ApiController]
10    [Route("api/[controller]")]
11    public class SupportController : ControllerBase
12    {
13        private readonly DBService _db;
14        public SupportController(DBService db)
15        {
16            _db = db;
17        }
18
19        [HttpGet()]
20        public async Task<List<Support>> GetSupport()
21        {
22            return await _db.GetSupports();
23        }
24
25        [HttpPost()]
26        public void CreateProduct(Support support)
27        {
28            support.Id = ObjectId.GenerateNewId();
29            support.Type = Support.DBType;
30            _db.CreateSupport(support);
31        }
32
33        [HttpDelete("{id}")]
34        public void DeleteSupport(int id)
35        {
36            _db.DeleteSupport(id);
37        }
38    }
39 }
```

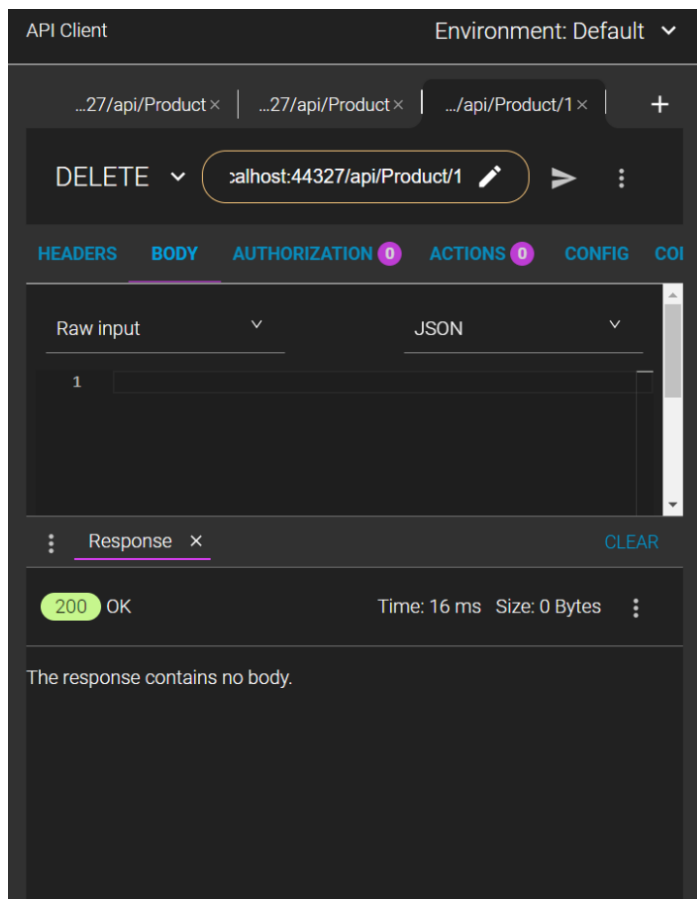
Output:



Operation of the application using ARC.**PRODUCT***Create Product**View Product*

