**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.**

**Завдання 2 - Створити ASP.NET Core Web 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 підписками

    /// </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<SubscriptionItems>()

                {

                    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", OwnerId = 1, SubId = 1 },

            new MessageItems(){Id = 4,Title = "Підписка прострочена", OwnerId = 2, SubId = 5 },

            new MessageItems(){Id = 5,Title = "Підписку скасовано", OwnerId = 3, SubId = 7 },

            new MessageItems(){Id = 6,Title = "Підписку просрочена", OwnerId = 4, SubId = 9 },

            new MessageItems(){Id = 7,Title = "Підпску продовжено", OwnerId = 4, SubId = 9 },

            new MessageItems(){Id = 8,Title = "Підписка прострочена", OwnerId = 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"/>.</param>

    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(peopleItem);

            })

            .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 за допомогою фільтрів")

            .Produces<PeopleItems>(StatusCodes.Status200OK)

            .Produces(StatusCodes.Status404NotFound)

            .WithName("GETPeoples")

            .WithDescription("Отримання всіх користувачів або конкреного за 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 error1))

                    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"/>.</param>

        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.subsripptionItems;

                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 допомогою фільтрів")

                .WithOpenApi();

            app.MapPost("/subs", (SubscriptionItems subscription) =>

            {

                if (!Valid.StringField(subscription.Service, 3, "Service", out var error))

                    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 error))

                    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 error))

                    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"/>.</param>

        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,message.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.OwnerId, 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 за 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 за 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

