Лабораторна робота №4

Тема. Створення RESTful API

Мета. Ознайомитися з процесом створення клієнт-серверної архітектури. Навчитися створювати RESTful API для взаємодії між клієнтом і сервером. Розвинути вміння проєктувати та реалізовувати ендпойнти для типових CRUD-операцій.

ЗАВДАННЯ

Завдання 1 - Створити ASP.NET Core Minimal API.

Хід роботи:

Завдання 1

Розроблено проект за допомогою C# ASP.Net Core.

Проект реалізує CRUD операції для власного API на тему: програма для керування підписками, надано три види ресурісів:

PeopleItems – модель, яка представляє користувача з властивостями: Id – унікальний ідентифікатор, Name – ім’я користувача та Email – електронна адреса користувача,

SubscribtionItems – модель, яка представляє об’єкт підписки з властивостями: Id – унікальний ідентифікатор, OwnerId – Ідентифікатор власника підписки та Service – сервіс на якому зроблено підписку,

MessageItems – модель, яка представляє об’єкт повідомлення про підписку з властивостями: Id – унікальний ідентифікатор, Title – повідомлення, OwnerId – Ідентифікатор власника підписки та Service – сервіс на якому зроблено підписку, SubId – ідентифікатор підписки.

Лістинг:

Створено каталог Models із такими файлами:

People.cs:

```
namespace CRUDAPI.Models
{
    /// <summary>
    /// Моедль що представляє користувачів програми для управління під
писками
    /// </summary>
    public class PeopleItems
    {
        /// <summary>
        /// Унікальний ідентифікатор
        /// </summary>
        public int Id { get; set; }
        /// <summary>
        /// Ім'я або логін користувача
        /// </summary>
        public string Name { get; set; }
        /// <summary>
        /// Поштова адреса користувача
        /// </summary>
        public string Email { get; set; }
    }
}
```

Subscription.cs:

```
namespace CRUDAPI.Models
{
    /// <summary>
    /// Модель, яка описує підпску, яку може мати користувач
    /// </summary>
    public class SubscriptionItems
    {
        /// <summary>
        /// Унікальний індентифікатор підписки
        /// </summary>
        public int Id { get; set; }
         /// <summary>
        /// Ідентиікатор власника пдписки
        /// </summary>
        public int OwnerId { get; set; }
        /// <summary>
        /// Сервіс для кого виконана підпска
        /// </summary>
        public string Service { get; set; }
    }
}
```

Message.cs:

```
namespace CRUDAPI.Models
{
   /// <summary>
   /// Модель реалізує сповіщення, кі отримує користувач про підписку
   /// </summary>
   public class MessageItems
   {
      /// <summary>
      /// Унікальний ідентифікатор повідомлення
      /// </summary>
      public int Id { get; set; }
      /// <summary>
      /// Заголовок повідомлення
      /// </summary>
      public string Title { get; set; }
      /// <summary>
      /// Кому надсилаєься повідомлення
      /// </summary>
      public int OwnerId { get; set; }
      /// <summary>
      /// Індетифікатор підписки
      /// </summary>
      public int SubId { get; set; }
   }
}
```

З цьому каталозі також розміщено файл для валідації данних:

```
using CRUDAPI.Data;
using System.Text.RegularExpressions;
 namespace CRUDAPI.Models
{
   public static class Valid
   {
      const string compareField = @"^[\w-]+@[\w-]+\.\w+$";
      public static bool StringField(string field, int minLen,string nameField,
out string error)
      {
         if (string.IsNullOrEmpty(field))
         {
            error = $"Текствове поле {nameField} пусте";
            return false;
         }
         else if(field.Length < minLen){
            error = $"Занадто коротке поле {nameField}";
            return false;
         }
```

```
                error = string.Empty;
                return true;
            }
            public static bool Email(string email,out string error)
            {
                if (!Regex.IsMatch(email, compareField))
                {
                    error = "Невірна поштова адреса";
                    return false;
                }
                error = string.Empty;
                return true;
            }
            public static bool CheckOwner(int id)
            {
                var owner = Elements.peopleItems.FirstOrDefault(A => A.Id == id);
                if (owner is null)
                    return false;
                return true;
            }
            public static bool Check_Subscribe_Owner(int Subid, int OwnerId,out
string error)
            {
                var subscription = Elements.subsripptionItems.FirstOrDefault(A =>
A.Id == Subid);
                if (subscription is null)
                {
                    error = "Не знайдено вказаної підписки";
                    return false;
                }
                else if (subscription.OwnerId != OwnerId)
                {
                    error = "Неправильний власник підписки";
                    return false;
                }
                error = string.Empty;
                return true;
            }
        }
    }
```