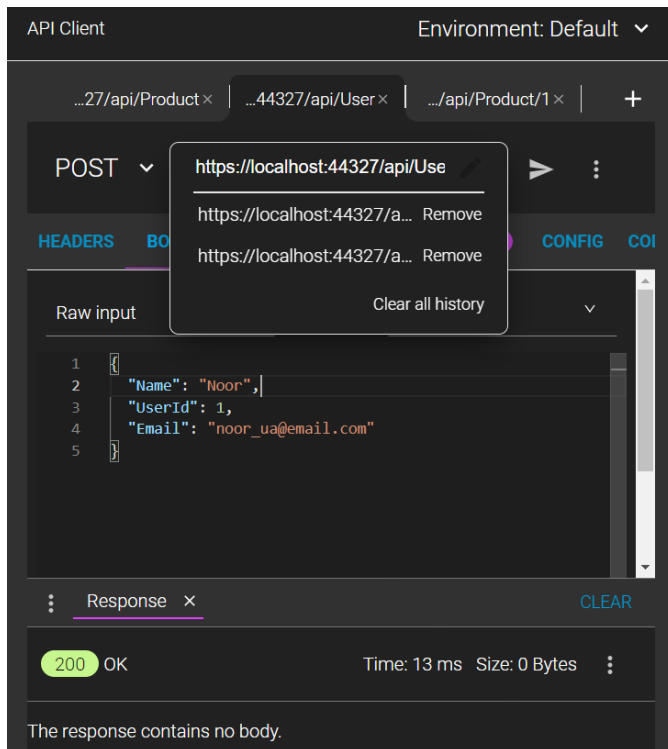


Delete Product

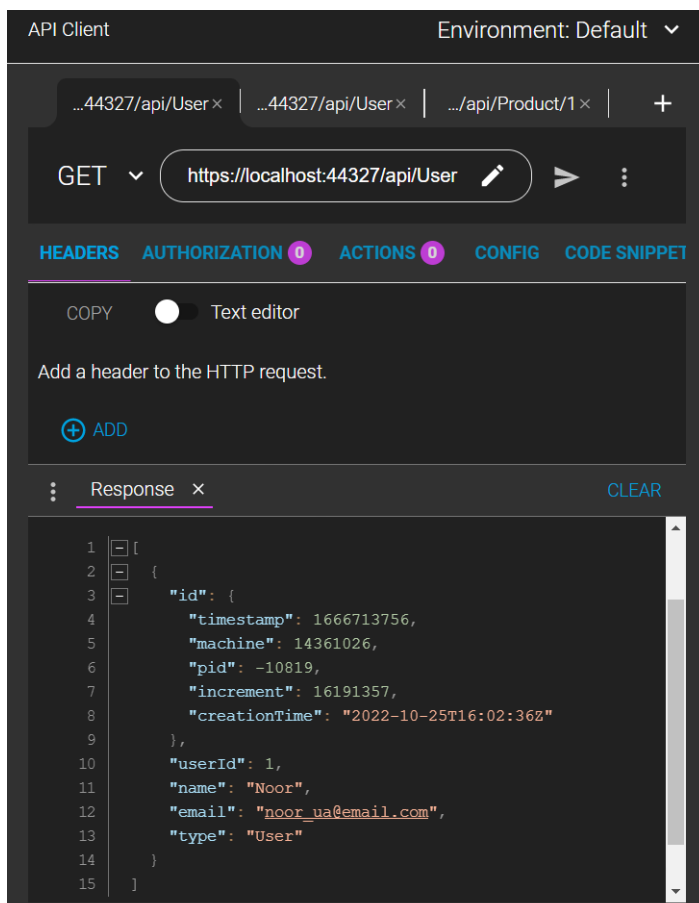


USER

Create User



View User



The screenshot shows the MongoDB WebApp0.data interface. At the top, there's a breadcrumb 'Documents' with a sub-item 'WebApp0.data'. Below this, the title 'WebApp0.data' is displayed, followed by counts: '0 DOCUMENTS' and '1 INDEXES'. A navigation bar includes 'Documents', 'Aggregations', 'Schema', 'Explain Plan', 'Indexes', and 'Validation'. The 'Documents' tab is active, showing a filter bar with '{ field: 'value' }', 'OPTIONS', 'FIND', 'RESET', and a refresh icon. Below the filter bar, there's an 'ADD DATA' button and a 'VIEW' section with icons for list, JSON, and grid views. The status bar indicates 'Displaying documents 1 - 2 of 2' with navigation arrows and a 'REFRESH' button. Two documents are listed:

```
{
  "_id": ObjectId('635806dcd21c2d5bdf70f7c'),
  "Name": "Apple",
  "ProductId": 1,
  "Price": 10.1,
  "Type": "Product"
}
```

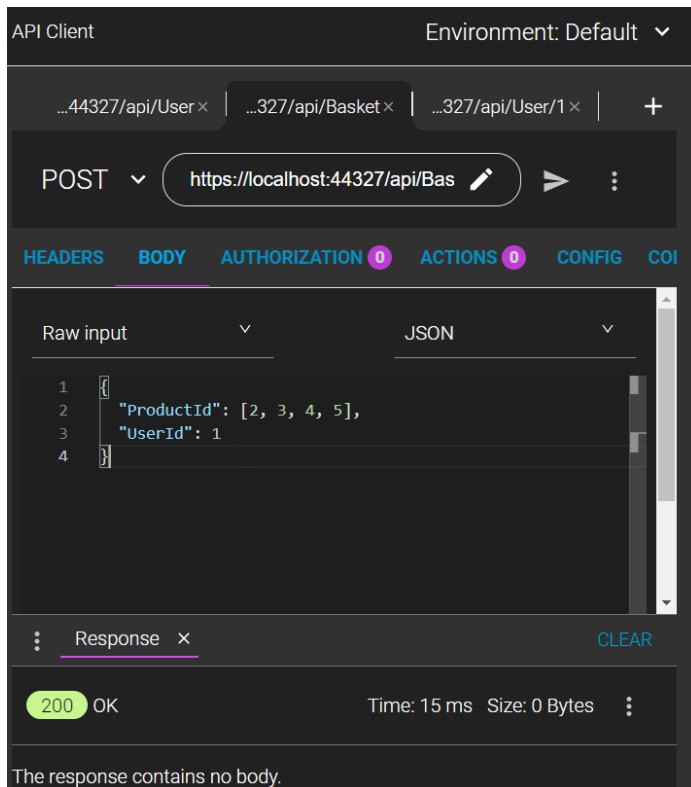
```
{
  "_id": ObjectId('6358089cdb21c2d5bdf70f7d'),
  "UserId": 1,
  "Name": "Noor",
  "Email": "noor_ua@email.com",
  "Type": "User"
}
```

Delete User

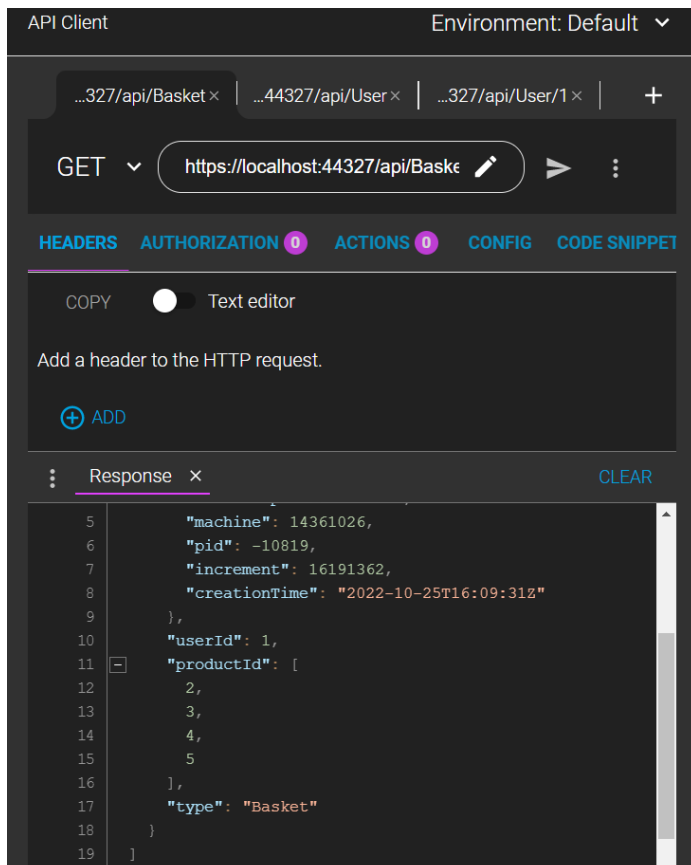
The screenshot shows the API Client interface. At the top, it says 'API Client' and 'Environment: Default'. Below this, there are tabs for different requests: '...44327/api/User', '...44327/api/User', and '...327/api/User/1'. The 'DELETE' method is selected for the first tab, with the URL 'https://localhost:44327/api/l'. The 'HEADERS' tab is active, showing 'Raw input' and 'JSON' views. The 'Response' tab is also active, showing a '200 OK' status, 'Time: 20 ms', and 'Size: 0 Bytes'. The message 'The response contains no body.' is displayed.

BASKET

Create Basket



View Basket



WebApp0.data 0 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

FILTER { field: 'value' } **OPTIONS** **FIND** **RESET** **REFRESH**

ADD DATA **VIEW** **Displaying documents 1 - 7 of 7**