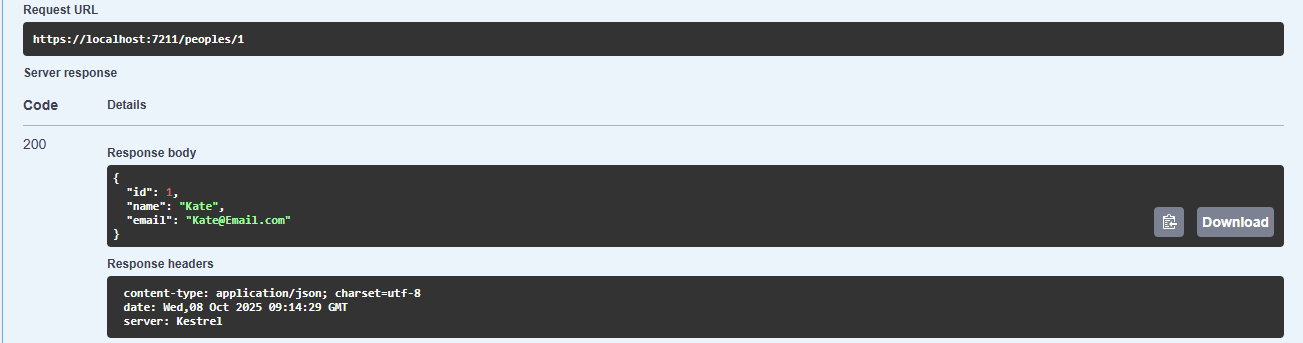
****

Рисунок 1 – Отримання користувача за Id

Запит 2

****

Рисунок 2 – Отримання користувача за відсутнім Id

Запит 3

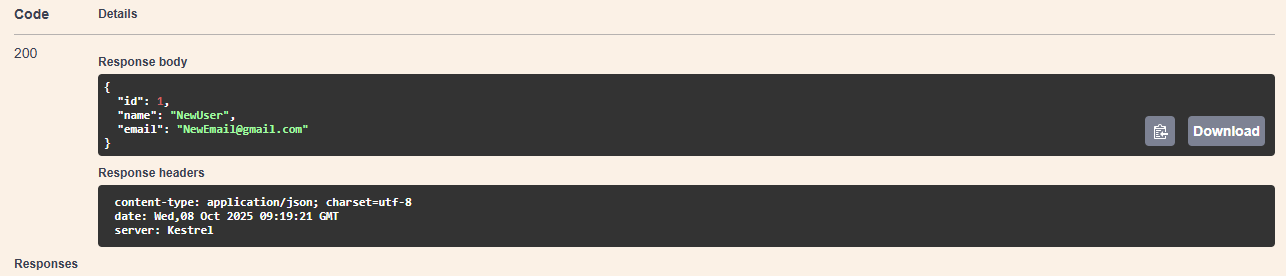


Рисунок 3 – Коректне оновлення користувача

Запит 4

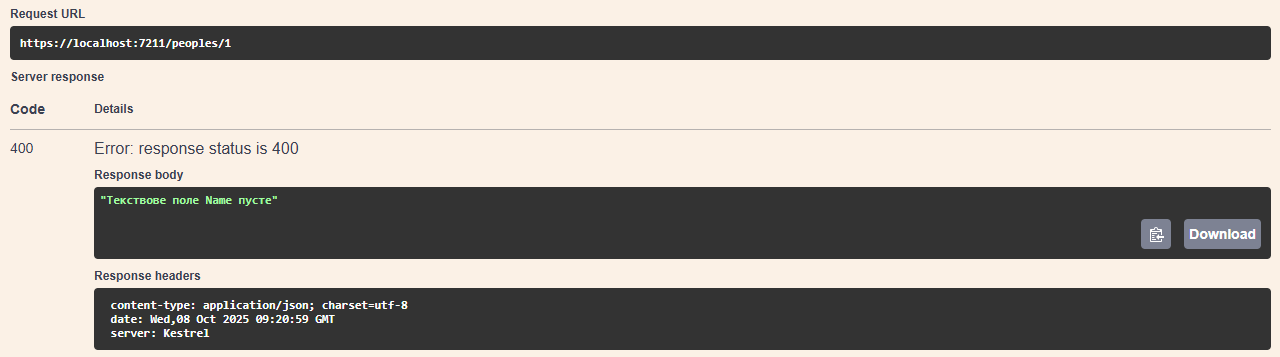
****

Рисунок 4 – Помилка під час оновлення

Запит5

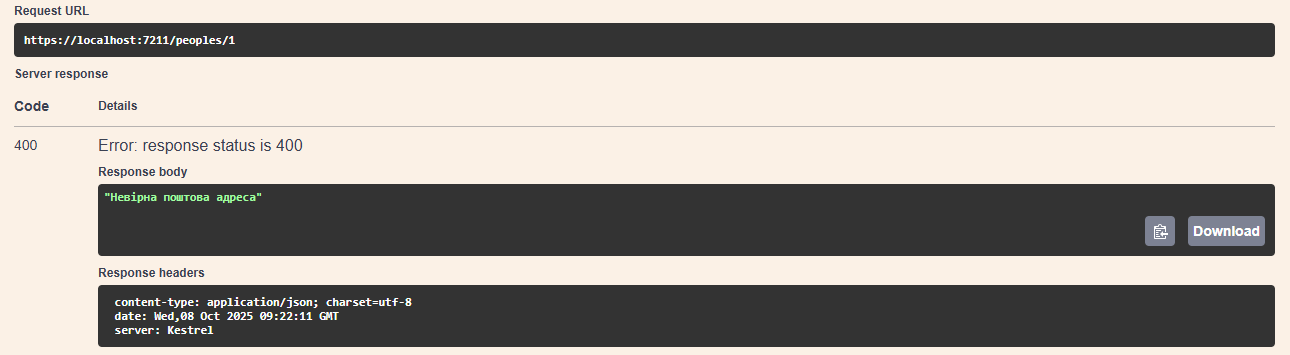


Рисунок 5 – Помилка валідації – невірна електронна адреса

Запит 6

****

Рисунок 6 – Часткове оновлення користувача

Запит 7

****

Рисунок 7 – Видалення користувача

Запит 8

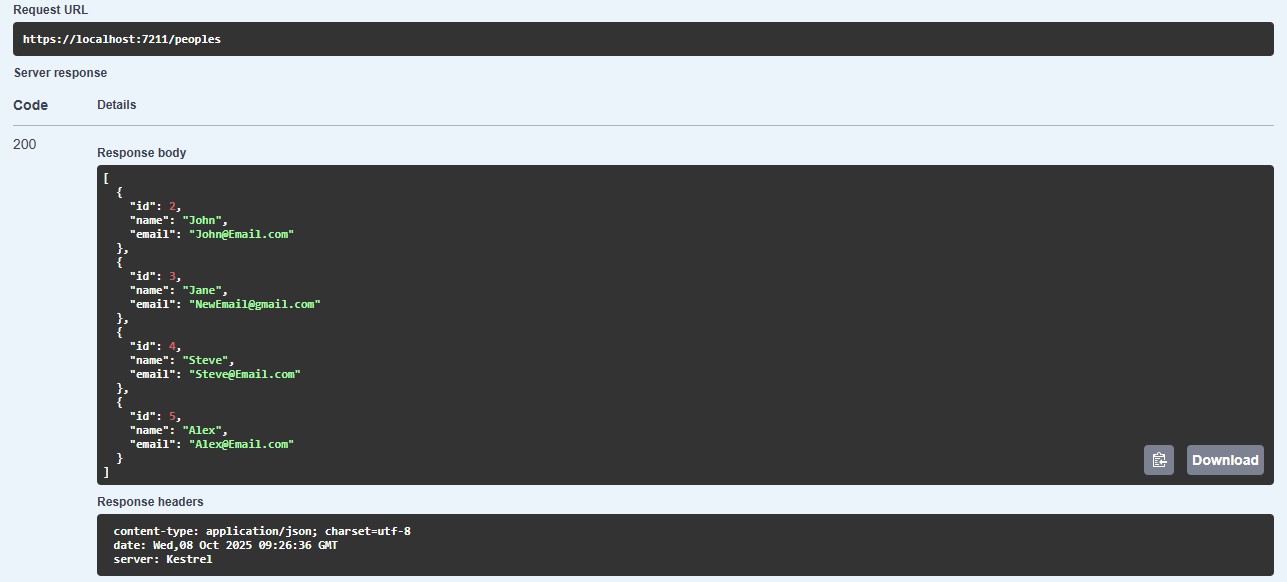
****

Рисунок 8 – Отримано список всіх користувачів

Запит 9

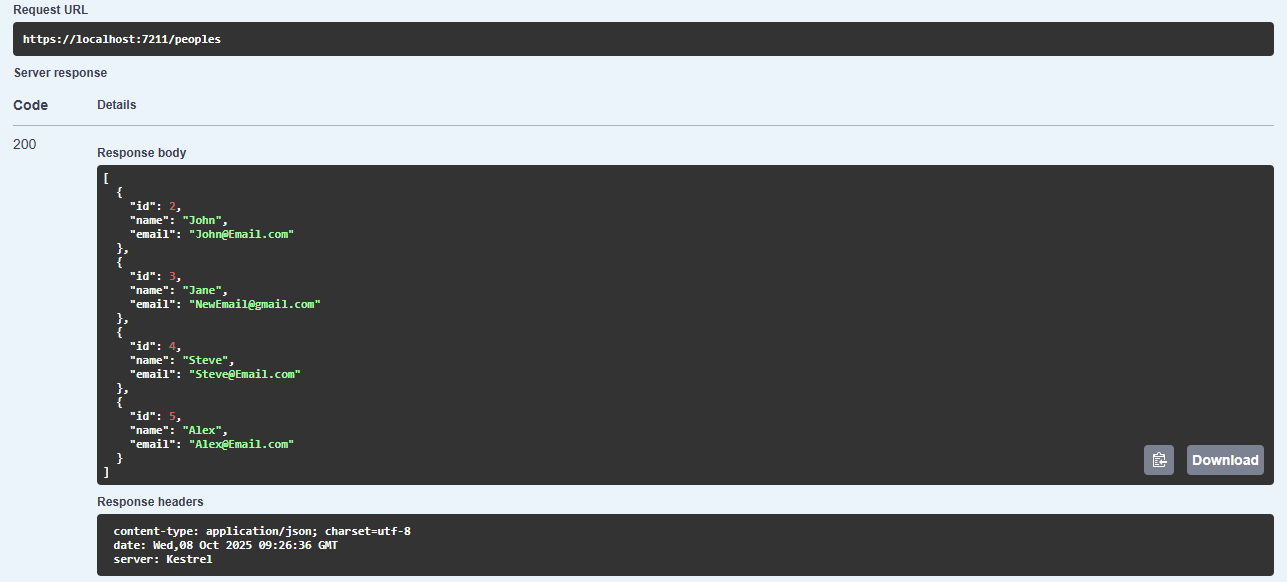


Рисунок 9 – Отримано всіх користувачів

Запит 10

****

Рисунок 10 – Отримано конкретного користувача

Запит 11

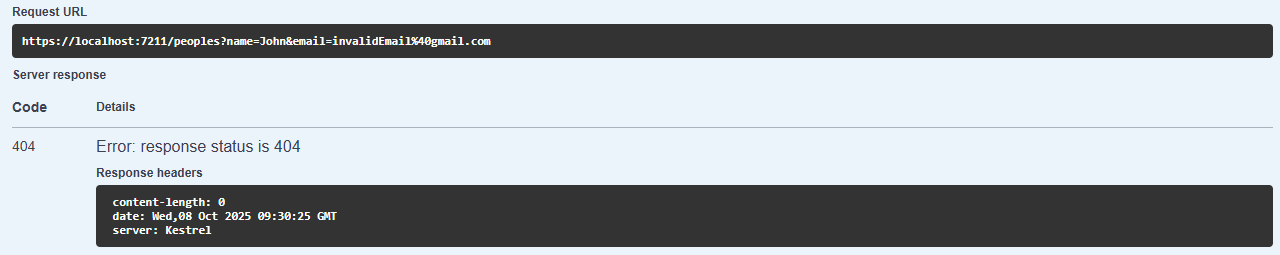
****

Рисунок 11 – Користувача не знайдено

Запит 12

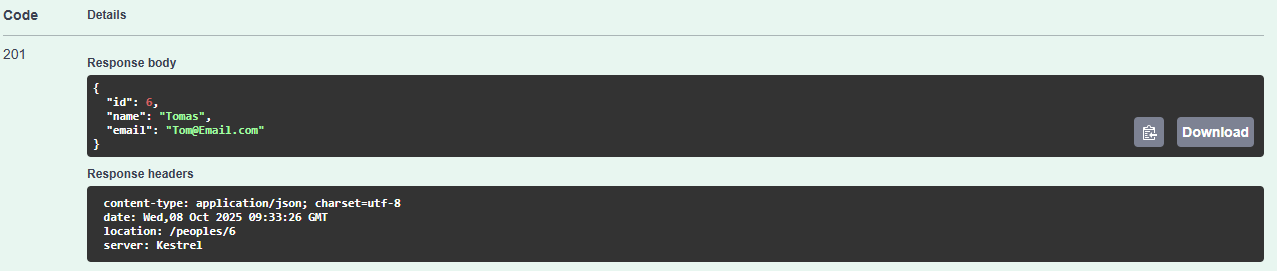
****

Рисунок 12 – Створення нового користувача

Запит 13

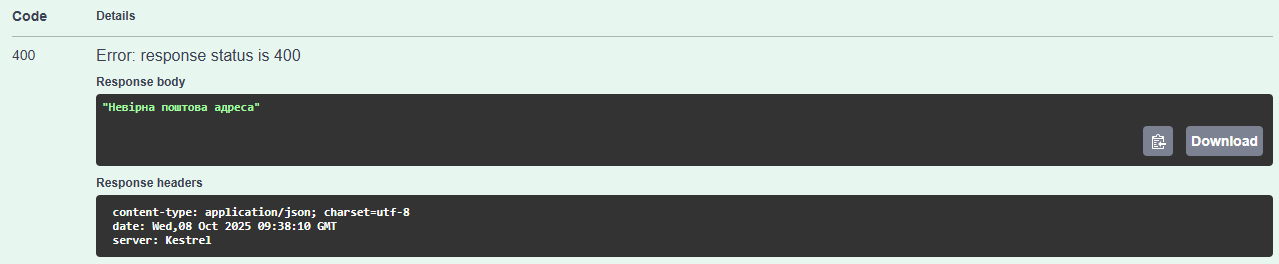


Рисунок 13 – Спроба додавання користувача з невалідною адресою

Запит 14

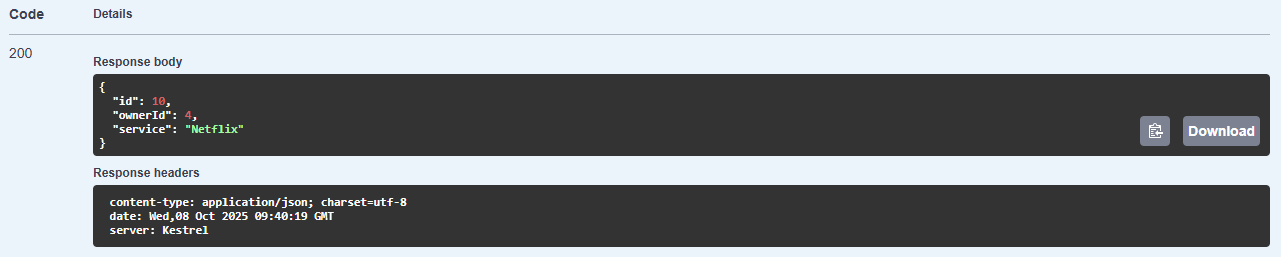


Рисунок 14 – Отримання конкретної підписки

Запит 15

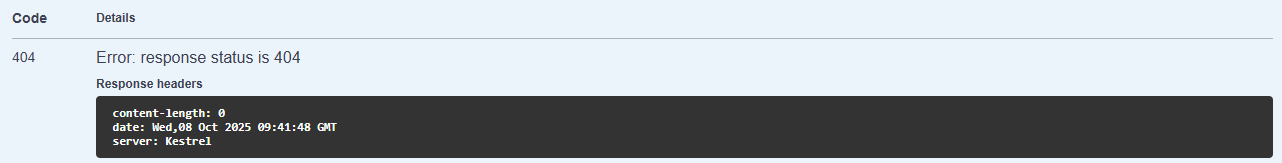


Рисунок 15 – спроба отримати неіснуючу підписку

Запит 16

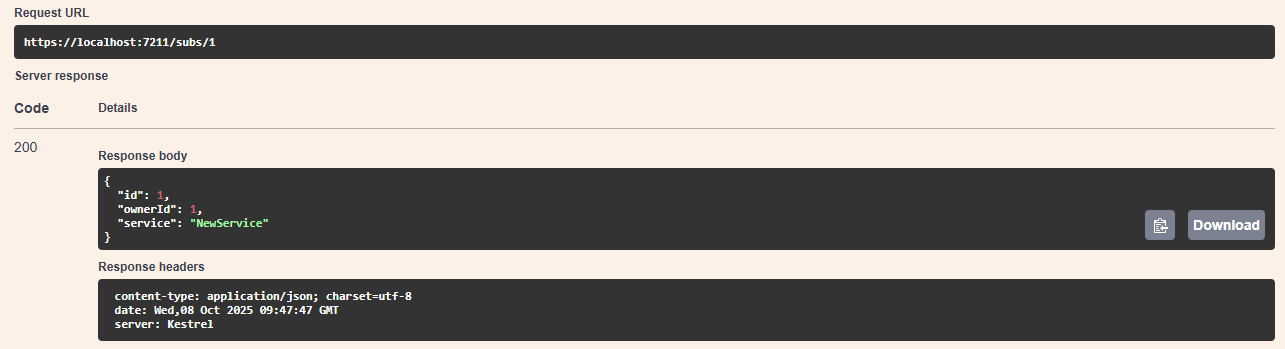


Рисунок 16 - Оновлення підписки

Запит 17

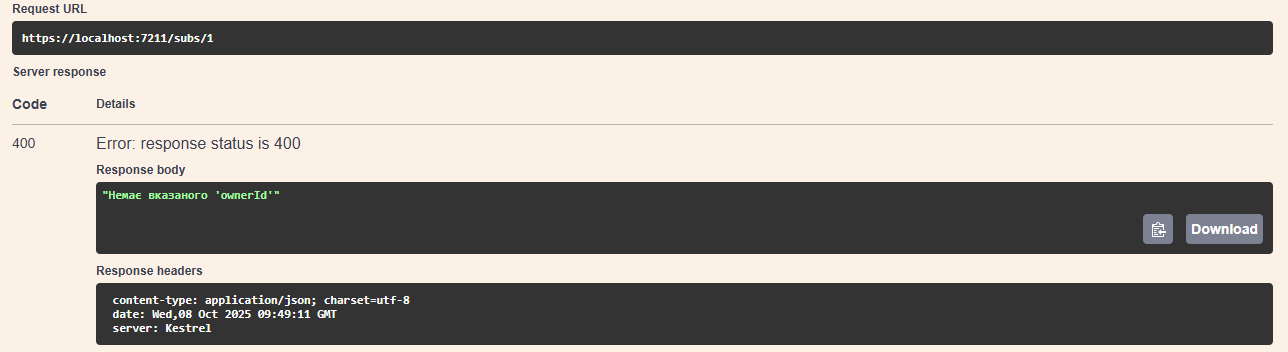


Рисунок 17 – Оновлення підписки з Id користувача, якого не існує

Запит 18

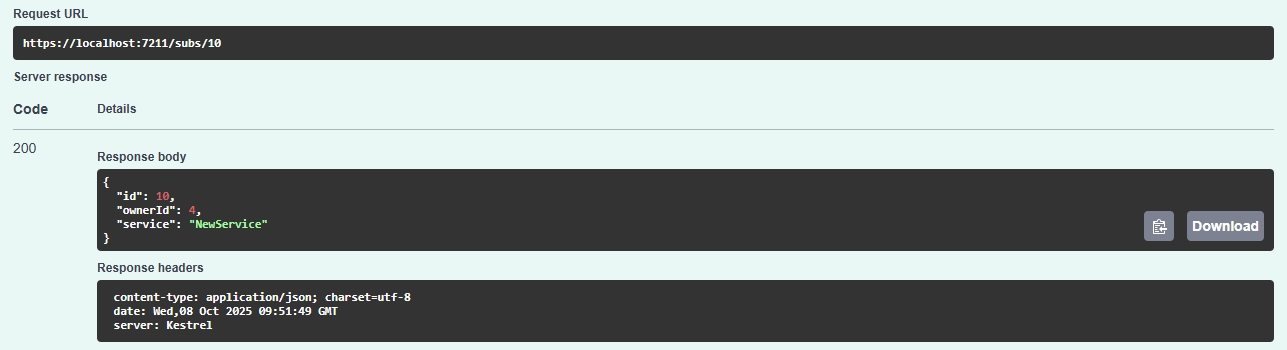


Рисунок 18 – Часткове оновлення користувача

Запит 19

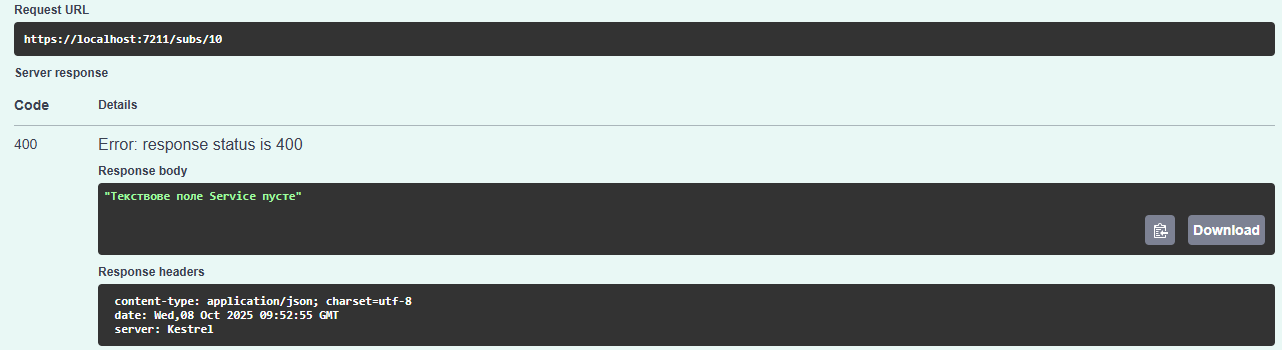


Рисунок 19 – Спроба оновлення поля підписки на пусте

Запит 20



Рисунок 20 – Видалення підписки

Запит 21

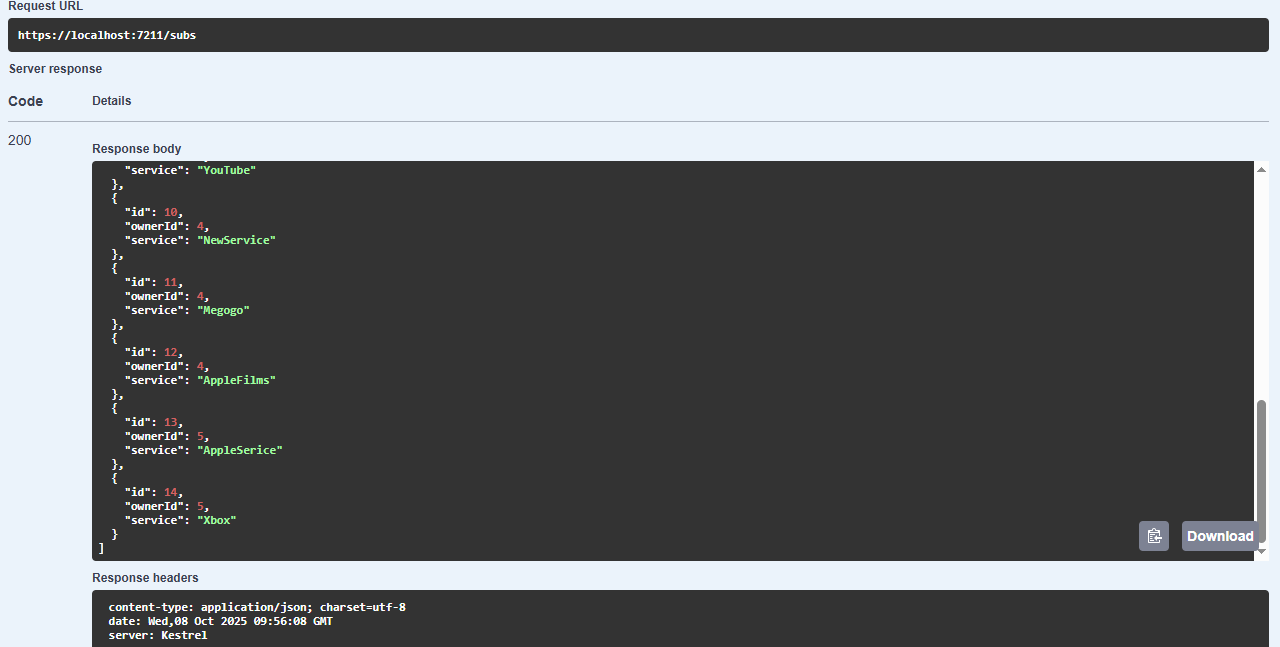


Рисунок 21 – Отримання списку всіх підписок

Запит 22

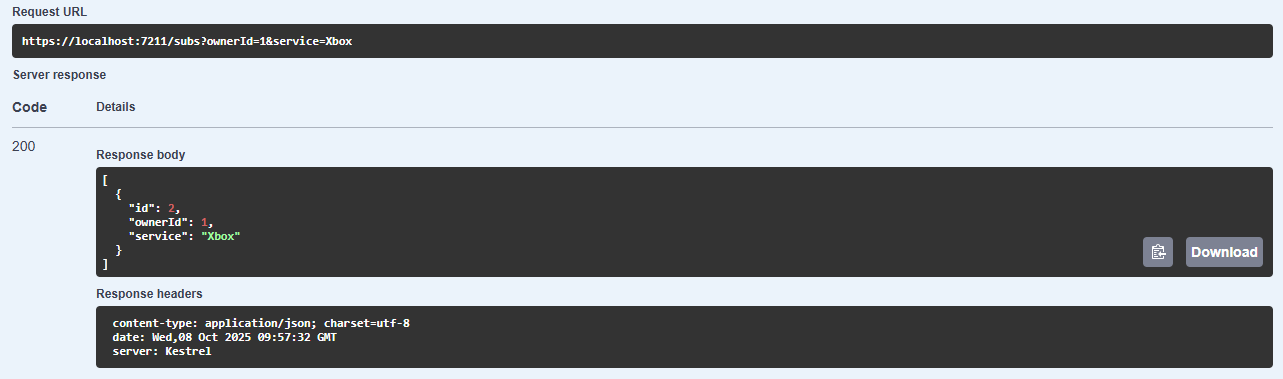


Рисунок 22 – Отримання підписки за вказаними полями

Запит 23

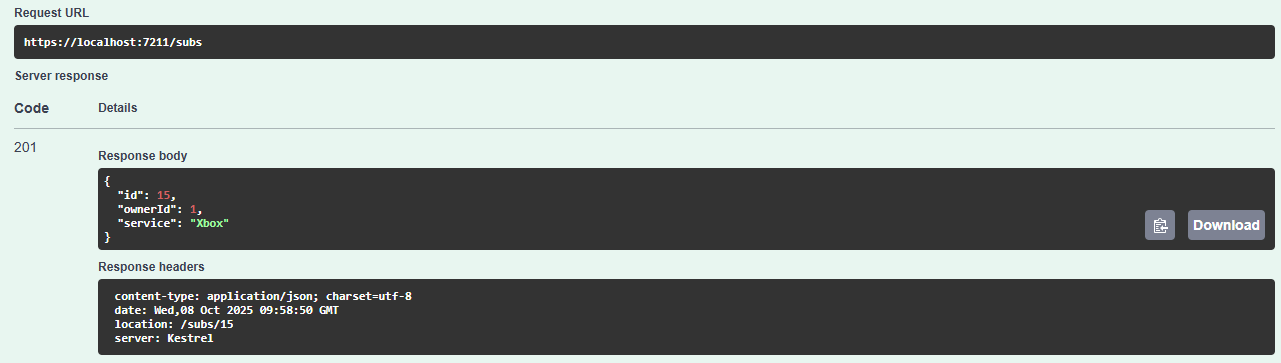


Рисунок 23 – Створення нової підписки

Запит 24

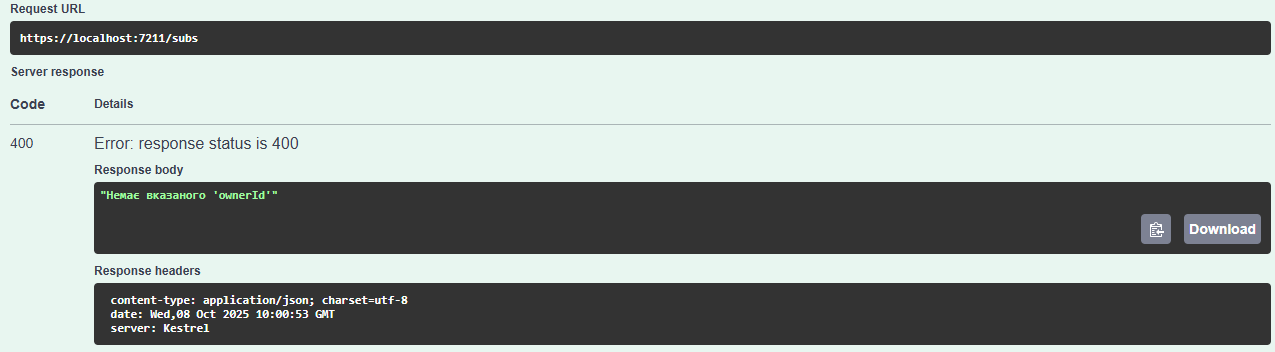


Рисунок 24 – Спроба додавання нової підписки з неіснуючм користувачем

Запит 25

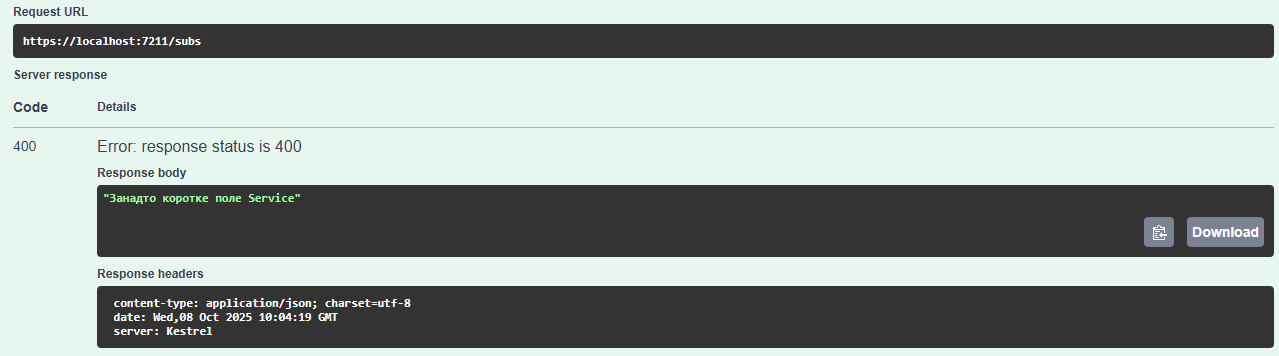


Рисунок 25 – Спроба створення підписки з невалідним сервісом

Запит 26

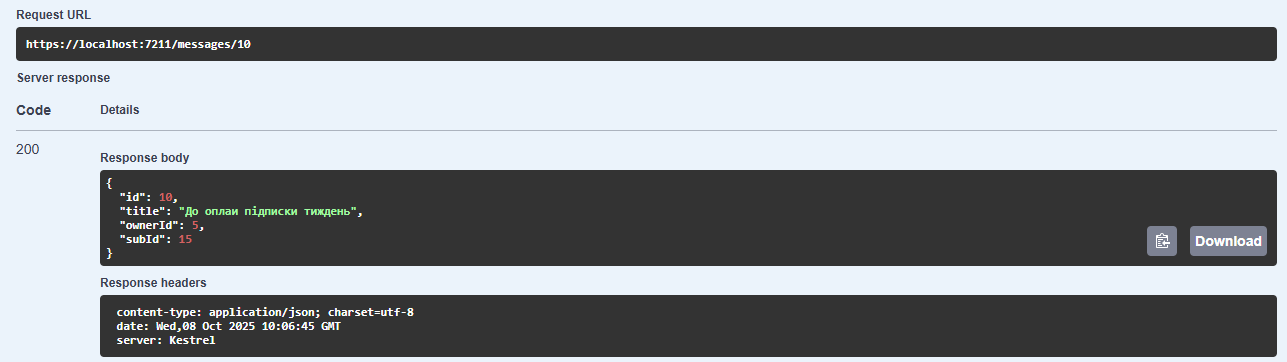


Рисунок 26 – Отримання конкретного повідомлення

Запит 27



Рисунок 27 – Спроба отримання неіснуючої підписки

Запит 28

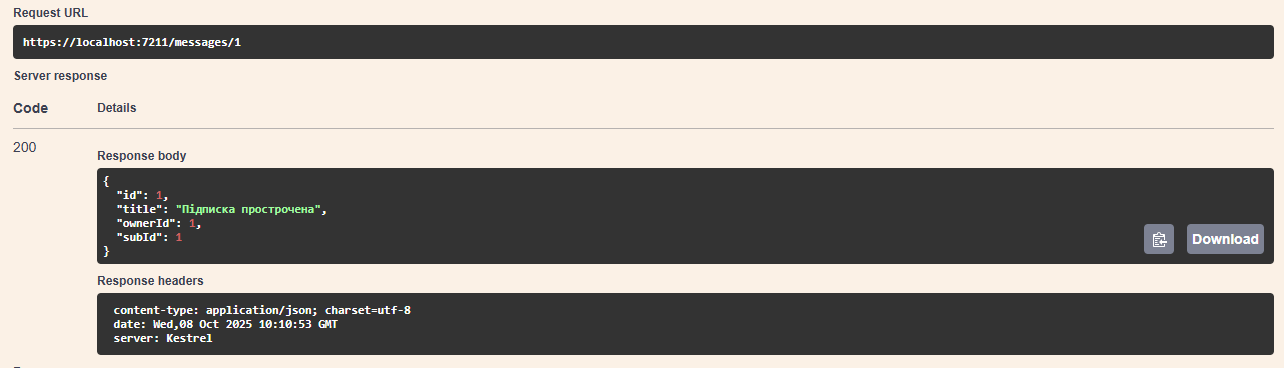


Рисунок 28 – Оновлення вмісту повідомлення

Запит 29

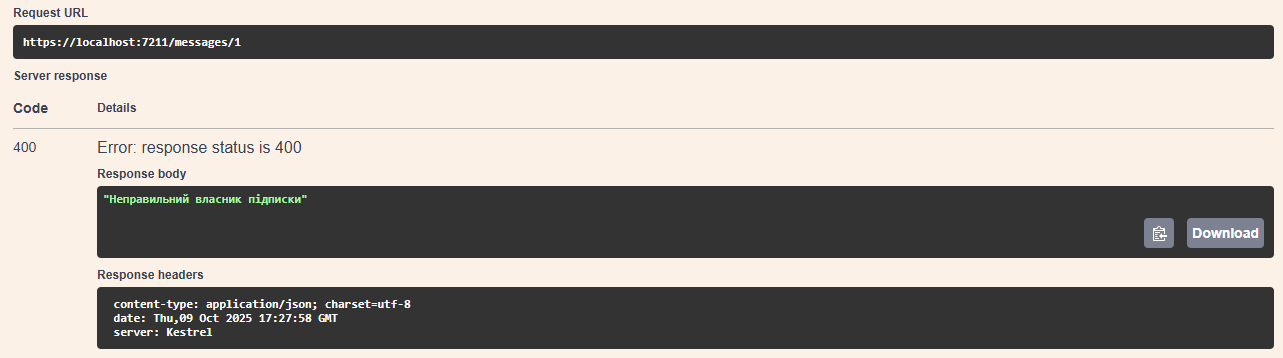


Рисунок 29 – Спроба відправит підписнику нагадування не про те повідолення

Запит 30



Рисунок 30 – Оновлення підписки в користувача

Запит 31

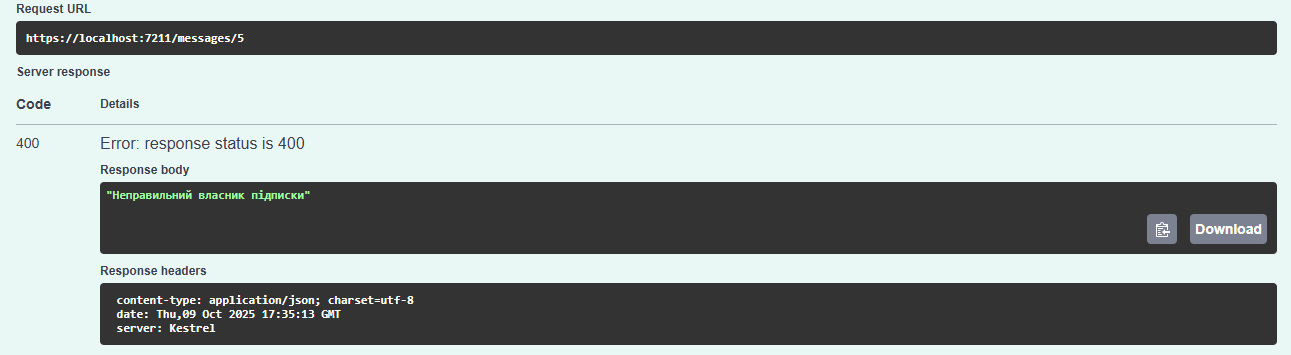


Рисунок 31 – Спроба оновлення невірної підписки користувача

Запит 32



Рисунок 32 – Видалення повідомленн про підписку

Запит 33

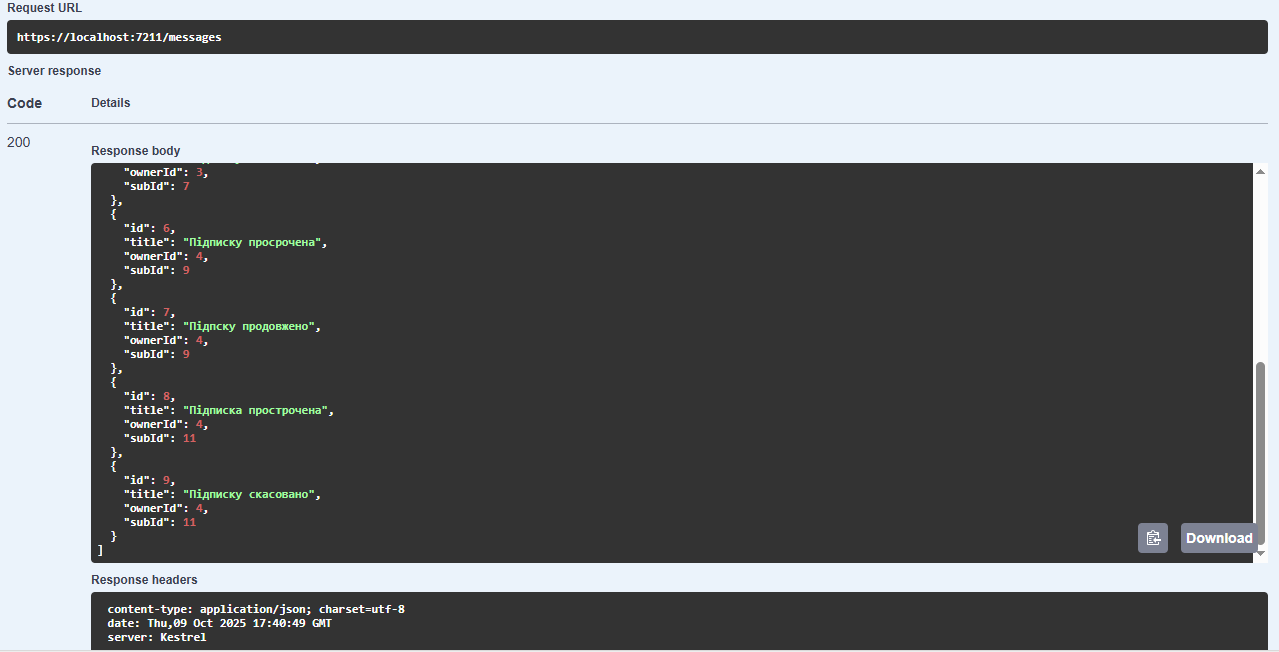


Рисунок 33 -Отримання списку усіх повідомлень

Запит 34



Рисунок 34 – Отримання всіх повідомлень до користувача

Запит 35

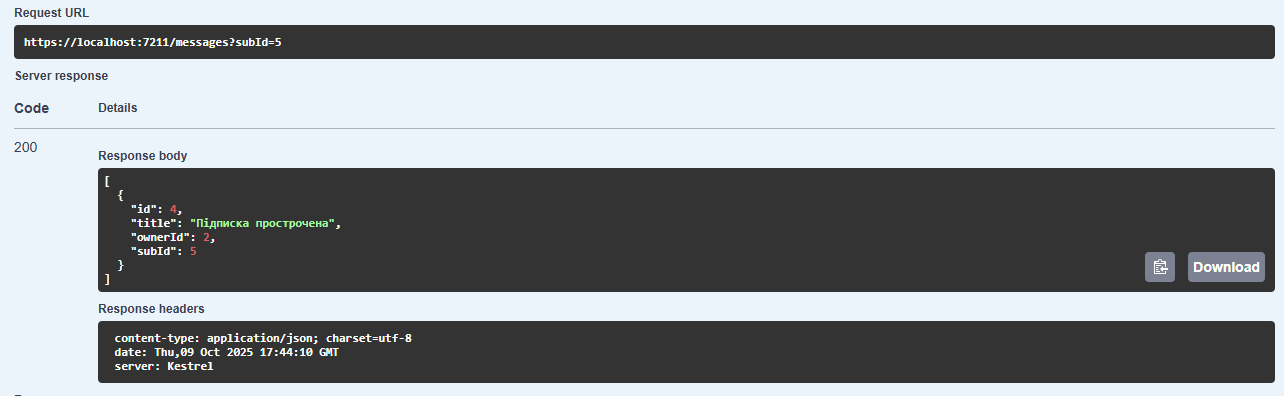


Рисунок 35 – Отримання всіх повідомлень про конретну підписку

Запит 36

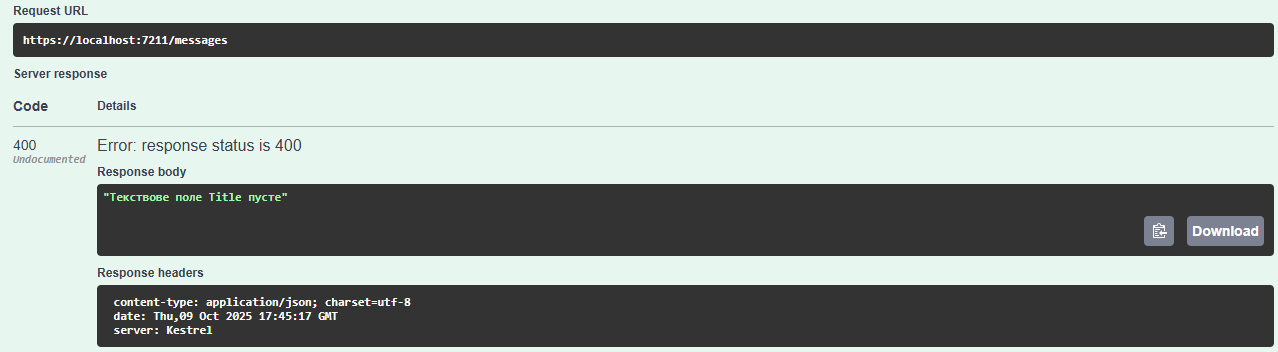