Створенно каталог Data, в якому містяться списки із елементів класів моделей:

```
using CRUDAPI.Models;
```

```
namespace CRUDAPI.Data
{
    public static class Elements
    {
public static List<PeopleItems> peopleItems = new List<PeopleItems>()
    {
      new PeopleItems() {Id = 1, Name = "Kate", Email = "Kate@Email.com" },
      new PeopleItems() {Id = 2, Name = "John", Email = "John@Email.com" },
      new PeopleItems() {Id = 3, Name = "Jane", Email = "Jane@Email.com" },
      new PeopleItems() {Id = 4, Name = "Steve", Email = "Steve@Email.com"
},
      new PeopleItems() {Id = 5, Name = "Alex", Email = "Alex@Email.com" }
    };
public static List<SubscriptionItems> subsripptionItems = new List<SubscriptionIt
ems>()
    {
      new SubscriptionItems(){Id = 1, OwnerId = 1, Service = "Netflix"},
      new SubscriptionItems(){Id = 2, OwnerId = 1, Service = "Xbox"},
      new SubscriptionItems(){Id = 3, OwnerId = 1, Service = "Amazon"},
       new SubscriptionItems(){Id = 4, OwnerId = 2, Service = "Steam"},
      new SubscriptionItems(){Id = 5, OwnerId = 2, Service = "Google"},
      new SubscriptionItems(){Id = 6, OwnerId = 2, Service = "Amazon"},
       new SubscriptionItems(){Id = 7, OwnerId = 3, Service = "YouTube"},
      new SubscriptionItems(){Id = 8, OwnerId = 3, Service = "Netflix"},
       new SubscriptionItems(){Id = 9, OwnerId = 4, Service = "YouTube"},
      new SubscriptionItems(){Id = 10, OwnerId = 4, Service = "Netflix"},
      new SubscriptionItems(){Id = 11, OwnerId = 4, Service = "Megogo"},
      new SubscriptionItems(){Id = 12, OwnerId = 4, Service = "AppleFilms"},
       new SubscriptionItems(){Id = 13, OwnerId = 5, Service = "AppleSerice"},
      new SubscriptionItems(){Id = 14, OwnerId = 5, Service = "Xbox"},
      new SubscriptionItems(){Id = 15, OwnerId = 5, Service = "AppleFilms"},
    };
public static List<MessageItems> messageItems = new List<MessageItems>()
            {
              new MessageItems(){Id = 1,Title = "До оплати три дні", OwnerId =
 1, SubId = 1 },
              new MessageItems(){Id = 2,Title = "До оплати два дні", OwnerId =
 1, SubId = 1 },
              new MessageItems(){Id = 3,Title = "До оплати один день", OwnerI
d = 1, SubId = 1 },
```

```
                new MessageItems(){Id = 4,Title = "Підписка прострочена", Own
erId = 2, SubId = 5 },
                new MessageItems(){Id = 5,Title = "Підписку скасовано", OwnerI
d = 3, SubId = 7 },
                new MessageItems(){Id = 6,Title = "Підписку просрочена", Owner
Id = 4, SubId = 9 },
                new MessageItems(){Id = 7,Title = "Підпску продовжено", OwnerI
d = 4, SubId = 9 },
                new MessageItems(){Id = 8,Title = "Підписка прострочена", Owne
rId = 4, SubId = 11 },
                new MessageItems(){Id = 9,Title = "Підписку скасовано", OwnerId
 = 4, SubId = 11 },
                new MessageItems(){Id = 10,Title = "До оплаи підписки тиждень",
 OwnerId = 5, SubId = 15 },
            };
        }
    }
```

Створено каталог EndPoints із класами для реалізації endpoints моделей:

PeopleEndPoints.cs:

```
using CRUDAPI.Models;
using CRUDAPI.Data;
 namespace CRUDAPI.EndPoints
{
    public static class PeopleEndpoints
    {
      ///<summary>
    ///Реєструє всі маршрути до ресурсу People
    ///</summary>
    ///<param name="app">Екземпляр<see cref="WebApplication"/>.</para
m>
    public static void mapPeople(this WebApplication app)
        {
         app.MapGet("/peoples/{id:int}", (int id) =>
          {
```

```
                var peopleItem = Elements.peopleItems.FirstOrDefault(A => A.Id
== id);
                 return peopleItem is null ? Results.NotFound() : Results.Ok(peopl
eItem);
             })
            .WithSummary("Отримати корисувача за Id")
            .Produces<PeopleItems>(StatusCodes.Status200OK)
            .Produces(StatusCodes.Status404NotFound)
            .WithName("GETPeople")
            .WithDescription("Отримати користувача за Id")
            .WithOpenApi();
           app.MapGet("/peoples", (string? name, string? email) =>
            {
               IEnumerable<PeopleItems> items = Elements.peopleItems;
                if (name is not null)
                   items = items.Where(A => A.Name == name);
               if (email is not null)
                   items = items.Where(A => A.Email == email);
                return items.Count() != 0 ? Results.Ok(items) : Results.NotFound(
);
            })
            .WithSummary("Отримати всіх корисувачів або конкретизувати з
а допомогою фільтрів")
            .Produces<PeopleItems>(StatusCodes.Status200OK)
            .Produces(StatusCodes.Status404NotFound)
            .WithName("GETPeoples")
            .WithDescription("Отримання всіх користувачів або конкреного з
а фільром")
            .WithOpenApi();
          app.MapPost("/peoples", (PeopleItems peopleItem) =>
         {
            if (!Valid.StringField(peopleItem.Name, 5, "Name", out var error1))
               return Results.BadRequest(error1);
             if(!Valid.Email(peopleItem.Email,out var error2))
               return Results.BadRequest(error2);
             peopleItem.Id = Elements.peopleItems.Max(A => A.Id) + 1;
             Elements.peopleItems.Add(peopleItem);
            return Results.Created($"/peoples/{peopleItem.Id}", peopleItem);
         })
            .WithSummary("Надіслати конкретного користувача в список")
            .Produces<PeopleItems>(StatusCodes.Status201Created)
            .Produces(StatusCodes.Status400BadRequest)
            .WithName("POSTPeople")
            .WithDescription("Відправити конкретного користувача")
            .WithOpenApi();
```

```
            app.MapPut("/peoples/{id:int}", (int id, PeopleItems updatePeople) =>
            {
               var people = Elements.peopleItems.Find(A => A.Id == id);
               if (people is null)
                  return Results.NotFound();
               if (!Valid.StringField(updatePeople.Name, 5, "Name", out var error1
))
                  return Results.BadRequest(error1);
               if (!Valid.Email(updatePeople.Email, out var error2))
                  return Results.BadRequest(error2);
               people.Name = updatePeople.Name;
               people.Email = updatePeople.Email;
               return Results.Ok(people);
            })
               .WithSummary("Оновити повнсітю конкретного користувача")
               .Produces<PeopleItems>(StatusCodes.Status200OK)
               .Produces(StatusCodes.Status404NotFound)
               .Produces(StatusCodes.Status400BadRequest)
               .WithName("PUTPeople")
               .WithDescription("Оновити повністю конкреного кристувача")
               .WithOpenApi();
            app.MapPatch("/peoples/{id:int}", (int id, PeopleItems updatePeople) =
>       {
               var people = Elements.peopleItems.Find(A => A.Id == id);
               if (people is null)
                  return Results.NotFound();
               if (updatePeople.Name is not null)
               {
                 if (!Valid.StringField(updatePeople.Name, 5, "Name", out var error
1))
                     return Results.BadRequest(error1);
                  people.Name = updatePeople.Name;
               }
               if (updatePeople.Email is not null)
               {
                  if (!Valid.Email(updatePeople.Email, out var error2))
                     return Results.BadRequest(error2);
                  people.Email = updatePeople.Email;
               }
               return Results.Ok(people);
            })
               .WithSummary("Оновити частково конкретного користувача")
               .Produces<PeopleItems>(StatusCodes.Status200OK)
               .Produces(StatusCodes.Status404NotFound)
               .Produces(StatusCodes.Status400BadRequest)
```

```
            .WithName("PATCHPeople")
            .WithDescription("Оновити частково конкреного кристувача")
            .WithOpenApi();
        app.MapDelete("/peoples/{id:int}", (int id) =>
        {
            var people = Elements.peopleItems.FirstOrDefault(A => A.Id == id);
            if (people is null)
                return Results.NotFound();
            Elements.peopleItems.Remove(people);
            return Results.NoContent();
        })
            .WithSummary("Видалити конкретного користувача")
            .Produces(StatusCodes.Status204NoContent)
            .Produces(StatusCodes.Status404NotFound)
            .WithName("DELETEPeople")
            .WithDescription("Видалити конкретного користувача")
            .WithOpenApi();
    }
    }
}
SuscriptionsEndPoints.cs:
using CRUDAPI.Models;
using CRUDAPI.Data;
 namespace CRUDAPI.EndPoints
{
    public static class SubscriptionsEndPoints
    {
        ///<summary>
        ///Реєструє всі маршрути до ресурсу Subscriptions
        ///</summary>
        ///<param name="app">Екземпляр<see cref="WebApplication"/>.</pa
ram>
        public static void mapSubscriptions(this WebApplication app)
        {
            app.MapGet("/subs/{id:int}", (int id) =>
            {
                var subItem = Elements.subsripptionItems.FirstOrDefault(A => A.
Id == id);
                return subItem is null ? Results.NotFound() : Results.Ok(subItem)
;
            })
                .WithSummary("Отримати підписку")
                .Produces<SubscriptionItems>(StatusCodes.Status200OK)
                .Produces(StatusCodes.Status404NotFound)
                .WithName("GETSub")
```

```
                .WithDescription("Отримати конкретну підписку за Id")
                .WithOpenApi();
            app.MapGet("/subs", (int? ownerId, string? service) =>
            {
                IEnumerable<SubscriptionItems> items = Elements.subsripptionIt
ems;
                 if (ownerId is not null)
                    items = items.Where(A => A.OwnerId == ownerId);
                if (service is not null)
                    items = items.Where(A => A.Service == service);

                return items.Count() != 0 ? Results.Ok(items) : Results.NotFound(
);
            })
                .WithSummary("Отримати всі підписки або конкретні")
                .Produces<SubscriptionItems>(StatusCodes.Status200OK)
                .Produces(StatusCodes.Status404NotFound)
                .WithName("GETSubs")
                .WithDescription("Отримати всі підписки або конкретні за доп
омогою фільтрів")
                .WithOpenApi();
            app.MapPost("/subs", (SubscriptionItems subscription) =>
            {
                if (!Valid.StringField(subscription.Service, 3, "Service", out var er
ror))
                    return Results.BadRequest(error);
                 if (!Valid.CheckOwner(subscription.OwnerId))
                    return Results.BadRequest("Немає вказаного 'ownerId'");
                 subscription.Id = Elements.subsripptionItems.Max(A => A.Id) +
1;
                 Elements.subsripptionItems.Add(subscription);
                 return Results.Created($"/subs/{subscription.Id}", subscription);
             })
                .WithSummary("Відправити підписку")
                .Produces<SubscriptionItems>(StatusCodes.Status201Created)
                .Produces(StatusCodes.Status400BadRequest)
                .WithName("POSTSub")
                .WithDescription("Відправити конкретну підписку")
                .WithOpenApi();
            app.MapPut("/subs/{id:int}", (int id, SubscriptionItems updateSub)
=>
            {
                var subscription = Elements.subsripptionItems.Find(A => A.Id ==
id);
                 if (subscription is null)
```

```
                    return Results.NotFound();
                if (!Valid.CheckOwner(subscription.OwnerId))
                    return Results.BadRequest("Немає вказаного 'ownerId'");
                if (!Valid.StringField(updateSub.Service, 3, "Service", out var err
or))
                    return Results.BadRequest(error);