```
{
  "_id": ObjectId('635809ffdb21c2d5bdf70f80'),
  "Name": "Grapes",
  "ProductId": 4,
  "Price": 40.4,
  "Type": "Product"
}
```

```
{
  "_id": ObjectId('63580a08db21c2d5bdf70f81'),
  "Name": "Mango",
  "ProductId": 5,
  "Price": 50.5,
  "Type": "Product"
}
```

```
{
  "_id": ObjectId('63580a3bdb21c2d5bdf70f82'),
  "UserId": 1,
  "ProductId": Array
    0: 2
    1: 3
    2: 4
    3: 5
  "Type": "Basket"
}
```

```
{
  "_id": ObjectId('63580a7fdb21c2d5bdf70f84'),
  "UserId": 1
}
```

Delete Basket

API Client Environment: Default

...327/api/Basket x | ...44327/api/User x | ...7/api/Basket/1 x | +

DELETE localhost:44327/api/Basket/1

HEADERS **BODY** **AUTHORIZATION** 0 **ACTIONS** 0 **CONFIG** **COI**

Raw input JSON

1

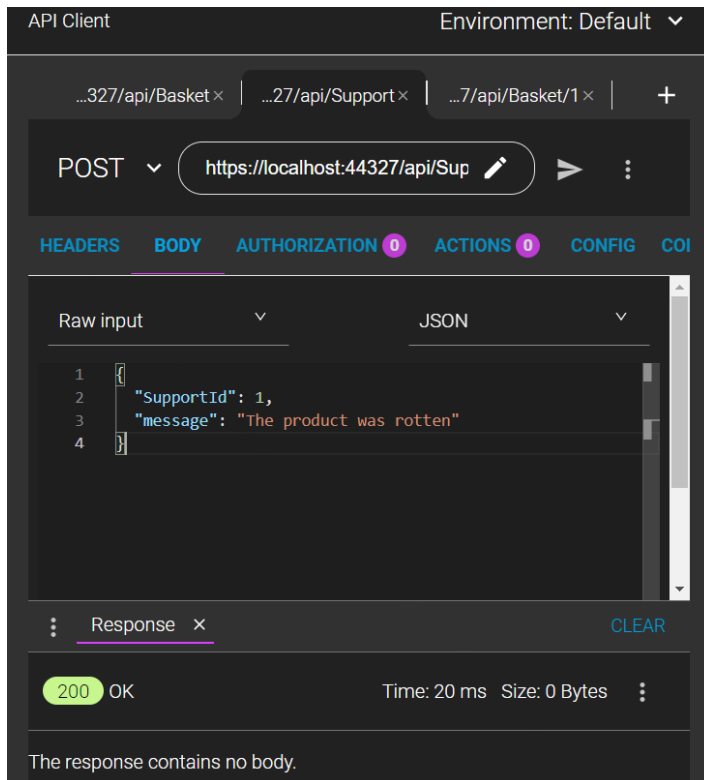
Response x **CLEAR**

200 OK Time: 10 ms Size: 0 Bytes

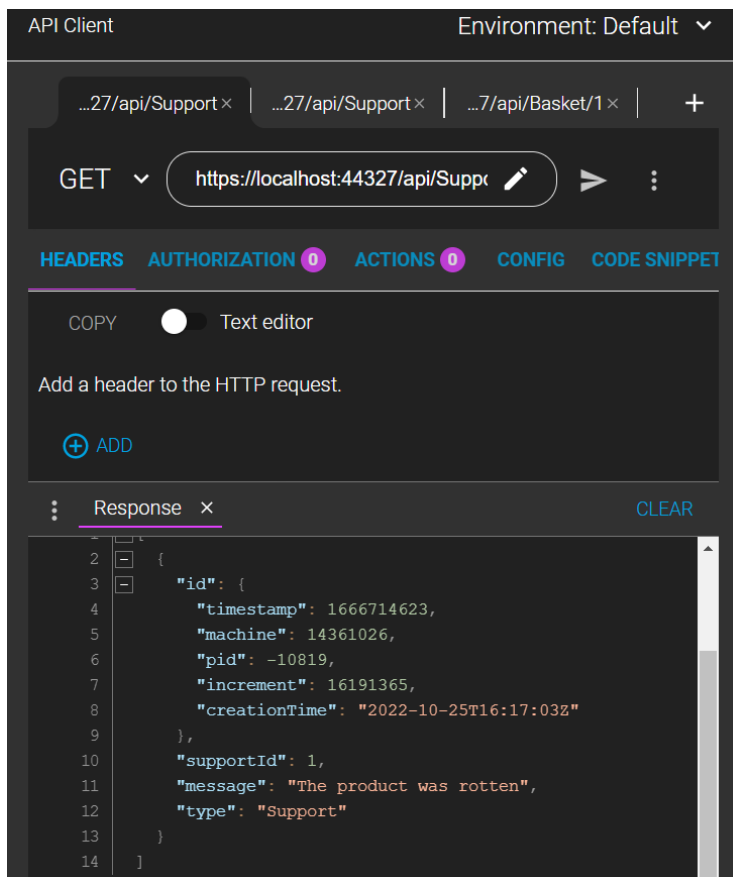
The response contains no body.

SUPPORT

Create Support



View Support



WebApp0.data 0 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

FILTER { field: 'value' } **OPTIONS** **FIND** **RESET** **REFRESH**

ADD DATA **VIEW** **JSON** **TABLE** Displaying documents 1 - 7 of 7