Рисунок 36 – Спроба відправити пусте повідомлення

Запит 37

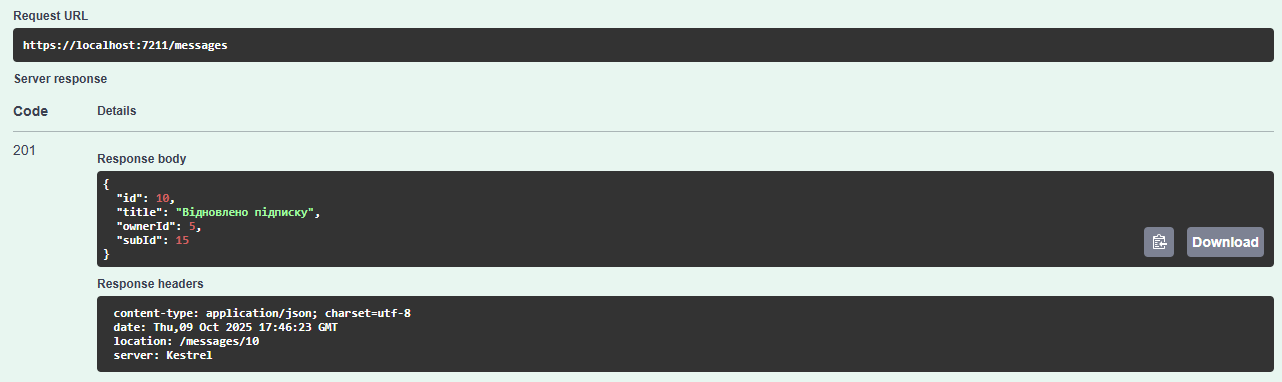


Рисунок 37 – Додано повідомлення для користувача та його підписки

Запит 38

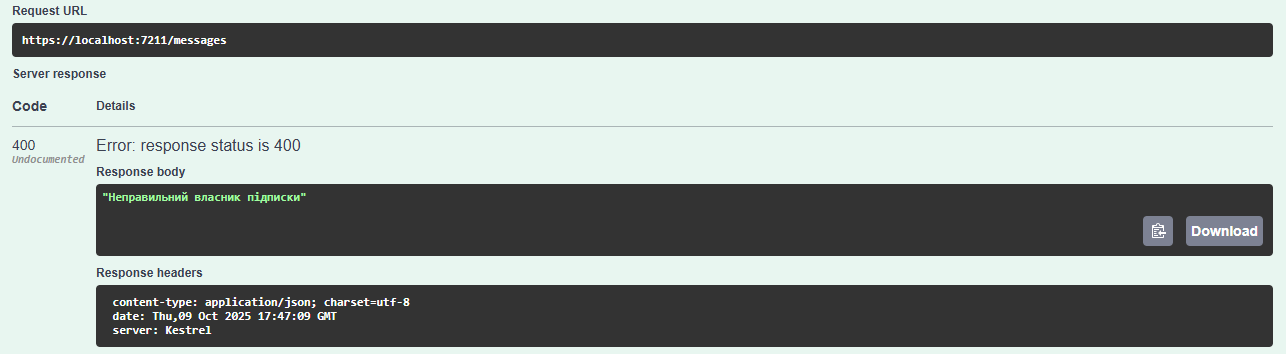


Рисунок 38 – Спроба відправити повідомленн користувачу не про його підписку

Завдання 2

Розроблено проект за допомогою C# ASP.Net Core WebAPI.

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, Status – перелік, для відображення станів підписки,

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.

Реалізовано валідацію в класові PeopleItems за допомогою Data.Anotation, для класів SubscriptionItems, MessageItems за допомогою FluentValidator та винесено логіку в океремий каталог Validator.

Лістинг:

People.cs:

using System.ComponentModel.DataAnnotations;

namespace LW\_4\_2.Models

{

    /// <summary>

    /// Моедль що представляє користувачів програми для управління підписками

    /// </summary>

    public class PeopleItems

    {

        /// <summary>

        /// Унікальний ідентифікатор

        /// </summary>

        public int Id { get; set; }

        /// <summary>

        /// Ім'я або логін користувача

        /// </summary>

        [Required(ErrorMessage = "Поле 'Name' Обов'язкове для заповнення")]

        [RegularExpression(@"^[A-Z][a-z]{5,10}[0-9]{0,5}$", ErrorMessage = "'Name' не відповідає вимогам:" +

            "Перші велика англійська літера, далі від 5 до 10 малих англійських символів та цифри за бажанням")]

        public string? Name { get; set; }

        /// <summary>

        /// Поштова адреса користувача

        /// </summary>

        [Required(ErrorMessage = "Поле 'Email' є обов'язковим")]

        [EmailAddress(ErrorMessage = "Не коректне значення 'Email'")]

        public string? Email { get; set; }

    }

}

Subscription.cs:

namespace LW\_4\_2.Models

{

    /// <summary>

    /// Перелік статусів підписки

    /// </summary>

    public enum SubStatus

    {

        /// <summary>

        /// Статус очікування - Expectation - 0

        /// </summary>

        Expectation,

        /// <summary>

        /// Статус ативна - Active - 1

        /// </summary>

        Active,

        /// <summary>

        /// Статус прострочена - Overdue - 2

        /// </summary>

        Overdue,

        /// <summary>

        /// Статус Архівована - Archived - 3

        /// </summary>

        Archived

    }

    /// <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; }

        /// <summary>

        /// Статус в якому перебуваєпідписка користувача

        /// </summary>

        public SubStatus? Status { get; set; }

    }

}

Message.cs:

namespace LW\_4\_2.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; }

    }

}

PeopleData.cs:

using LW\_4\_2.Models;

namespace LW\_4\_2.Data

{

    public static class PeopleData

    {

        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" }

        };

    }

}

SubData.cs:

using LW\_4\_2.Models;

namespace LW\_4\_2.Data

{

    public static class SubData

    {

        public static List<SubscriptionItems> subsripptionItems = new List<SubscriptionItems>()

                {

                    new SubscriptionItems(){Id = 1, OwnerId = 1, Service = "Netflix", Status = SubStatus.Active},

