

Лабораторна робота №4

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

Завдання 1

Ознайомитися з кроками створення **ASP.NET Core Minimal API** | [Check out the](#).

Створити **ASP.NET Core Minimal API**:

1. Створити **3–4 групи маршрутів** відповідно до Додатку А (наприклад: /todos, /users, /categories).
2. Використати **in-memory “сховище”** (простий список List в коді) для зберігання даних.
3. Реалізувати **CRUD** для кожної групи маршрутів:
 - **Create** (POST) — створення нового об’єкта.
 - **Read** (GET) — отримання списку об’єктів або одного об’єкта за ID.
 - **Update** (PUT/PATCH) — оновлення існуючого об’єкта.
 - **Delete** (DELETE) — видалення об’єкта за ID.
4. Виконати **валідацію даних** під час створення нових об’єктів (наприклад, перевірка порожніх полів, мінімальної довжини рядка, формату email тощо).
5. Повертає стандартні HTTP-відповіді:
 - 200 OK — успішний запит,
 - 201 Created — створення нового ресурсу,
 - 400 Bad Request — помилка валідації,
 - 404 Not Found — об’єкт не знайдено.
6. Для кожної групи створити **окремий набір ендпойнтів** у коді.
7. **Винести ендпойнти в окремі файли** (наприклад: Endpoints/ToDoEndpoints.cs, Endpoints/UserEndpoints.cs) і підключити їх у Program.cs.
8. **Організувати структуру проєкту** за папками:
/Models
/Endpoints
/Data (за потреби)
Program.cs
9. **Додати Swagger (OpenAPI)** для зручного тестування API через браузер:

					ЛР.ОК24.ПІ231.13.04					
Змн.	Арк.	№ докум.	Підпис	Дата						
Розроб.		Євчук М.В.			Створення RESTful API			Лім.	Арк.	Акрушів
Перевір.		Жереб Д.В.							1	
								ХПК		
Н. Контр.										
Затверд.										

10.Перевірити роботу API через **Swagger** або **Postman**.

Завдання 2

Ознайомитися з кроками створення **ASP.NET Core Web API** | [Check out the.](#)

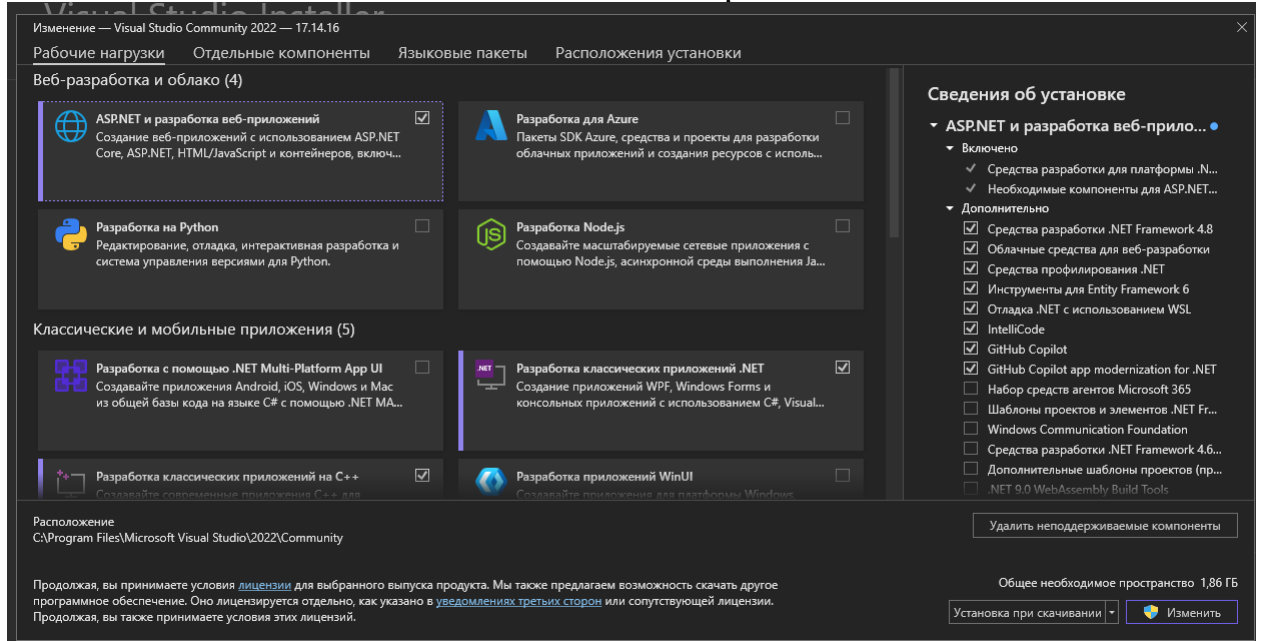
Створити **ASP.NET Core Web API**:

1. Створити **3–4 контролера** відповідно до Додатку А(наприклад: `TodoController`, `UserController`, `CategoryController`).
2. Створити відповідні моделі даних. Використати `Enum`, як поле класу. [Про Enum тут](#)
3. Використати **in-memory “сховище”** (простий список `List` в коді) для зберігання даних.
4. Реалізувати **CRUD** для кожної моделі в контролерах:
 - **Create** (POST) — створення нового об’єкта.
 - **Read** (GET) — отримання списку об’єктів або одного об’єкта за ID.
 - **Update** (PUT/PATCH) — оновлення існуючого об’єкта.
 - **Delete** (DELETE) — видалення об’єкта за ID.
5. Виконати **валідацію даних** під час створення нових об’єктів (наприклад, перевірка порожніх полів, мінімальної довжини рядка, формату email тощо). Для однієї моделі (самої меншої за к-тю полів) виконати валідацію через `DataAnnotations`. Для інших моделей виконати валідацію через `FluentValidation`. Для одного з полів використати [Regular Expressions](#)
6. Повертає стандартні HTTP-відповіді:
 - 200 OK — успішний запит,
 - 201 Created — створення нового ресурсу,
 - 400 Bad Request — помилка валідації,
 - 404 Not Found — об’єкт не знайдено.
7. **Організувати структуру проєкту** за папками:
/Models
/Controllers
/Data (за потреби)
/Validators
Program.cs
8. Перевірити роботу API через **Swagger** або **Postman**.
9. Підготувати **звіт**, який містить:
 - короткий опис проєкту;
 - код `Program.cs`, моделей та ендпойнтів;
 - скріншоти виконання запитів (GET, POST, помилки валідації);