```
{
  "_id": ObjectId('635809ffdb21c2d5bdf70f80'),
  "Name": "Grapes",
  "ProductId": 4,
  "Price": 40.4,
  "Type": "Product"
}
```

```
{
  "_id": ObjectId('63580a08db21c2d5bdf70f81'),
  "Name": "Mango",
  "ProductId": 5,
  "Price": 50.5,
  "Type": "Product"
}
```

```
{
  "_id": ObjectId('63580a7fdb21c2d5bdf70f84'),
  "UserId": 1,
  "Name": "anna",
  "Email": "anna@email.com",
  "Type": "User"
}
```

```
{
  "_id": ObjectId('63580bffd21c2d5bdf70f85'),
  "SupportId": 1,
  "Message": "The product was rotten",
  "Type": "Support"
}
```

Delete Support

API Client Environment: Default

...27/api/Support × | ...27/api/Support × | .../api/Support/1 × | +

DELETE **URL** localhost:44327/api/Support/1 **▶**

HEADERS **BODY** **AUTHORIZATION** 0 **ACTIONS** 0 **CONFIG** **COI**

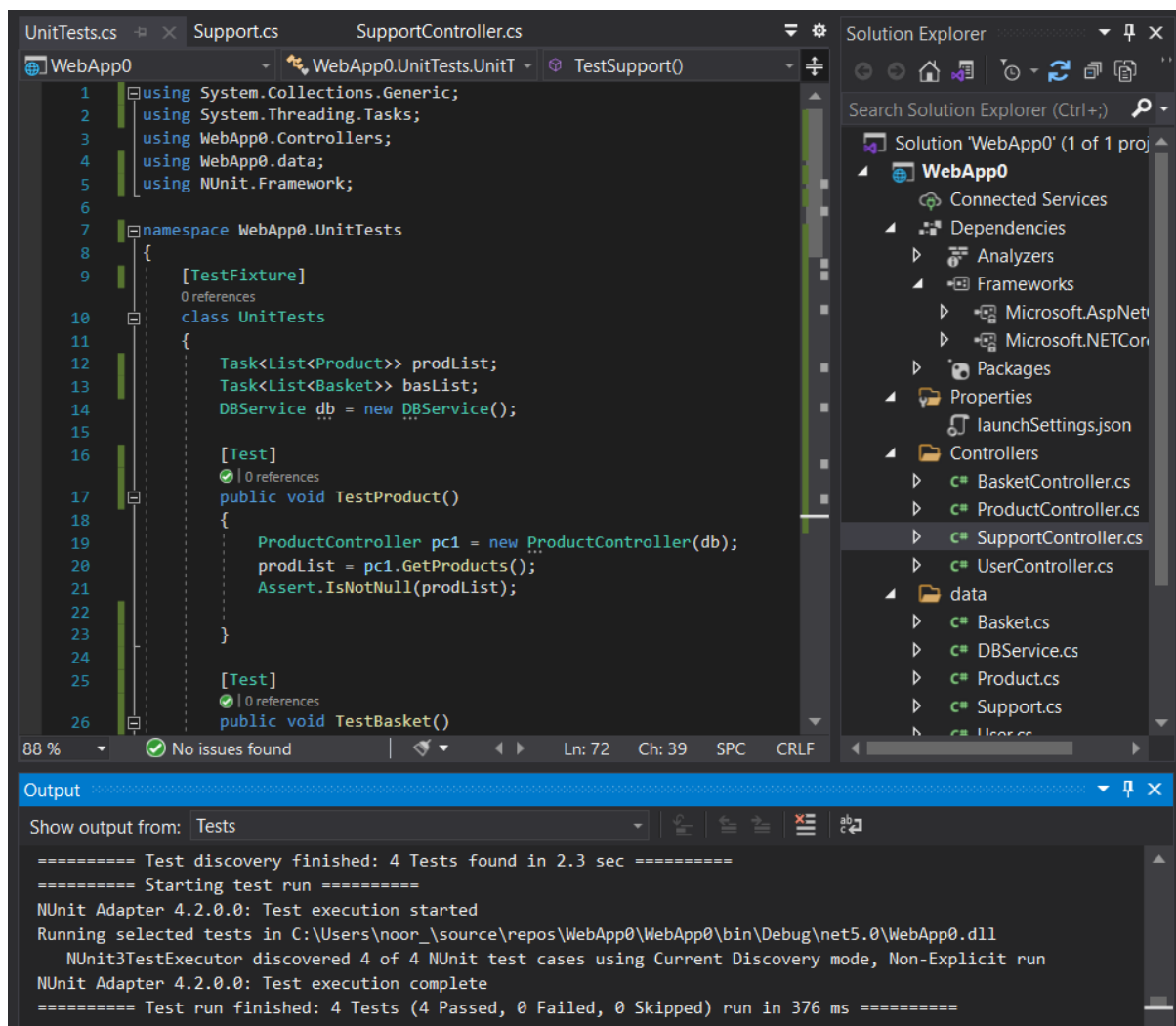
Raw input **JSON**

1

Response **CLEAR**

200 OK Time: 17 ms Size: 0 Bytes

The response contains no body.

Unit test cases for all end points of the application.

The screenshot shows the Visual Studio IDE with the 'UnitTests.cs' file open. The code contains two test methods: `TestBasket()` and `TestUser()`. `TestBasket()` creates a list of products, adds one, creates a basket, and asserts it is not null. `TestUser()` creates a user and asserts it is not null. The Solution Explorer on the right shows the project structure for 'WebApp0'. The Output window at the bottom shows the test results: 4 tests found, 4 passed, 0 failed, 0 skipped, run in 376 ms.