                    new SubscriptionItems(){Id = 2, OwnerId = 1, Service = "Xbox",Status = SubStatus.Active},

                    new SubscriptionItems(){Id = 3, OwnerId = 1, Service = "Amazon",Status = SubStatus.Active},

                    new SubscriptionItems(){Id = 4, OwnerId = 2, Service = "Steam",Status = SubStatus.Active },

                    new SubscriptionItems(){Id = 5, OwnerId = 2, Service = "Google", Status = SubStatus.Overdue},

                    new SubscriptionItems(){Id = 6, OwnerId = 2, Service = "Amazon", Status = SubStatus.Active},

                    new SubscriptionItems(){Id = 7, OwnerId = 3, Service = "YouTube",Status = SubStatus.Archived},

                    new SubscriptionItems(){Id = 8, OwnerId = 3, Service = "Netflix", Status = SubStatus.Active},

                    new SubscriptionItems(){Id = 9, OwnerId = 4, Service = "YouTube", Status = SubStatus.Active},

                    new SubscriptionItems(){Id = 10, OwnerId = 4, Service = "Netflix", Status = SubStatus.Active},

                    new SubscriptionItems(){Id = 11, OwnerId = 4, Service = "Megogo", Status = SubStatus.Archived},

                    new SubscriptionItems(){Id = 12, OwnerId = 4, Service = "AppleFilms", Status = SubStatus.Active},

                    new SubscriptionItems(){Id = 13, OwnerId = 5, Service = "AppleSerice", Status = SubStatus.Active},

                    new SubscriptionItems(){Id = 14, OwnerId = 5, Service = "Xbox", Status = SubStatus.Active},

                    new SubscriptionItems(){Id = 15, OwnerId = 5, Service = "AppleFilms", Status = SubStatus.Active},

                };

    }

}

MessageData.cs:

using LW\_4\_2.Models;

namespace LW\_4\_2.Data

{

    public static class MessageData

    {

        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", OwnerId = 1, SubId = 1 },

            new MessageItems(){Id = 4,Title = "Підписка прострочена", OwnerId = 2, SubId = 5 },

            new MessageItems(){Id = 5,Title = "Підписку скасовано", OwnerId = 3, SubId = 7 },

            new MessageItems(){Id = 6,Title = "Підписку просрочена", OwnerId = 4, SubId = 9 },

            new MessageItems(){Id = 7,Title = "Підпску продовжено", OwnerId = 4, SubId = 9 },

            new MessageItems(){Id = 8,Title = "Підписка прострочена", OwnerId = 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 },

        };

    }

}

MessageValidator.cs:

using FluentValidation;

using LW\_4\_2.Models;

using LW\_4\_2.Data;

namespace LW\_4\_2.Validator

{

    public class MessageValidator: AbstractValidator<MessageItems>

    {

        public MessageValidator()

        {

            RuleFor(x => x.Title)

                .NotEmpty().WithMessage("Поле 'Title' не може бути порожнім")

                .Length(5, 100).WithMessage("Поле 'Title' має бути від трьох до ста символів");

            RuleFor(x => x.SubId)

                .NotNull().WithMessage("Поле 'SubId' не може бути порожнім")

                .Must(SubId => SubData.subsripptionItems.Any(s => s.Id == SubId))

                .WithMessage("Не знайдено 'SubId' із таким індексом");

            RuleFor(x => x.OwnerId)

                .NotEmpty().WithMessage("Поле 'OwnerId' не можу бу порожнім")