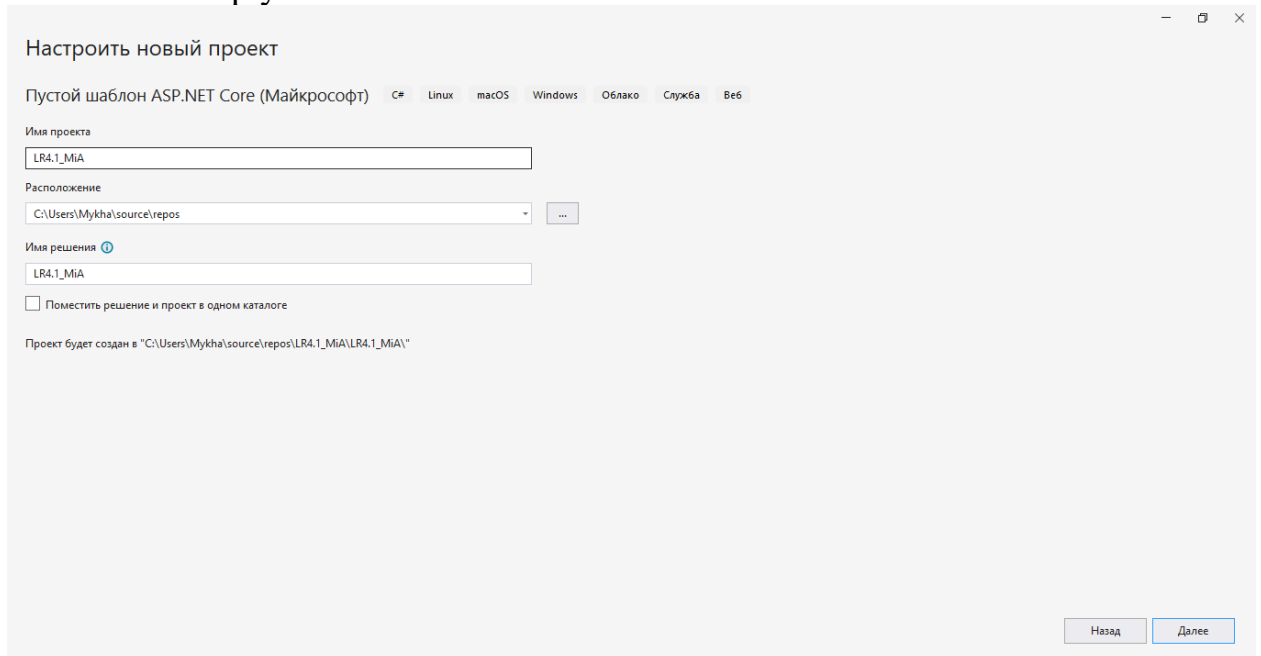
Хід роботи

40.Онлайн-сервіс для обміну кулінарними рецептами з рейтингами

1. Під час встановлення Visual Studio обираємо ASP.NET



2. Перейдемо до створення проєкту. Для цього знаходимо проєкт ASP.NET Core empty.

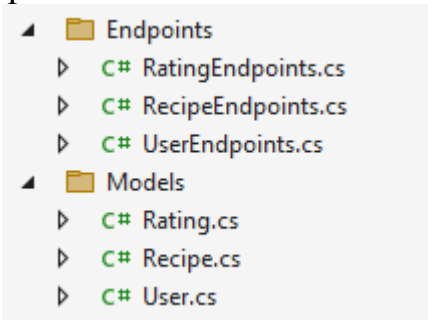


Перейдемо до написання програми.

Створимо папки Models, Endpoints. В папці Models створюємо 3 класа, а саме User.cs, Recipe.cs, Rating.cs. В папці Endpoints створюємо також 3 класа, а саме UserEndpoints.cs, RecipeEndpoints.cs, RatingEndpoints.cs.

					ЛР.ОК24.ПІ231.13.04	Арк.
						3
Змн.	Арк.	№ докум.	Підпис	Дата		

Ось як виглядає наша ієрархія.



Тепер перейдемо до написання коду.

Models

User.cs

```
namespace LW4.Task1_MiA.Models
{
    public class User
    {
        public int Id { get; set; }
        public string Username { get; set; }
        public string Email { get; set; }
    }
}
```

Recipe.cs

```
namespace LW4.Task1_MiA.Models
{
    public class Recipe
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Instructions { get; set; }
        public List<Rating> Ratings { get; set; }
    }
}
```

Rating.cs

```
namespace LW4.Task1_MiA.Models
{
    public class Rating
    {
        public int Id { get; set; }
        public int RecipeId { get; set; }
        public int UserId { get; set; }
        public int Score { get; set; }
        public string Review { get; set; }
    }
}
```

Endpoints

UserEndpoints.cs

```
using LW4.Task1_MiA.Models;
namespace LW4.Task1_MiA.Endpoints
{
    public static class UserEndpoints
    {
        public static void MapUserEndpoints(this WebApplication app)
        {
            var users = new List<User>
            {
                new User { Id = 1, Username = "john_doe", Email = "john@example.com" },
                new User { Id = 2, Username = "jane_smith", Email = "jane@example.com" },
            },
        }
    }
}
```

```

        new User { Id = 3, Username = "alice_wonder", Email =
"alice@example.com" }
    };
    app.MapGet("/users", () => Results.Ok(users));
    app.MapGet("/users/{id:int}", (int id) =>
    {
        var user = users.FirstOrDefault(u => u.Id == id);
        return user is not null ? Results.Ok(user) : Results.NotFound();
    });

    app.MapPost("/users", (User newUser) =>
    {
        newUser.Id = users.Count + 1;
        users.Add(newUser);
        return Results.Created($"/users/{newUser.Id}", newUser);
    });

    app.MapPut("/users/{id:int}", (int id, User updatedUser) =>
    {
        var user = users.FirstOrDefault(u => u.Id == id);
        if (user is null) return Results.NotFound();
        user.Username = updatedUser.Username;
        user.Email = updatedUser.Email;
        return Results.Ok(user);
    });

    app.MapDelete("/users/{id:int}", (int id) =>
    {
        var user = users.FirstOrDefault(u => u.Id == id);
        if (user is null) return Results.NotFound();
        users.Remove(user);
        return Results.NoContent();
    });
}
}
}