                subscription.OwnerId = updateSub.OwnerId;
                subscription.Service = updateSub.Service;
                return Results.Ok(subscription);
            })
                .WithSummary("Оновити повністю підписку")
                .Produces<SubscriptionItems>(StatusCodes.Status200OK)
                .Produces(StatusCodes.Status404NotFound)
                .Produces(StatusCodes.Status400BadRequest)
                .WithName("PUTSub")
                .WithDescription("Оновити повністю конкретну підписку за Id
")
                .WithOpenApi();

            app.MapPatch("/subs/{id:int}", (int id, SubscriptionItems updateSub)
 =>
            {
                var subscription = Elements.subsripptionItems.Find(A => A.Id ==
id);
                if (subscription is null)
                    return Results.NotFound();
                if (!Valid.StringField(updateSub.Service, 3, "Service", out var err
or))
                    return Results.BadRequest(error);
                if (updateSub.OwnerId != 0)
                {
                    if (!Valid.CheckOwner(subscription.OwnerId))
                        return Results.BadRequest("Немає вказаного 'ownerId'");
                    subscription.OwnerId = updateSub.OwnerId;
                }
                if(updateSub.Service is not null)
                    subscription.Service = updateSub.Service;
                return Results.Ok(subscription);
            })
                .WithSummary("Оновити частково підписку")
                .Produces<SubscriptionItems>(StatusCodes.Status200OK)
                .Produces(StatusCodes.Status404NotFound)
                .Produces(StatusCodes.Status400BadRequest)
                .WithName("PATCHSub")
```

```
                .WithDescription("Оновити частково конкретну підписку за Id"
)
                .WithOpenApi();
               app.MapDelete("/subs/{id:int}", (int id) =>
               {
                  var sub = Elements.subsripptionItems.FirstOrDefault(A => A.Id =
= id);
               if (sub is null)
                     return Results.NotFound();
                  Elements.subsripptionItems.Remove(sub);
                 return Results.NoContent();
               })
                  .WithSummary("Видалити підписку")
                  .Produces(StatusCodes.Status204NoContent)
                  .Produces(StatusCodes.Status404NotFound)
                  .WithName("DELETESub")
                  .WithDescription("Видалити конкретну підписку за Id")
                  .WithOpenApi();
            }
         }
      }
      MessageEndPoints.cs:
      using CRUDAPI.Data;
      using CRUDAPI.Models;
       namespace CRUDAPI.EndPoints
      {
         public static class MessageEndPoints
         {
             ///<summary>
            ///Реєструє всі маршрути до ресурсу Subscriptions
            ///</summary>
            ///<param name="app">Екземпляр<see cref="WebApplication"/>.</pa
ram>
            public static void mapMessages(this WebApplication app)
            {
               app.MapGet("/messages/{id:int}", (int id) =>
               {
                  var mItem = Elements.messageItems.FirstOrDefault(A => A.Id ==
 id);
                   return mItem is null ? Results.NotFound() : Results.Ok(mItem);
                })
                  .WithSummary("Отримати повідомлення")
                  .Produces<MessageItems>(StatusCodes.Status200OK)
                  .Produces(StatusCodes.Status404NotFound)
                  .WithName("GETmessage")
```

```
                .WithDescription("Отримати конкретне повідомлення за Id")
                .WithOpenApi();
            app.MapGet("/messages", (int? ownerId, int? subId,string? title) =>
            {
                IEnumerable<MessageItems> items = Elements.messageItems;
                if (ownerId is not null)
                    items = items.Where(A => A.OwnerId == ownerId);
                if (subId is not null)
                    items = items.Where(A => A.SubId == subId);
                if(title is not null)
                    items = items.Where(A => A.Title == title);
                return items.Count() != 0 ? Results.Ok(items) : Results.NotFound(
);
            })
                .WithSummary("Отримати всі повідомлення або конкретні")
                .Produces<MessageItems>(StatusCodes.Status200OK)
                .Produces(StatusCodes.Status404NotFound)
                .WithName("GETmessages")
                .WithDescription("Отримати всі повідомлення або використати
 фільри для конкретизації")
                .WithOpenApi();
            app.MapPost("/messages", (MessageItems message) =>
            {
                if (!Valid.StringField(message.Title, 10, "Title", out var error))
                    return Results.BadRequest(error);
                            if(!Valid.Check_Subscribe_Owner(message.SubId,mess
age.OwnerId,out string error1))
                  return Results.BadRequest(error1);
                message.Id = Elements.messageItems.Max(A => A.Id) + 1;
                Elements.messageItems.Add(message);
                return Results.Created($"/messages/{message.Id}", message);
            })
                .WithSummary("Зафіксувати повідомлення в колекції")
                .Produces<MessageItems>(StatusCodes.Status201Created)
                .Produces(StatusCodes.Status404NotFound)
                .WithName("POSTmessage")
                .WithDescription("Відправити повідомлення в колекцію")
                .WithOpenApi();
            app.MapPut("/messages/{id:int}", (int id, MessageItems updateM) =
>
            {
                var m = Elements.messageItems.Find(A => A.Id == id);
                if (m is null)
                    return Results.NotFound();
```

```
                    if (!Valid.Check_Subscribe_Owner(updateM.SubId, updateM.Ow
nerId, out string error1))
                        return Results.BadRequest(error1);
                    if (!Valid.StringField(updateM.Title, 10, "Title", out var error))
                        return Results.BadRequest(error);
                    m.Title = updateM.Title;
                   m.OwnerId = updateM.OwnerId;
                   m.SubId = updateM.SubId;
                    return Results.Ok(m);
                })
                   .WithSummary("Оновити повністю повідомлення")
                   .Produces<MessageItems>(StatusCodes.Status200OK)
                   .Produces(StatusCodes.Status404NotFound)
                   .Produces(StatusCodes.Status400BadRequest)
                   .WithName("PUTmessage")
                   .WithDescription("Оновити повністю повідомлення в колекції з
a Id")
                   .WithOpenApi();
                app.MapPatch("/messages/{id:int}", (int id, MessageItems updateM)
 =>
                {
                   var m = Elements.messageItems.Find(A => A.Id == id);
                    if (m is null)
                       return Results.NotFound();
                    int OId = updateM.OwnerId != 0? updateM.OwnerId : m.OwnerId
;
                   int SId = updateM.SubId != 0? updateM.SubId : m.SubId;
                    if (!Valid.Check_Subscribe_Owner(SId, OId, out string error1))
                       return Results.BadRequest(error1);
                    if (updateM.OwnerId != 0)
                       m.OwnerId = updateM.OwnerId;
                   if(updateM.SubId != 0)
                       m.SubId = updateM.SubId;
                   if (updateM.Title is not null)
                   {
                       if (!Valid.StringField(updateM.Title, 10, "Title", out var error))
                           return Results.BadRequest(error);
                       m.Title = updateM.Title;
                   }
                   return Results.Ok(m);
                })
                   .WithSummary("Оновити частково повідомлення")
                   .Produces<MessageItems>(StatusCodes.Status200OK)
                   .Produces(StatusCodes.Status404NotFound)
                   .Produces(StatusCodes.Status400BadRequest)
```

```
                .WithName("PATCHmessage")
                .WithDescription("Оновити частково повідомлення в колекції з
a Id")
                .WithOpenApi();
            app.MapDelete("/messages/{id:int}", (int id) =>
            {
                var m = Elements.messageItems.FirstOrDefault(A => A.Id == id);
                if (m is null)
                    return Results.NotFound();
                Elements.messageItems.Remove(m);
                return Results.NoContent();
            })
                .WithSummary("Видалити повідомлення")
                .Produces(StatusCodes.Status204NoContent)
                .Produces(StatusCodes.Status404NotFound)
                .WithName("DELETEmessage")
                .WithDescription("Видалити повідомлення з колекції за Id")
                .WithOpenApi();
        }
    }
}
```