                .Must(ownerId => PeopleData.peopleItems.Any(u => u.Id == ownerId))

                .WithMessage("Немає людина із таким значенням 'Id'")

                .Must((mes, ownerId) =>

                {

                    var sub = SubData.subsripptionItems.FirstOrDefault(s => s.Id == mes.SubId);

                    if (sub is null)

                        return false;

                    return sub.OwnerId == ownerId;

                }).WithMessage("Відсутня підписка за вказаним 'SubId' або для неї не існує користувача");

        }

    }

}

SubscriptionValidtor.cs:

using LW\_4\_2.Models;

using LW\_4\_2.Data;

using FluentValidation;

namespace LW\_4\_2.Validator

{

    public class SubscriptionValidator: AbstractValidator<SubscriptionItems>

    {

        public SubscriptionValidator()

        {

            RuleFor(x => x.OwnerId)

                .NotEmpty().WithMessage("Поле 'OwnerId не може бути порожнім'")

                .Must(ownerID => SubData.subsripptionItems.Any(s => s.OwnerId == ownerID))

                .WithMessage("Не знайдено користувача за вказаним 'OwnerId'");

            RuleFor(x => x.Service)

                .NotEmpty().WithMessage("Поле 'Service' не може бути порожнім")

                .Length(3, 50).WithMessage("Поле 'Service' має бути в межах від 3 до 50 символів");

            RuleFor(x => x.Status)

                .NotEmpty().WithMessage("Поле 'Status' не можу бути порожнім")

                .IsInEnum().WithMessage("Поле 'Status' має бути вибране в заданих межах");

        }

    }

}

PeopleController.cs:

    using LW\_4\_2.Data;

    using LW\_4\_2.Models;

    using Microsoft.AspNetCore.Mvc;

using System.Text.Json;

namespace LW\_4\_2.Controllers

    {

        [ApiController]

        [Route("[controller]")]

        public class PeopleController : ControllerBase

        {

            [HttpGet]

            public ActionResult<IEnumerable<PeopleItems>> GetAll(

                [FromQuery] string? Name, [FromQuery] string? Email)

            {

                IEnumerable<PeopleItems> peoples = PeopleData.peopleItems;

                if(Name is not null)

                    peoples = peoples.Where(x => x.Name == Name);

                if(Email is not null)

                    peoples = peoples.Where(x => x.Email == Email);

                if (!peoples.Any())

                    return NotFound("Не знайдено елементів");

                return Ok(peoples);

            }

            [HttpGet("{id}")]

            public ActionResult<PeopleItems> GetById(int id)

            {

                var people = PeopleData.peopleItems.FirstOrDefault(x => x.Id == id);

                if (people is null)

                    return NotFound($"Не знайдено людину із вказаним 'Id': {id}");

                return Ok(people);

            }

            [HttpPost]

            public ActionResult<PeopleItems> Create(PeopleItems item)

            {

                item.Id = PeopleData.peopleItems.Max(x => x.Id) + 1;

                PeopleData.peopleItems.Add(item);

                return CreatedAtAction(nameof(GetById), new { Id = item.Id }, item);

            }

            [HttpPut("{id}")]

            public IActionResult Update(int id, PeopleItems upPeople)

            {

                var people = PeopleData.peopleItems.FirstOrDefault(p => p.Id == id);

                if (people is null)

                    return NotFound($"Не знайдено людини із Id {id}");

                people.Name = upPeople.Name;

                people.Email = upPeople.Email;

                return NoContent();

            }

            [HttpPatch("{id}")]

            public IActionResult UpdatePart(int id, [FromBody] JsonElement upPeople)

            {

                ModelState.ClearValidationState(nameof(upPeople));

                var people = PeopleData.peopleItems.FirstOrDefault(x => x.Id == id);

                if(people is null)

                    return NotFound($"Не знайдено людини за вказаним Id {id}");

            var temp = new PeopleItems

            {

                Id = id,

                Email = people.Email,

                Name = people.Name

            }

                    if(upPeople.TryGetProperty("name",out var name))

                    temp.Name = name.GetString();

                if(upPeople.TryGetProperty("email",out var email))

                     temp.Email = email.GetString();

                if (!TryValidateModel(temp))

                return BadRequest(ModelState);

                people.Name = temp.Name;

                people.Email = temp.Email;

               return NoContent();

            }

            [HttpDelete("{id}")]

            public IActionResult DeletePart(int id)

            {

                var people = PeopleData.peopleItems.FirstOrDefault(p => p.Id == id);

                if (people is null)

                    return NotFound($"Не знайдено людину із вказаним Id {id}");

                PeopleData.peopleItems.Remove(people);

                return NoContent();

            }

        }

    }

SubController.cs:

using LW\_4\_2.Data;

using LW\_4\_2.Models;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.ApplicationParts;

using System.Text.Json;

namespace LW\_4\_2.Controllers

{

    [ApiController]

    [Route("[controller]")]

    public class SubController : ControllerBase

    {

        [HttpGet]

        public ActionResult<IEnumerable<SubscriptionItems>> GetAll(

            [FromQuery] int? ownerId, [FromQuery] string? service, [FromQuery] SubStatus? subStatus)

        {

            IEnumerable<SubscriptionItems> subItems = SubData.subsripptionItems;

            if (ownerId is not null)

                subItems = subItems.Where(x => x.OwnerId == ownerId);

            if (service is not null)

                subItems = subItems.Where(x => x.Service == service);

            if (subStatus is not null)

                subItems = subItems.Where(x => x.Status == subStatus);

            if (subItems.Count() == 0)

                return NotFound("Не знайдено елементів за запитом");

            return Ok(subItems);

        }

        [HttpGet("{id}")]

        public ActionResult<SubscriptionItems> GetById(int id)

        {

            var sub = SubData.subsripptionItems.FirstOrDefault(x => x.Id == id);

            if (sub is null)

                return NotFound($"Не знайдено підписки з Id {id}");

            return Ok(sub);

        }

        [HttpPost]

        public ActionResult<SubscriptionItems> Create(SubscriptionItems item)

        {

            item.Id = SubData.subsripptionItems.Max(x => x.Id) + 1;

            SubData.subsripptionItems.Add(item);

            return CreatedAtAction(nameof(GetById), new { Id = item.Id }, item);

        }

        [HttpPut("{id}")]

        public IActionResult Update(int id, SubscriptionItems item)

        {

            var sub = SubData.subsripptionItems.FirstOrDefault(x => x.Id == id);

            if (sub is null)

                return NotFound($"Не знайдено підписки з Id {id}");

            sub.OwnerId = item.OwnerId;

            sub.Service = item.Service;

            sub.Status = item.Status;

            return NoContent();

        }

        [HttpPatch("{id}")]

        public IActionResult UpdatePart(int id, JsonElement item)

        {

            var sub = SubData.subsripptionItems.FirstOrDefault(x => x.Id == id);

            if (sub is null)

                return NotFound($"Не знайдено підписки з Id {id}");

            var temp = new SubscriptionItems

            {

                Id = sub.Id,

                OwnerId = sub.OwnerId,

                Service = sub.Service,

                Status = sub.Status

            };

            if(item.TryGetProperty("ownerId",out var upId))

                temp.OwnerId = upId.GetInt32();

            if (item.TryGetProperty("status",out var upStatus))

                temp.Status = (SubStatus)upStatus.GetInt32();

            if (item.TryGetProperty("service",out var upServ))

                temp.Service = upServ.GetString();

            TryValidateModel(temp);

            if (!ModelState.IsValid)

                return BadRequest(ModelState);

            sub.OwnerId = temp.OwnerId;

            sub.Service = temp.Service;

            sub.Status = temp.Status;

            return NoContent();

        }

        [HttpDelete("{id}")]

        public IActionResult Delete(int id)

        {

            var sub = SubData.subsripptionItems.FirstOrDefault(x => x.Id == id);

            if (sub is null)

                return NotFound($"Не знайдено підписки з Id {id}");

            SubData.subsripptionItems.Remove(sub);

            return NoContent();

        }

    }

}

MessageController.cs:

using LW\_4\_2.Data;

using LW\_4\_2.Models;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.ModelBinding;

using System.Text.Json;

namespace LW\_4\_2.Controllers

{

    [ApiController]

    [Route("[controller]")]

    public class MessageController: ControllerBase

    {

        [HttpGet]

        public ActionResult<IEnumerable<MessageItems>> GetAll(

            [FromQuery] int? ownerId,

            [FromQuery] int? subId,

            [FromQuery] string? title)

        {

            IEnumerable<MessageItems> messages = MessageData.messageItems;

            if(ownerId is not null)

                messages = messages.Where(x => x.OwnerId == ownerId);

            if (subId is not null)

                messages = messages.Where(x => x.SubId == subId);

            if (title is not null)

                messages = messages.Where(x => x.Title == title);

            if (!messages.Any())

                return NotFound("Не знайдено елементів за вказаним запитом");

            return Ok(messages);

        }

        [HttpGet("{id}")]

        public ActionResult<MessageItems> GetById(int id)

        {

            var message = MessageData.messageItems.FirstOrDefault(x => x.Id == id);

            if (message is null)

                return NotFound($"Повідомлення із вказаним Id {id} не знайдено");

            return Ok(message);

        }

        [HttpPost]

        public ActionResult<MessageItems> Create(MessageItems message)

        {

            message.Id = MessageData.messageItems.Max(x => x.Id) + 1;

            MessageData.messageItems.Add(message);

            return CreatedAtAction(nameof(GetById), new { Id = message.Id}, message);

        }

        [HttpPut("{id}")]

        public IActionResult Update(int id, MessageItems message)

        {

            var mes = MessageData.messageItems.FirstOrDefault(x => x.Id == id);

            if (mes is null)

                return NotFound($"Повідомлення із вказаним Id {id} не знайдено");

            mes.OwnerId = message.OwnerId;

            mes.SubId = message.SubId;

            mes.Title = message.Title;

            return NoContent();

        }

        [HttpPatch("{id}")]

        public IActionResult UpdatePart(int id,JsonElement message)

        {

            var mes = MessageData.messageItems.FirstOrDefault(x => x.Id == id);

            if (mes is null)

                return NotFound($"Повідомлення із вказаним Id {id} не знайдено");

            var temp = new MessageItems()

            {

                Id = id,

                OwnerId = mes.OwnerId,

                SubId = mes.SubId,

                Title = mes.Title,

            };

            if(message.TryGetProperty("ownerId",out var upO))

                temp.OwnerId = upO.GetInt32();

            if(message.TryGetProperty("subId", out var upS))

                temp.SubId = upS.GetInt32();

            if (message.TryGetProperty("title", out var upT))

                temp.Title = upT.GetString();

             TryValidateModel(temp);

            if(!ModelState.IsValid)

                return BadRequest(ModelState);

            mes.OwnerId = temp.OwnerId;

            mes.SubId = temp.SubId;

            mes.Title = temp.Title;

            return NoContent();

        }

        [HttpDelete("{id}")]

        public IActionResult Delete(int id)

        {

            var mes = MessageData.messageItems.FirstOrDefault(x => x.Id == id);

            if (mes is null)

                return NotFound($"Повідомлення із вказаним Id {id} не знайдено");

            MessageData.messageItems.Remove(mes);

            return NoContent();

        }

    }

}

Program.cs:

using FluentValidation;

using FluentValidation.AspNetCore;

using LW\_4\_2.Models;

using LW\_4\_2.Validator;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers().AddFluentValidation();

// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle

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);

});

builder.Services.AddValidatorsFromAssemblyContaining<SubscriptionValidator>();

builder.Services.AddValidatorsFromAssemblyContaining<MessageValidator>();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

    app.UseSwagger();

    app.UseSwaggerUI();

}

app.UseAuthorization();

app.MapControllers();

app.Run();

Протестовано роботу програми за допомогою swagger:

Запит 1

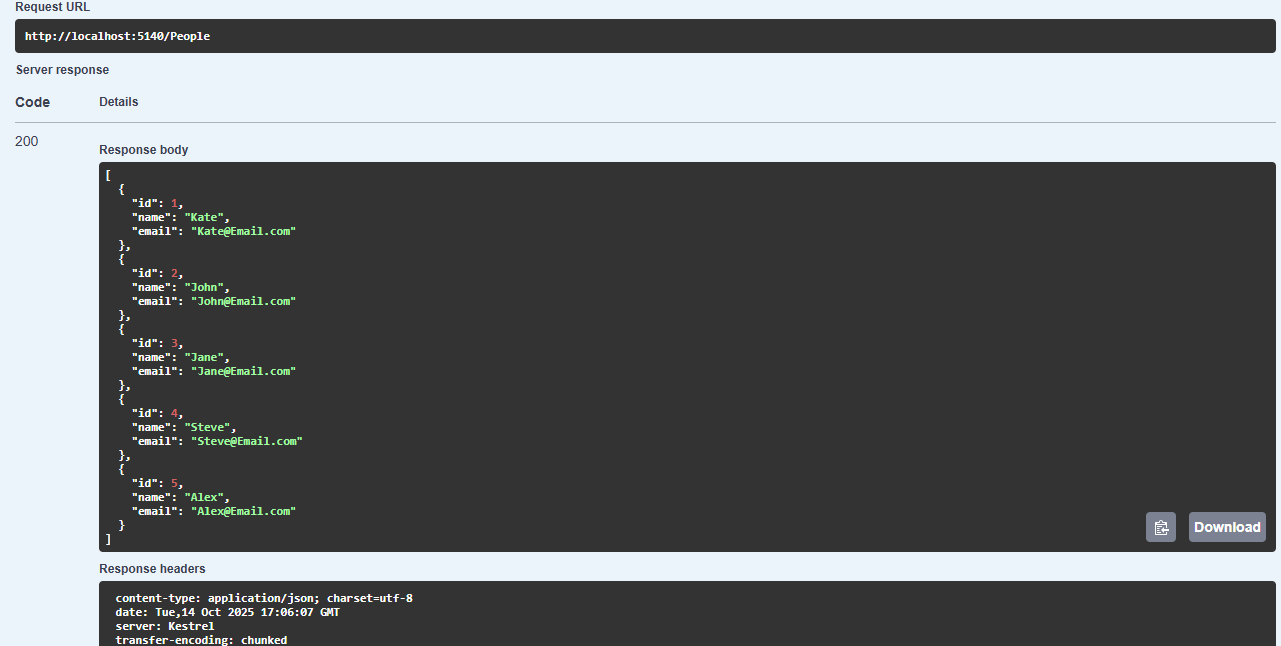


Рисунок 39 – Отримання всіх користувачів

Запит 2

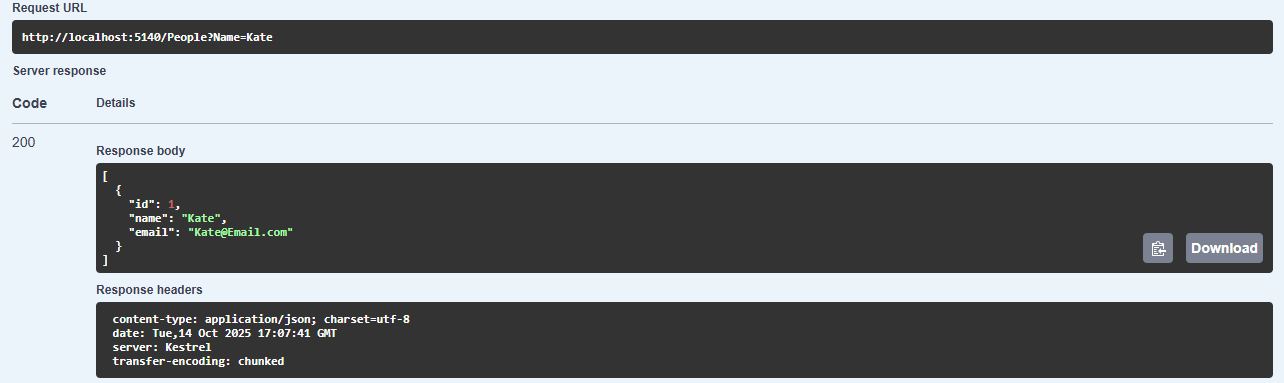


Рисунок 40 – Отримання користувачів за одним параметром

Запит 3

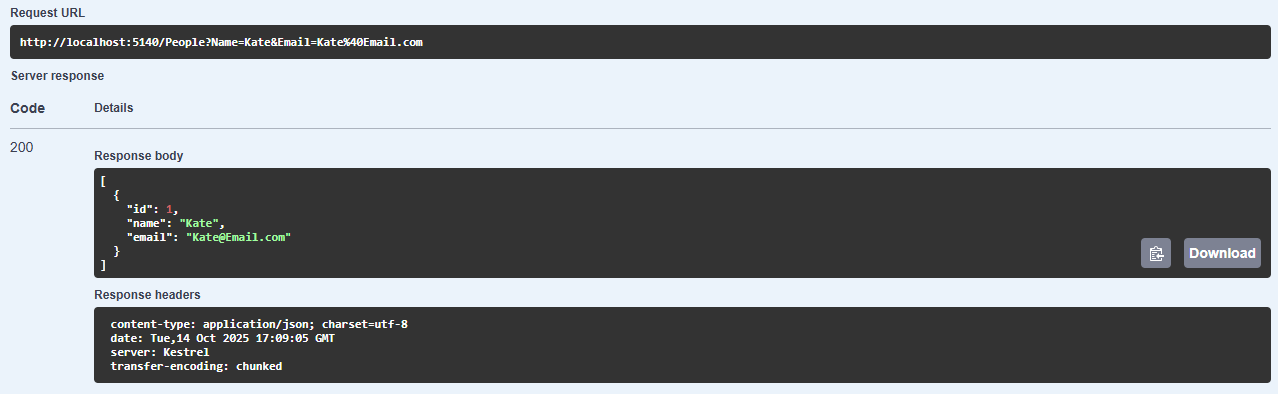


Рисунок 41 -Отримання користувачів за двома параметрами

Запит 4



Рисунок 42 -Спроба отримати користувачів за хибним параметром запиту

Запит 5



Рисунок 43 – Отримання користувача за Id

Запит 6



Рисунок 44 – Спроба отримати неіснуючого користувача

Запит 7

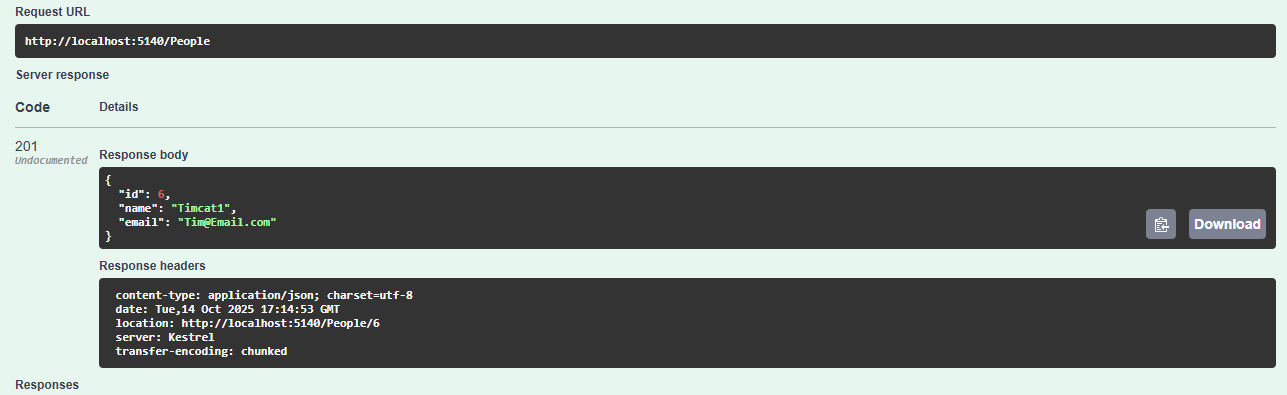


Рисунок 45 – Додавання нового користувача

Запит 8

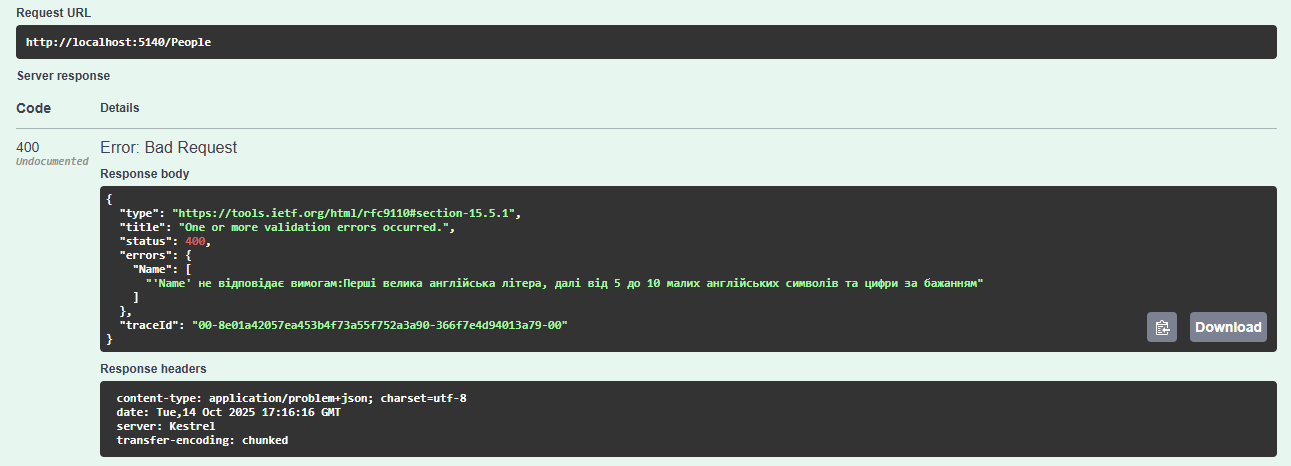


Рисунок 46 – Спроба додавання користуваа із некоректним логінок

Запит 9

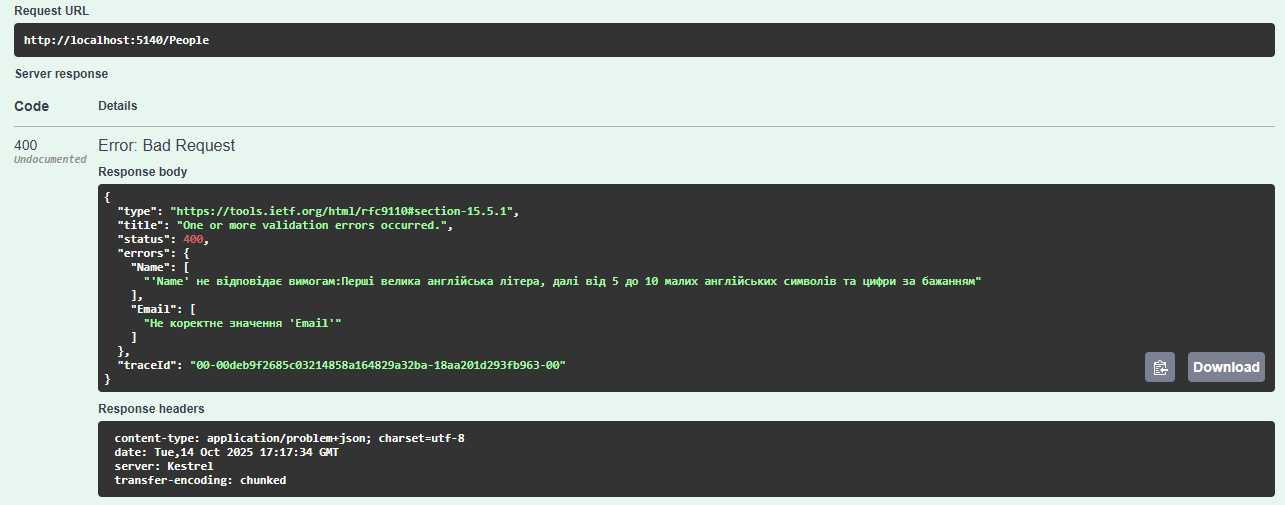


Рисунок 47 – Спроа додавання користувача із некоректними полями

Запит 10

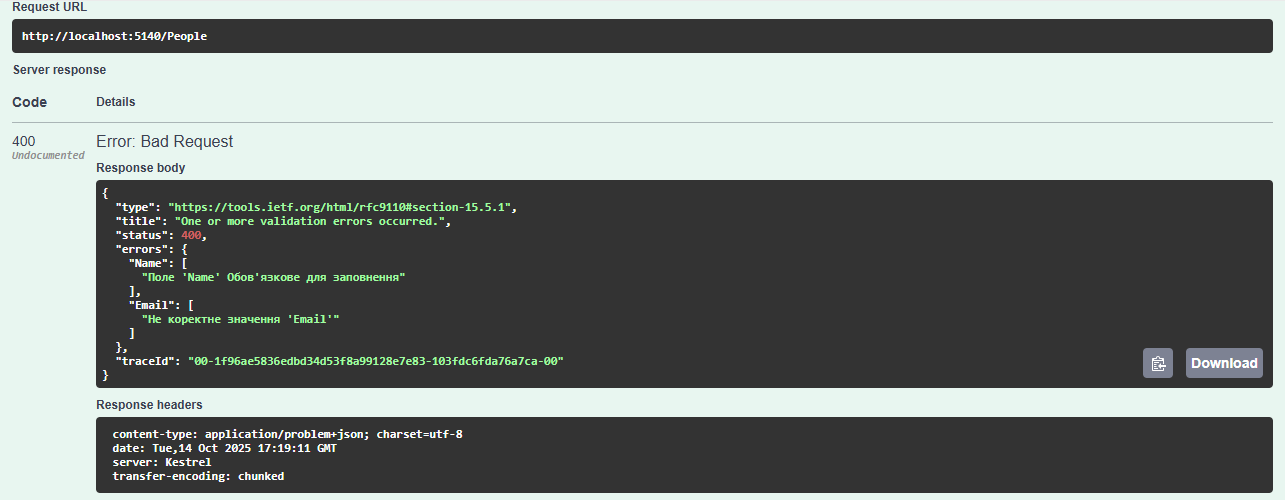


Рисунок 48 – Спроба відправлення пустого поля

Запит 11



Рисунок 49 – Оновлення інформації про першого користувача

Запит 12



Рисунок 50 – Спроа оновлення з використання некоректного логіну

Запит 13



Рисунок 51 – Спроба оновленн на пусті поля

Запит 14

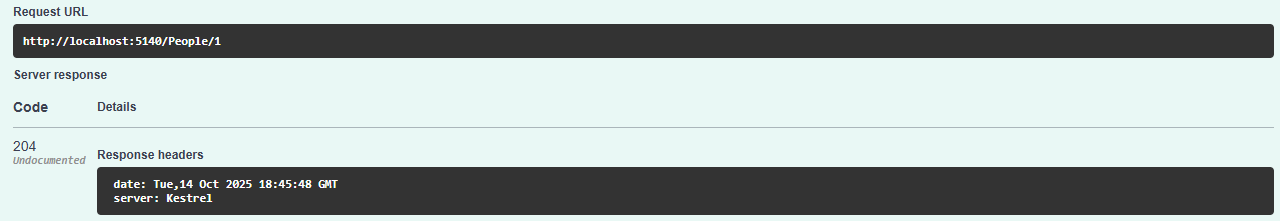


Рисунок 52 – Часткове оновлення користувача

Запит 15



Рисунок 53 – Спроба оновлення на пусте поле

Запит 16