```

RecipeEndpoints.cs

```

using LW4.Task1_MiA.Models;
namespace LW4.Task1_MiA.Endpoints
{
    public static class RecipeEndpoints
    {
        public static void MapRecipeEndpoints(this WebApplication app)
        {
            var recipes = new List<Recipe>
            {
                new Recipe { Id = 1, Name = "Pasta", Instructions = "Boil water, add
pasta" },
                new Recipe { Id = 2, Name = "Salad", Instructions = "Mix vegetables"
},
                new Recipe { Id = 3, Name = "Sandwich", Instructions = "Put
ingredients between bread slices" },
                new Recipe { Id = 4, Name = "Omelette", Instructions = "Beat eggs,
cook in a pan" },
                new Recipe { Id = 5, Name = "Soup", Instructions = "Boil broth, add
ingredients" },
                new Recipe { Id = 6, Name = "Steak", Instructions = "Season meat,
cook to desired doneness" },
                new Recipe { Id = 7, Name = "Pancakes", Instructions = "Mix batter,
cook on griddle" },
                new Recipe { Id = 8, Name = "Tacos", Instructions = "Fill tortillas
with ingredients" },
                new Recipe { Id = 9, Name = "Pizza", Instructions = "Prepare dough,
add toppings, bake" },
                new Recipe { Id = 10, Name = "Curry", Instructions = "Cook spices,
add meat/vegetables, simmer" },
            };
        }
    }
}

```

					ЛР.ОК24.ПІ231.13.04	Арк.
						5
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        new Recipe { Id = 11, Name = "Grilled Cheese", Instructions =
" Butter bread, add cheese, grill" },
        new Recipe { Id = 12, Name = "Fruit Smoothie", Instructions = "Blend
fruits with yogurt or milk" },
        new Recipe { Id = 13, Name = "Roast Chicken", Instructions = "Season
chicken, roast in oven" },
    };
    app.MapGet("/recipes", () => Results.Ok(recipes));
    app.MapGet("/recipes/{id:int}", (int id) =>
    {
        var recipe = recipes.FirstOrDefault(r => r.Id == id);
        return recipe is not null ? Results.Ok(recipe) : Results.NotFound();
    });
    app.MapPost("/recipes", (Recipe newRecipe) =>
    {
        // Перевірка: чи заповнене поле Name
        if (string.IsNullOrEmpty(newRecipe.Name))
            return Results.BadRequest("Recipe name cannot be empty.");
        // Додавання нового рецепту
        newRecipe.Id = recipes.Count + 1;
        recipes.Add(newRecipe);
        return Results.Created($" /recipes/{newRecipe.Id}", newRecipe);
    });
    app.MapPut("/recipes/{id:int}", (int id, Recipe updatedRecipe) =>
    {
        var recipe = recipes.FirstOrDefault(r => r.Id == id);
        if (recipe is null) return Results.NotFound();
        recipe.Name = updatedRecipe.Name;
        recipe.Instructions = updatedRecipe.Instructions;
        return Results.Ok(recipe);
    });
    app.MapDelete("/recipes/{id:int}", (int id) =>
    {
        var recipe = recipes.FirstOrDefault(r => r.Id == id);
        if (recipe is null) return Results.NotFound();
        recipes.Remove(recipe);
        return Results.NoContent();
    });
}
}
}

```

RatingEndpoints.cs

```

using LW4.Task1_MiA.Models;
namespace LW4.Task1_MiA.Endpoints {
    public static class RatingEndpoints
    {
        public static void MapRatingEndpoints(this WebApplication app)
        {
            var ratings = new List<Rating>
            {
                new Rating { Id = 1, RecipeId = 1, UserId = 1, Score = 5, Review =
"Delicious!" },
                new Rating { Id = 2, RecipeId = 2, UserId = 2, Score = 4, Review =
"Tasty, but could use more dressing." },
                new Rating { Id = 3, RecipeId = 3, UserId = 3, Score = 3, Review =
"Average." },
                new Rating { Id = 4, RecipeId = 1, UserId = 2, Score = 4, Review =
"Pretty good!" },
                new Rating { Id = 5, RecipeId = 2, UserId = 3, Score = 5, Review =
"Loved it!" },
                new Rating { Id = 6, RecipeId = 3, UserId = 1, Score = 2, Review =
"Not my favorite." },
                new Rating { Id = 7, RecipeId = 1, UserId = 3, Score = 5, Review =
"Absolutely fantastic!" },
            };
        }
    }
}

```

					ЛР.ОК24.ПІ231.13.04	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