Підключенно в program.cs всі endpoints:

```
using CRUDAPI.EndPoints;
 var builder = WebApplication.CreateBuilder(args);
builder.Services.AddEndpointsApiExplorer();
 builder.Services.AddSwaggerGen(oprions =>
{
    var xmlFile = $"{System.Reflection.Assembly.GetExecutingAssembly().
GetName().Name}.xml";
    var xmlPath = Path.Combine(AppContext.BaseDirectory, xmlFile);
    oprions.IncludeXmlComments(xmlPath);
});
var app = builder.Build();
app.mapPeople();
app.mapSubscriptions();
app.mapMessages();
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}
app.Run();
```

Протестовано роботу програму за допомогою swagger:

Запит1

Request URL

```
https://localhost:7211/peoples/1
```

Server response

Code	Details
200	Response body

```
{
  "id": 1,
  "name": "Kate",
  "email": "Kate@Email.com"
}
```

Download

Response headers

```
content-type: application/json; charset=utf-8
date: Wed,08 Oct 2025 09:14:29 GMT
server: Kestrel
```

Рисунок 1 – Отримання користувача за Id

Запит 2

Curl

```
curl -X 'GET' \
  'https://localhost:7211/peoples/10' \
  -H 'accept: application/json'
```

Request URL

```
https://localhost:7211/peoples/10
```

Server response

Code	Details
404	Error: response status is 404

Response headers

```
content-length: 0
date: Wed,08 Oct 2025 09:17:56 GMT
server: Kestrel
```

Рисунок 2 – Отримання користувача за відсутнім Id

Запит 3

Code	Details
200	Response body

```
{
  "id": 1,
  "name": "NewUser",
  "email": "NewEmail@gmail.com"
}
```

Download

Response headers

```
content-type: application/json; charset=utf-8
date: Wed,08 Oct 2025 09:19:21 GMT
server: Kestrel
```

Responses

Рисунок 3 – Коректне оновлення користувача

Запит 4

Request URL

```
https://localhost:7211/peoples/1
```

Server response

Code	Details
400	Error: response status is 400

Response body

```
"Текствове поле Name пусте"
```

Download

Response headers

```
content-type: application/json; charset=utf-8
date: Wed,08 Oct 2025 09:20:59 GMT
server: Kestrel
```

Рисунок 4 – Помилка під час оновлення

Запит5

Request URL

```
https://localhost:7211/peoples/1
```

Server response

Code	Details
400	Error: response status is 400

Response body

```
"Невірна поштова адреса"
```

Download

Response headers

```
content-type: application/json; charset=utf-8
date: Wed,08 Oct 2025 09:22:11 GMT
server: Kestrel
```

Рисунок 5 – Помилка валідації – невірна електронна адреса

Запит 6

Request URL

```
https://localhost:7211/peoples/3
```

Server response

Code	Details
200	

Response body

```
{
  "id": 3,
  "name": "Jane",
  "email": "NewEmail@gmail.com"
}
```

Download

Response headers

```
content-type: application/json; charset=utf-8
date: Wed,08 Oct 2025 09:24:25 GMT
server: Kestrel
```