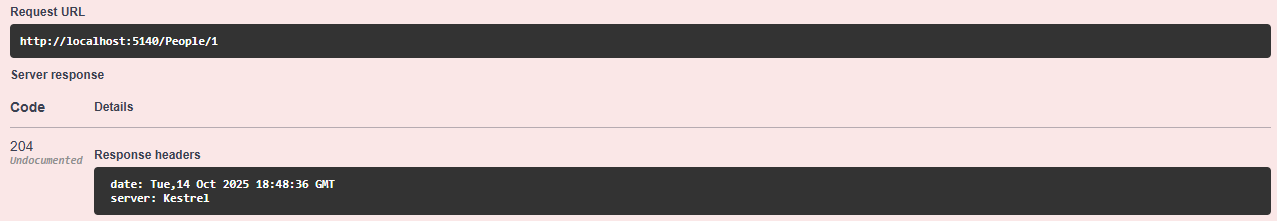


Рисунок 54 – Видалення користувача

Запит 17

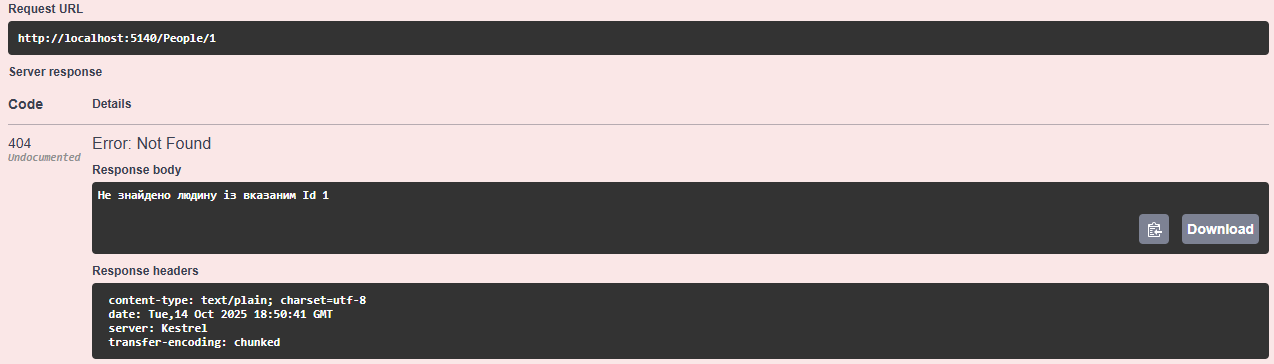


Рисунок 55 – Спроба видалення користувача

Запит 18

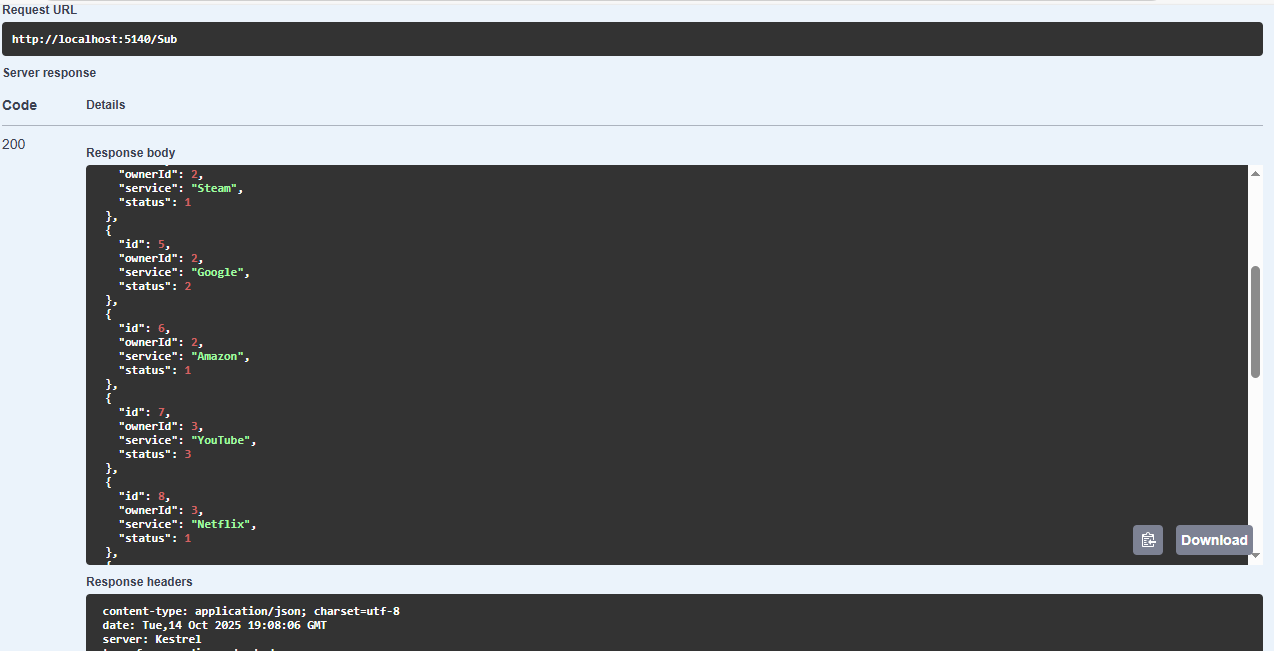


Рисунок 56 – Отримано список всіх підписок

Запит 19

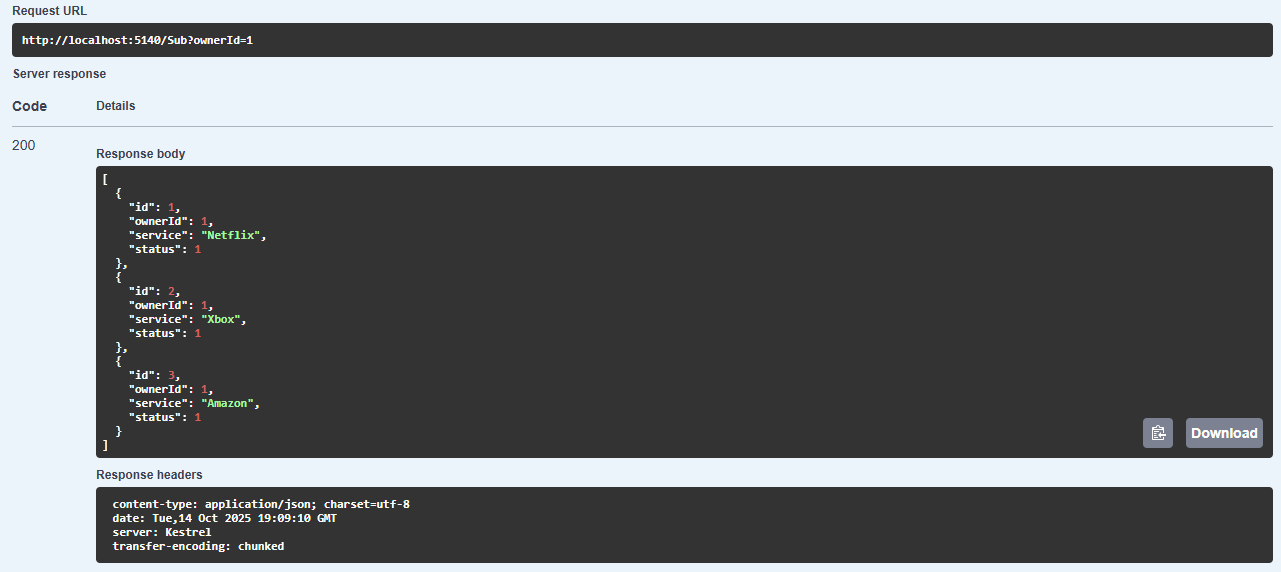


Рисунок 57 – Отримання підписок з викорисанням параметра

Запит 20



Рисунок 58 – Отримання підписок за деількома параметрами

Запит 21

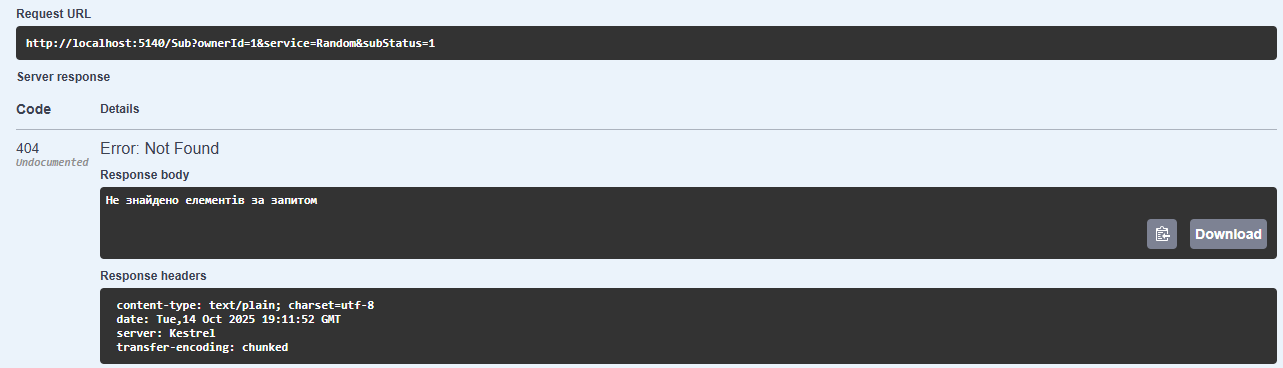


Рисунок 59 – Спроба отримати підписки за відсутнім параметром

Запит 22

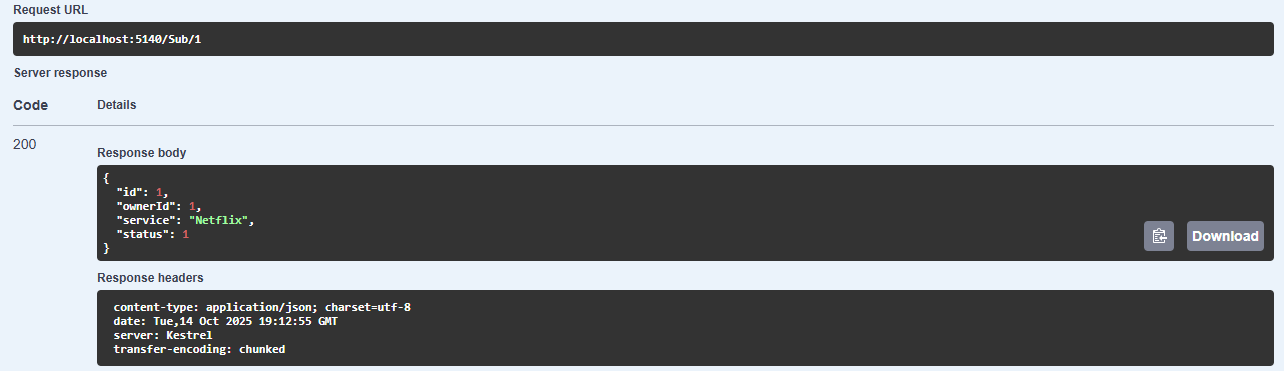


Рисунок 60 – Отримання підписки за Id

Запит 23

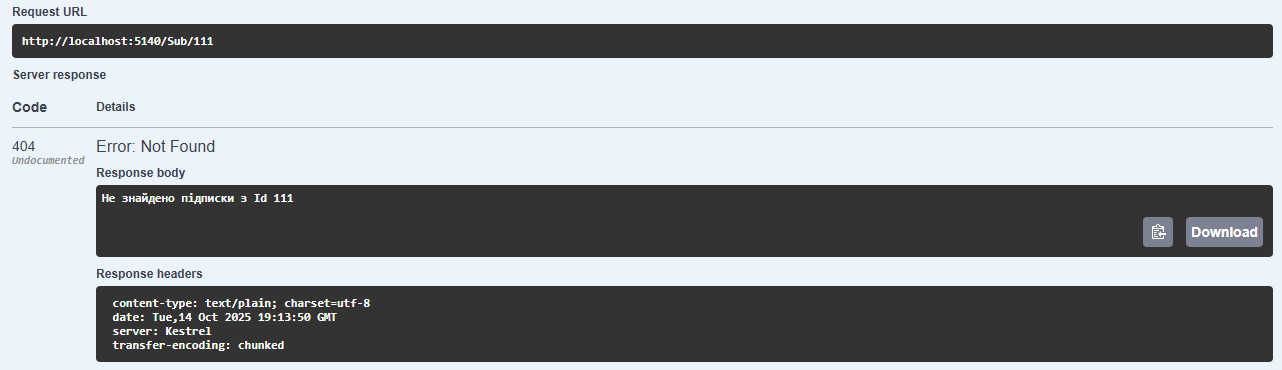


Рисунок 61 – Спроба отримання неіснуючої підписки за Id

Запит 24



Рисунок 62 – Додавання нової підписки

Запит 25



Рисунок 64 – Спроа додавання підписки із неіснуючим користувачем

Запит 26

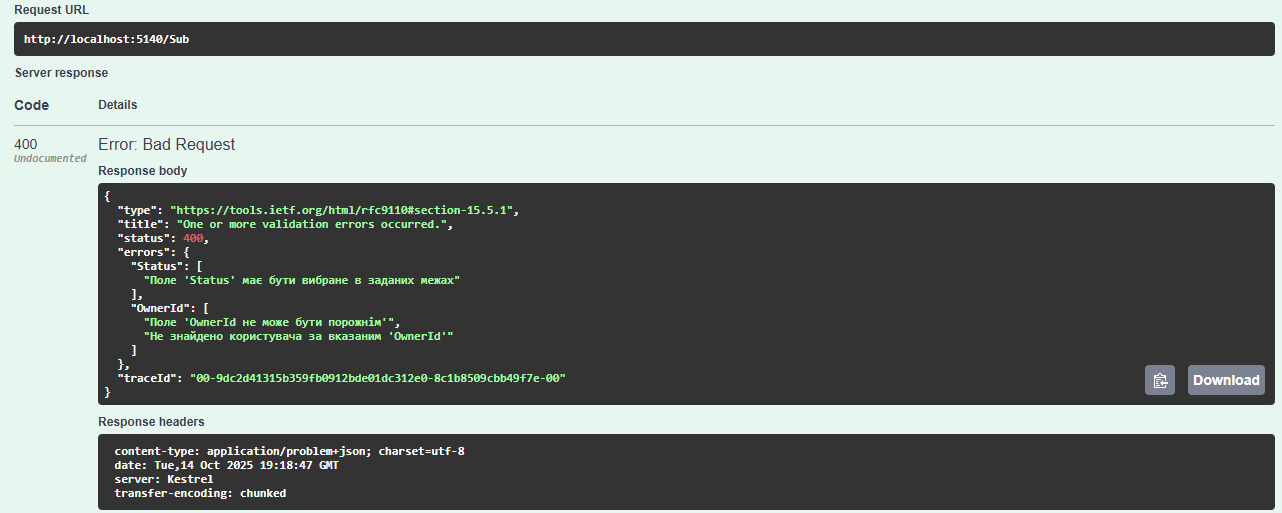


Рисунок 65 – Ще одна спроба додати невірну інормацію про підписку

Запит 27

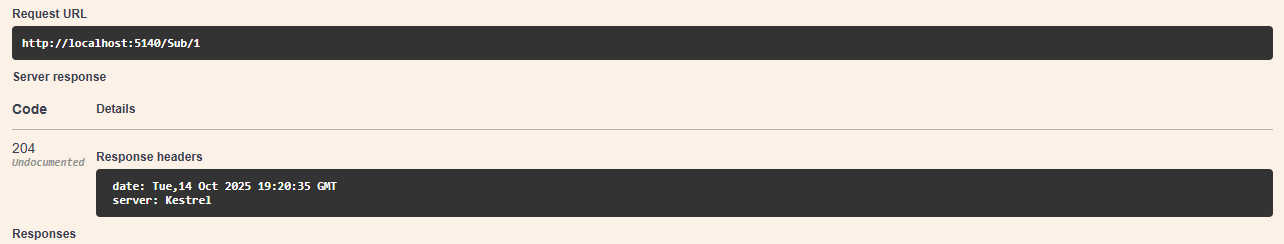


Рисунок 66 – Оновлення інформації про підпску

Запит 28



Рисунок 67 – Спроа некоректного оновлення підписки

Запит 29

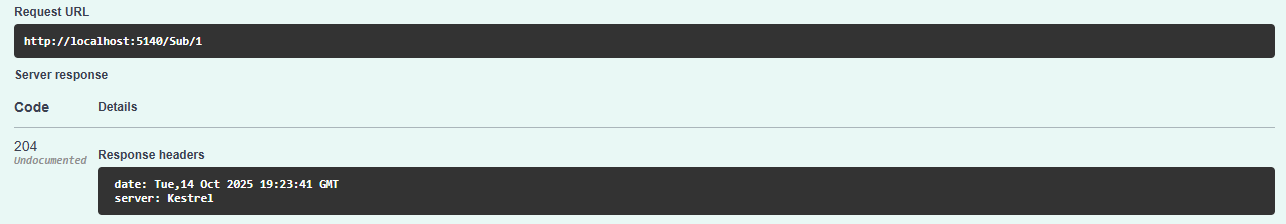


Рисунок 68 – Часткове оновлення власника підписки

Запит 30



Рисунок 69 – Спроба часткового оновлення підписки невірним користувачем