```

app.MapGet("/ratings", () => Results.Ok(ratings));
app.MapGet("/ratings/{id:int}", (int id) =>
{
    var rating = ratings.FirstOrDefault(r => r.Id == id);
    return rating is not null ? Results.Ok(rating) : Results.NotFound();
});
app.MapPost("/ratings", (Rating newRating) =>
{
    newRating.Id = ratings.Count + 1;
    ratings.Add(newRating);
    return Results.Created($"/ratings/{newRating.Id}", newRating);
});
app.MapPut("/ratings/{id:int}", (int id, Rating updatedRating) =>
{
    var rating = ratings.FirstOrDefault(r => r.Id == id);
    if (rating is null) return Results.NotFound();
    rating.Score = updatedRating.Score;
    rating.Review = updatedRating.Review;
    return Results.Ok(rating);
});
app.MapDelete("/ratings/{id:int}", (int id) =>
{
    var rating = ratings.FirstOrDefault(r => r.Id == id);
    if (rating is null) return Results.NotFound();
    ratings.Remove(rating);
    return Results.NoContent();
});
}
}
}

```

Program.cs

```

using LW4.Task1_MiA.Endpoints;

var builder = WebApplication.CreateBuilder(args);

// Swagger
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
var app = builder.Build();
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI(opt =>
    {
        opt.SwaggerEndpoint("/swagger/v1/swagger.json", "Recipes API v1");
        opt.RoutePrefix = "swagger";
    });
}
app.MapRecipeEndpoints();
app.MapUserEndpoints();
app.MapRatingEndpoints();

app.Run();

```

					ЛР.ОК24.ПІ231.13.04	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		7

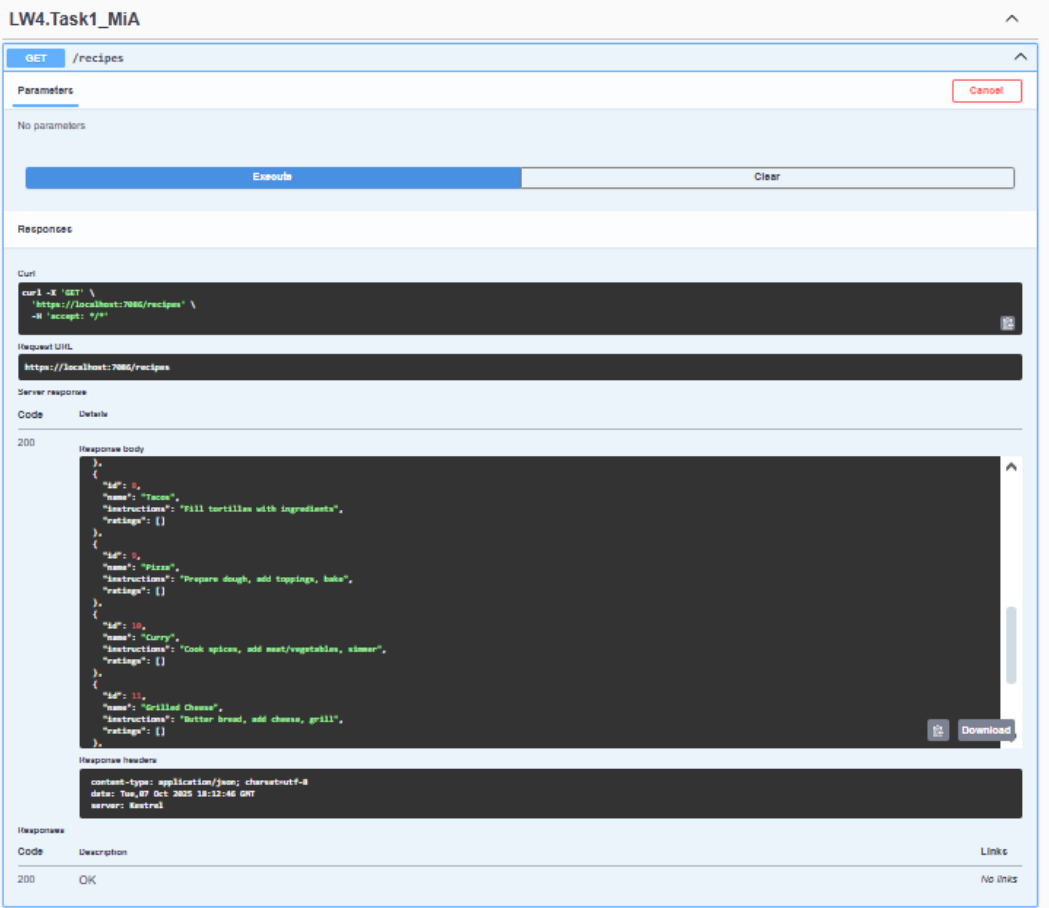
Результат роботи:

```
info: Microsoft.Hosting.Lifetime[14]
Now listening on: https://localhost:7086
info: Microsoft.Hosting.Lifetime[14]
Now listening on: http://localhost:5128
info: Microsoft.Hosting.Lifetime[0]
Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
Content root path: C:\Users\Mykha\source\repos\LW4.Task1_MiA\LW4.Task1_MiA
```

Програма показує те, що нам доступна вся інформація по посиланню.

Запити

Запит на отримання рецептів



Запит на публікацію рецепта

POST /recipes

Рекордів: 1

Cancel

Save

Request body

application/json

Full Name | Schema

```
{
  "id": 1,
  "name": "Borscht",
  "instructions": "To prepare classic borscht, first boil the beets, then add chopped potatoes and cabbage, and finally add a dressing of sautéed beef, carrots, onions, and tomato paste. It is important to preserve the bright color of the beets by adding citric acid or lemon juice during cooking. Borscht is served with sour cream and herbs, letting it steep before serving.",
  "ratings": [
    {
      "id": 1,
      "rating": 5,
      "comment": "Great!"
    },
    {
      "id": 2,
      "rating": 4,
      "comment": "Nice!"
    },
    {
      "id": 3,
      "rating": 5,
      "comment": "It's amazing borscht!"
    }
  ]
}
```

Cancel

Clear

Рекордів: 1

Full Name | Schema

```
curl -X POST \
  https://localhost:7000/recipes \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 1,
    "name": "Borscht",
    "instructions": "To prepare classic borscht, first boil the beets, then add chopped potatoes and cabbage, and finally add a dressing of sautéed beef, carrots, onions, and tomato paste. It is important to preserve the bright color of the beets by adding citric acid or lemon juice during cooking. Borscht is served with sour cream and herbs, letting it steep before serving.",
    "ratings": [
      {
        "id": 1,
        "rating": 5,
        "comment": "Great!"
      },
      {
        "id": 2,
        "rating": 4,
        "comment": "Nice!"
      },
      {
        "id": 3,
        "rating": 5,
        "comment": "It's amazing borscht!"
      }
    ]
  }'
```

Cancel

Clear

Response (201)

https://localhost:7000/recipes/1

Response headers

Code

Details

201

Request body

```
{
  "id": 1,
  "name": "Borscht",
  "instructions": "To prepare classic borscht, first boil the beets, then add chopped potatoes and cabbage, and finally add a dressing of sautéed beef, carrots, onions, and tomato paste. It is important to preserve the bright color of the beets by adding citric acid or lemon juice during cooking. Borscht is served with sour cream and herbs, letting it steep before serving.",
  "ratings": [
    {
      "id": 1,
      "rating": 5,
      "comment": "Great!"
    },
    {
      "id": 2,
      "rating": 4,
      "comment": "Nice!"
    },
    {
      "id": 3,
      "rating": 5,
      "comment": "It's amazing borscht!"
    }
  ]
}
```

Download

Response headers

```
Content-Type: application/json; charset=utf-8
Date: Tue, 27 Dec 2023 15:15:08 GMT
Server: Express/4.18.2
x-powered-by: Express
```

Response

Code

Details

200

OK

No links

Запит на отримання конкретного рецепта

GET /recipes/{id}

Рекордів: 1

Cancel

Request body

application/json

Full Name | Schema