Рисунок 6 – Часткове оновлення користувача

Запит 7

Request URL

```
https://localhost:7211/peoples/1
```

Server response

Code	Details
204	

Response headers

```
date: Wed,08 Oct 2025 09:25:49 GMT
server: Kestrel
```

Рисунок 7 – Видалення користувача

Запит 8

Request URL

```
https://localhost:7211/peoples
```

Server response

Code	Details
200	

Response body

```
[
  {
    "id": 2,
    "name": "John",
    "email": "John@Email.com"
  },
  {
    "id": 3,
    "name": "Jane",
    "email": "NewEmail@gmail.com"
  },
  {
    "id": 4,
    "name": "Steve",
    "email": "Steve@Email.com"
  },
  {
    "id": 5,
    "name": "Alex",
    "email": "Alex@Email.com"
  }
]
```

Download

Response headers

```
content-type: application/json; charset=utf-8
date: Wed,08 Oct 2025 09:26:36 GMT
server: Kestrel
```

Рисунок 8 – Отримано список всіх користувачів

Запит 9

Рисунок 9 – Отримано всіх користувачів

Запит 10

Рисунок 10 – Отримано конкретного користувача

Запит 11

Рисунок 11 – Користувача не знайдено

Запит 12

Рисунок 12 – Створення нового користувача

Запит 13

Рисунок 13 – Спроба додавання користувача з невалідною адресою

Запит 14

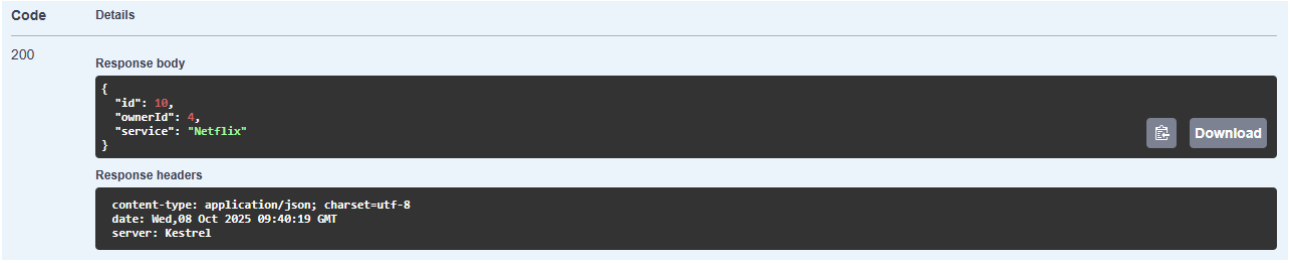

Рисунок 14 – Отримання конкретної підписки

Запит 15

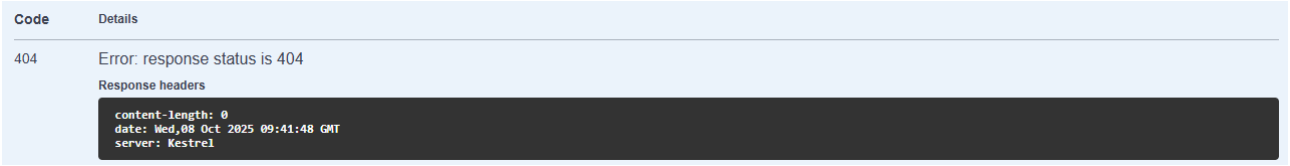

Рисунок 15 – спроба отримати неіснуючу підписку

Запит 16

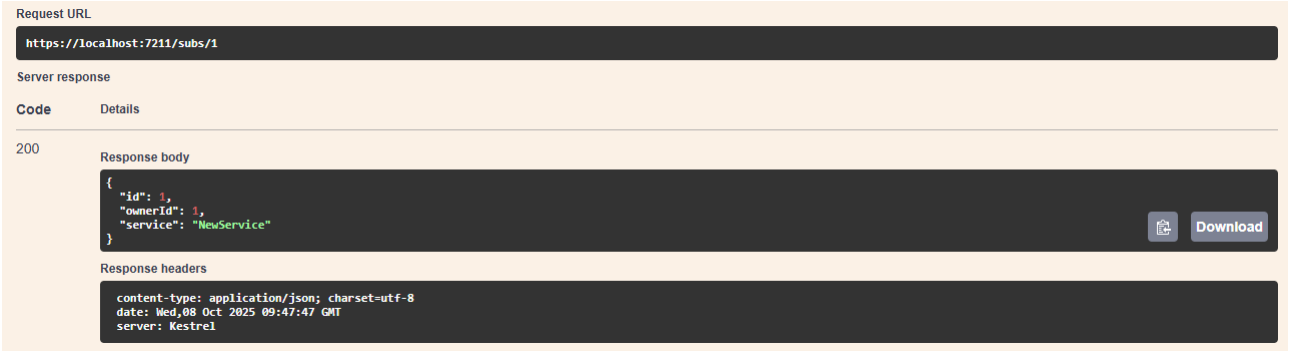

Рисунок 16 - Оновлення підписки

Запит 17

Рисунок 17 – Оновлення підписки з Id користувача, якого не існує

Запит 18

Рисунок 18 – Часткове оновлення користувача

Запит 19

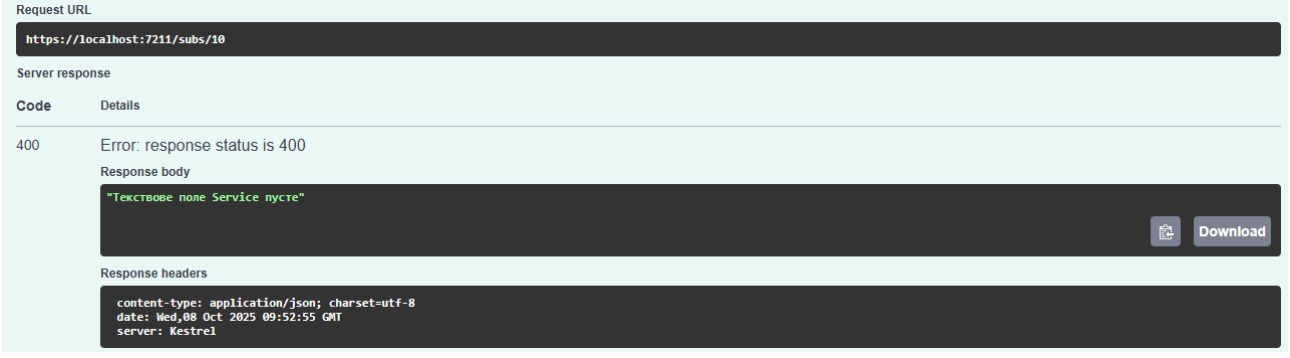

Рисунок 19 – Спроба оновлення поля підписки на пусте

Запит 20

Рисунок 20 – Видалення підписки

Запит 21

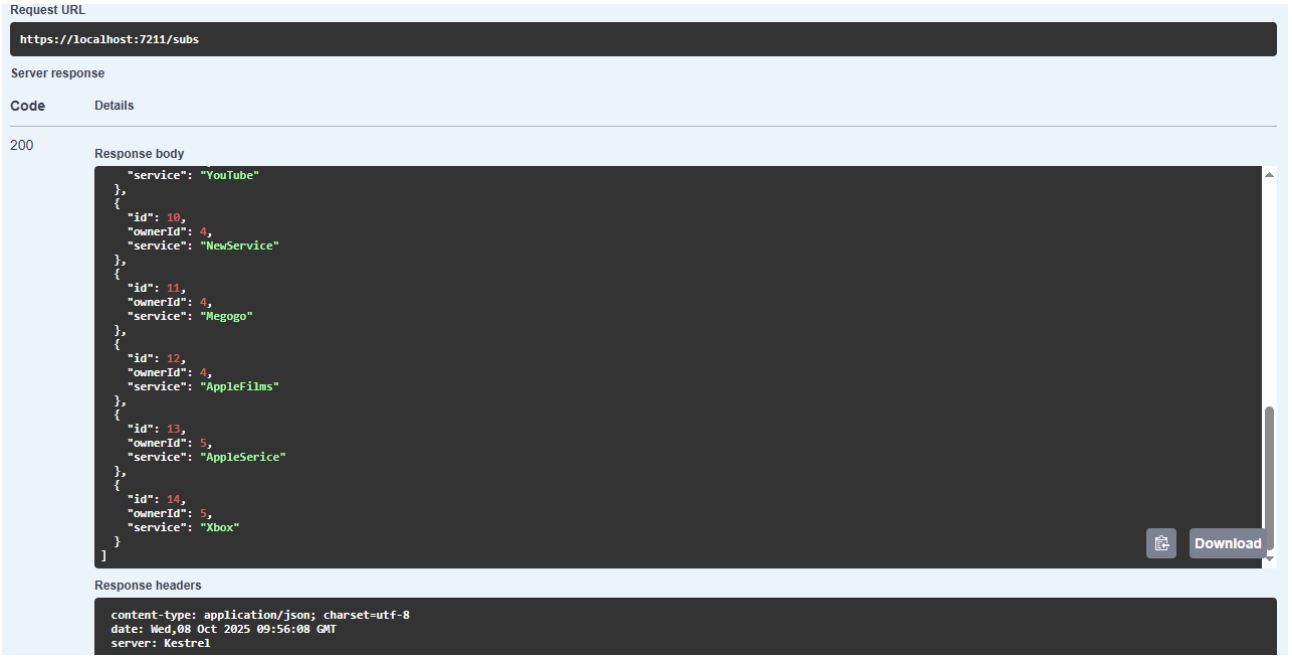

Рисунок 21 – Отримання списку всіх підписок

Запит 22

Рисунок 22 – Отримання підписки за вказаними полями

Запит 23

Рисунок 23 – Створення нової підписки

Запит 24

Рисунок 24 – Спроба додавання нової підписки з неіснуючм користувачем

Запит 25

Рисунок 25 – Спроба створення підписки з невалідним сервісом