```
[Test]
public void TestBasket()
{
    List<int> prods = new List<int>();
    prods.Add(1);
    Basket newBas = new Basket
    {
        ProductId = prods,
        UserId = 1
    };

    BasketController pc2 = new BasketController(db);
    pc2.CreateBasket(newBas);
    basList = pc2.GetBasket();
    Assert.IsNotNull(basList);
}

[Test]
public void TestUser()
{
    Task<List<User>> userList;
    User user = new User
    {
        Name = "john",
        Email = "johnappleseed@apple.com",
    };

    UserController uc1 = new UserController(db);
    uc1.CreateUser(user);
    userList = uc1.GetUsers();
    Assert.IsNotNull(userList);
}

[Test]
public void TestSupport()
{
    Task<List<Support>> supList;
    Support sup = new Support
    {
        SupportId = 1,
        Message = "I want to leave a bigger tip"
    };

    SupportController sc1 = new SupportController(db);
    sc1.CreateSupport(sup);
    supList = sc1.GetSupport();
    Assert.IsNotNull(supList);
}
```

Output

```
===== Test discovery finished: 4 Tests found in 2.3 sec =====
===== Starting test run =====
NUnit Adapter 4.2.0.0: Test execution started
Running selected tests in C:\Users\noor_\source\repos\WebApp0\WebApp0\bin\Debug\net5.0\WebApp0.dll
NUnit3TestExecutor discovered 4 of 4 NUnit test cases using Current Discovery mode, Non-Explicit run
NUnit Adapter 4.2.0.0: Test execution complete
===== Test run finished: 4 Tests (4 Passed, 0 Failed, 0 Skipped) run in 376 ms =====
```

The screenshot shows the Visual Studio IDE with the 'UnitTests.cs' file open. The code contains two test methods: `TestSupport()` and `TestUser()`. `TestSupport()` creates a support message and asserts it is not null. `TestUser()` creates a user and asserts it is not null. The Solution Explorer on the right shows the project structure for 'WebApp0'. The Output window at the bottom shows the test results: 4 tests found, 4 passed, 0 failed, 0 skipped, run in 376 ms.

```
[Test]
public void TestSupport()
{
    Task<List<Support>> supList;
    Support sup = new Support
    {
        SupportId = 1,
        Message = "I want to leave a bigger tip"
    };

    SupportController sc1 = new SupportController(db);
    sc1.CreateSupport(sup);
    supList = sc1.GetSupport();
    Assert.IsNotNull(supList);
}

[Test]
public void TestUser()
{
    Task<List<User>> userList;
    User user = new User
    {
        Name = "john",
        Email = "johnappleseed@apple.com",
    };

    UserController uc1 = new UserController(db);
    uc1.CreateUser(user);
    userList = uc1.GetUsers();
    Assert.IsNotNull(userList);
}

[Test]
public void TestBasket()
{
    List<int> prods = new List<int>();
    prods.Add(1);
    Basket newBas = new Basket
    {
        ProductId = prods,
        UserId = 1
    };

    BasketController pc2 = new BasketController(db);
    pc2.CreateBasket(newBas);
    basList = pc2.GetBasket();
    Assert.IsNotNull(basList);
}
```

Output

```
===== Test discovery finished: 4 Tests found in 2.3 sec =====
===== Starting test run =====
NUnit Adapter 4.2.0.0: Test execution started
Running selected tests in C:\Users\noor_\source\repos\WebApp0\WebApp0\bin\Debug\net5.0\WebApp0.dll
NUnit3TestExecutor discovered 4 of 4 NUnit test cases using Current Discovery mode, Non-Explicit run
NUnit Adapter 4.2.0.0: Test execution complete
===== Test run finished: 4 Tests (4 Passed, 0 Failed, 0 Skipped) run in 376 ms =====
```