Запит 31

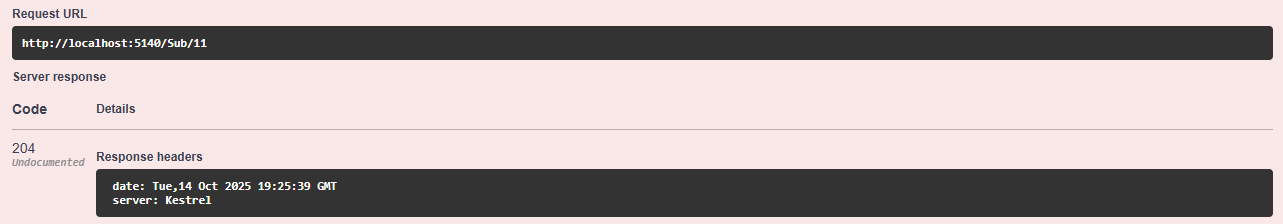


Рисунок 70 – Видалення підписки за Id

Запит 32

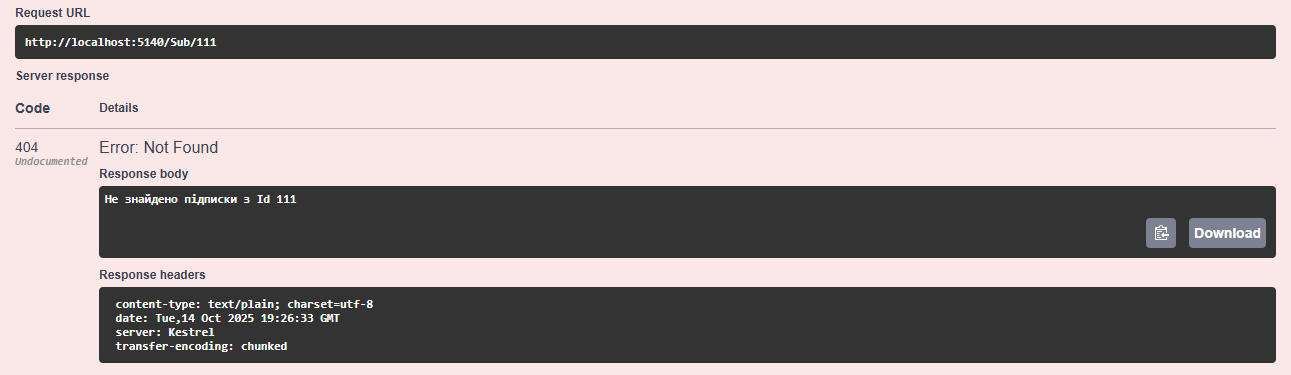


Рисунок 71 – Спроба видалення неіснуючої підписки

Запит 33

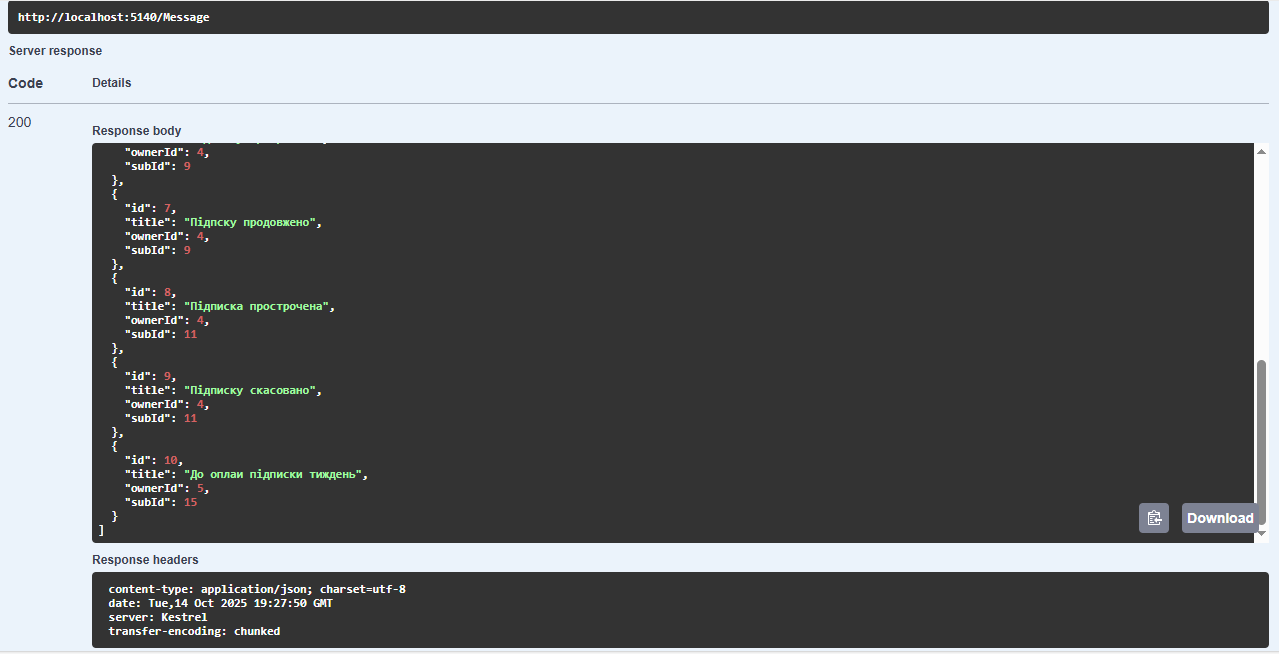


Рисунок 72 – Отримання списку всіх повідомлень

Запит 34

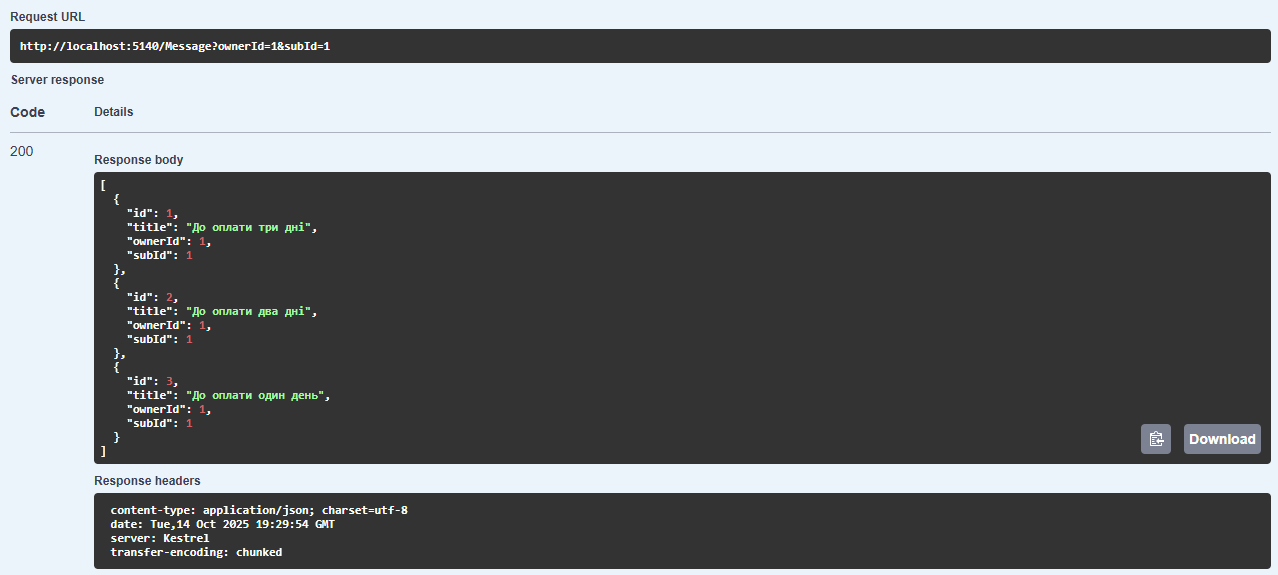


Рисунок 73 – Отримання повідомлень за вказаними парамерами

Запит 35



Рисунок 74 – Спроба отримати неіснуюче повідомлення

Запит 36



Рисунок 75 – Отримання повідомлення за Id

Запит 37



Рисунок 76 – Спроба отримання неіснуючого повідомлення за Id

Запит 38



Рисунок 77 – Додавання нової підписки

Запит 39



Рисунок 78 – Спроба додавання повідомлення про підписку не тому користувачеві та з порожніс текстом

Запит 40



Рисунок 79 – Коректне оновлення повідомлення

Запит 41



Рисунок 80 – Спроба оновити повідомлення, присвоївши підписку не ому користувачеві та з порожнім підписом

Запит 42



Рисунок 81 – Коректне часткове оновлення

Запит 43



Рисунок 82 – Спроба часткового оновлення повідомлення пустим тектом

Запит 44



Рисунок 83 – Видалення повідомлення

Запит 45

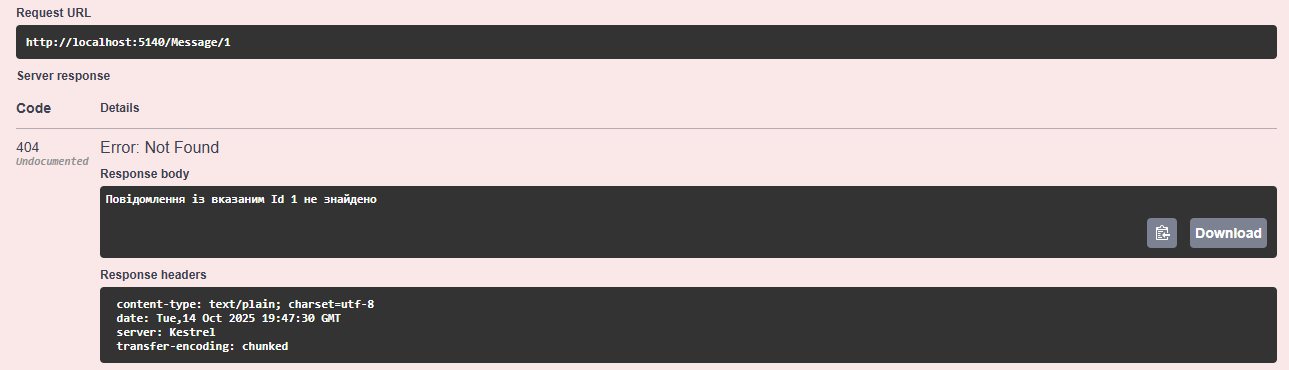


Рисунок 84 – Спроба видалення повідолення ще раз

Запит 45

Запит 46

Запит 47

Запит 48

Запит 49

Запит 50