Запит 26

Рисунок 26 – Отримання конкретного повідомлення

Запит 27

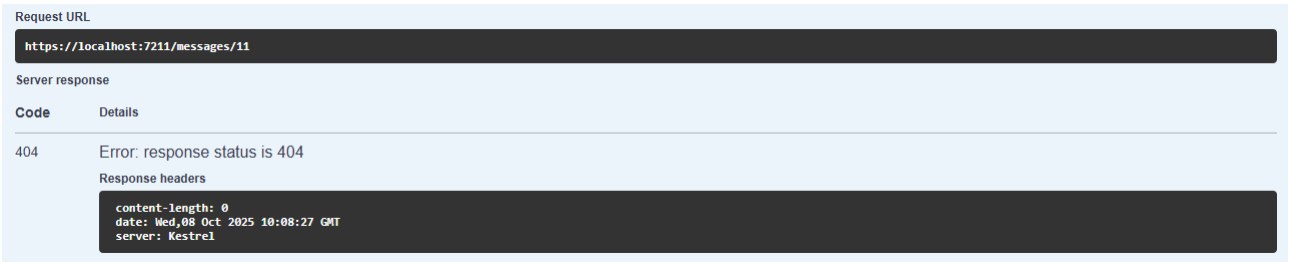

Рисунок 27 – Спроба отримання неіснуючої підписки

Запит 28

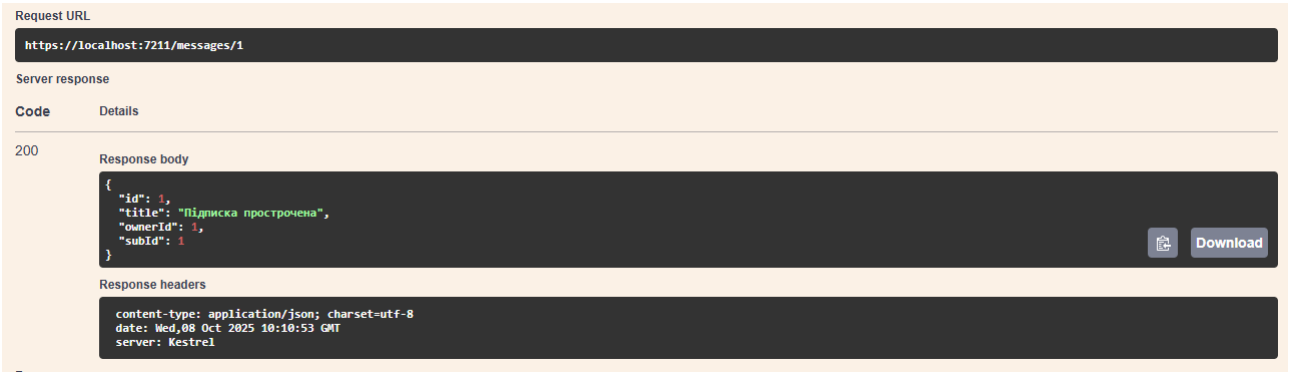

Рисунок 28 – Оновлення вмісту повідомлення

Запит 29

Рисунок 29 – Спроба відправит підписнику нагадування не про те повідолення

Запит 30

Рисунок 30 – Оновлення підписки в користувача

Запит 31

Рисунок 31 – Спроба оновлення невірної підписки користувача

Запит 32

Рисунок 32 – Видалення повідомленн про підписку

Запит 33

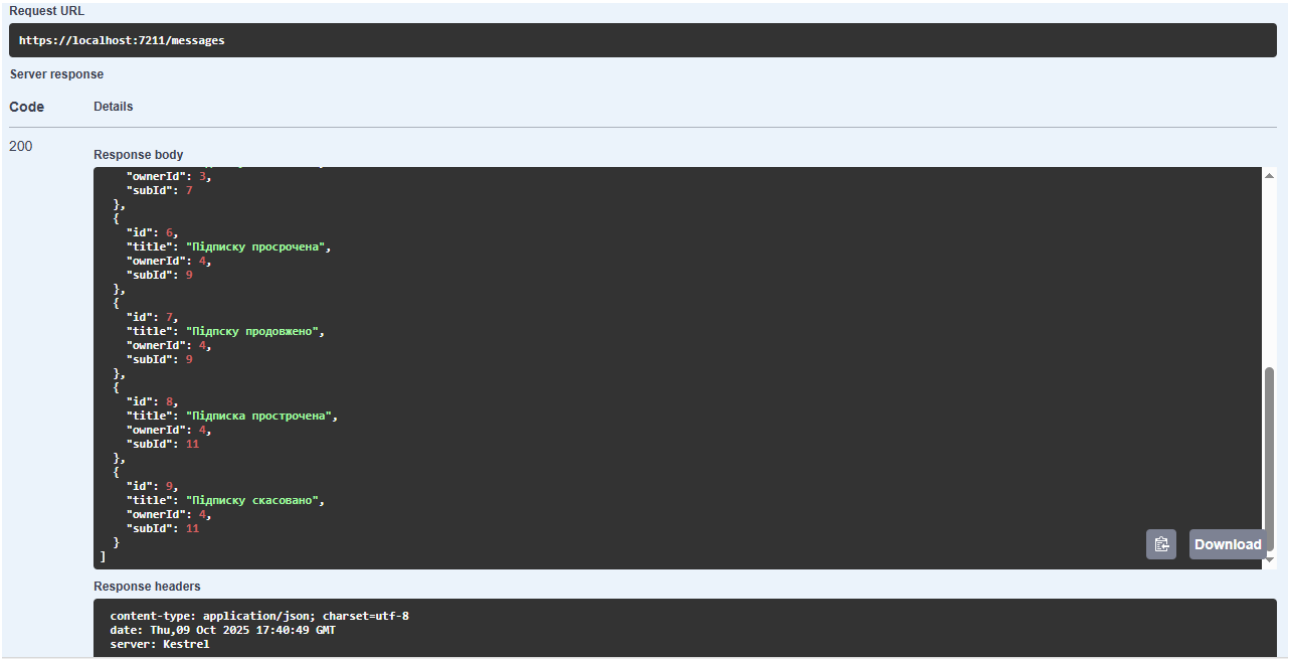

Рисунок 33 -Отримання списку усіх повідомлень

Запит 34

Рисунок 34 – Отримання всіх повідомлень до користувача

Запит 35

Рисунок 35 – Отримання всіх повідомлень про конретну підписку

Запит 36

Рисунок 36 – Спроба відправити пусте повідомлення

Запит 37

Рисунок 37 – Додано повідомлення для користувача та його підписки

Запит 38

Рисунок 38 – Спроба відправити повідомленн користувачу не про його підписку