```
{
  "id": 1,
  "name": "Borscht",
  "instructions": "To prepare classic borscht, first boil the beets, then add chopped potatoes and cabbage, and finally add a dressing of sautéed beef, carrots, onions, and tomato paste. It is important to preserve the bright color of the beets by adding citric acid or lemon juice during cooking. Borscht is served with sour cream and herbs, letting it steep before serving.",
  "ratings": [
    {
      "id": 1,
      "rating": 5,
      "comment": "Great!"
    },
    {
      "id": 2,
      "rating": 4,
      "comment": "Nice!"
    },
    {
      "id": 3,
      "rating": 5,
      "comment": "It's amazing borscht!"
    }
  ]
}
```

Cancel

Clear

Рекордів: 1

Full Name | Schema

```
curl -X GET \
  https://localhost:7000/recipes/1 \
  -H 'accept: */*' \
  -H 'Content-Type: application/json'
```

Cancel

Clear

Response (200)

https://localhost:7000/recipes/1

Response headers

Code

Details

200

Request body

```
{
  "id": 1,
  "name": "Borscht",
  "instructions": "To prepare classic borscht, first boil the beets, then add chopped potatoes and cabbage, and finally add a dressing of sautéed beef, carrots, onions, and tomato paste. It is important to preserve the bright color of the beets by adding citric acid or lemon juice during cooking. Borscht is served with sour cream and herbs, letting it steep before serving.",
  "ratings": [
    {
      "id": 1,
      "rating": 5,
      "comment": "Great!"
    },
    {
      "id": 2,
      "rating": 4,
      "comment": "Nice!"
    },
    {
      "id": 3,
      "rating": 5,
      "comment": "It's amazing borscht!"
    }
  ]
}
```

Download

Response headers

```
Content-Type: application/json; charset=utf-8
Date: Tue, 27 Dec 2023 15:15:08 GMT
Server: Express/4.18.2
x-powered-by: Express
```

Response

Code

Details

200

OK

No links

Запит на оновлення рецепта

The screenshot shows the AWS IAM console interface. At the top, there's a navigation bar with 'Users' and 'Groups' tabs. The 'Users' tab is active, and a user named 'aws-logs' is selected. Below the navigation bar, there's a 'Permissions' tab selected, showing a list of permissions. The 'aws-logs' permission is highlighted. The 'Permissions' tab is also selected in the top navigation bar. The 'Permissions' tab is selected in the top navigation bar. The 'Permissions' tab is selected in the top navigation bar.

Запит на видалення рецепта

DELETE

/recipies/{id}

Cancel

Parameters

Name

Search/Filter

Id

integer (32-bit)

1

Cancel

Clear

Responses

200

Content

```
curl -d '{"recipe": {"title": "MILKSHAKE", "ingredients": [{"ingredient": "MILKSHAKE"}]}}' http://localhost:3000/recipies/1
```

Response (JSON)

```
{ "recipe": { "id": 1, "title": "MILKSHAKE", "ingredients": [{"ingredient": "MILKSHAKE"}] } }
```

201

Content

```
curl -d '{"recipe": {"title": "MILKSHAKE", "ingredients": [{"ingredient": "MILKSHAKE"}]}}' http://localhost:3000/recipies/
```

Response (JSON)

```
{ "recipe": { "id": 2, "title": "MILKSHAKE", "ingredients": [{"ingredient": "MILKSHAKE"}] } }
```

Запит на отримання користувачів

GET /users

Cancel

No parameters

Cancel

Clear

Request

URL

```
url: <URL> \
  headers: { "Host": "192.168.1.100" } \
  <body> <body>
```

Response URL

```
https://192.168.1.100/users
```

Response headers

Code

Details

200

Response body

```
{
  "id": 1,
  "username": "john_doe",
  "email": "john.doe@example.com"
},
{
  "id": 2,
  "username": "jane_smith",
  "email": "jane.smith@example.com"
},
{
  "id": 3,
  "username": "alice_jones",
  "email": "alice.jones@example.com"
}
```

Download

Response footer

```
Content-Type: application/json; charset=utf-8
Date: Tue, 27 Oct 2020 10:14:11 GMT
Server: Apache/2.4.18
```

Response

Code

Details

Links

200

OK

No links

Запит на додавання користувача

POST /users

Cancel

Reset

No parameters

Request body

application/json

Full Value | Schema

```
{
  "id": 1,
  "username": "john_doe",
  "email": "john.doe@example.com"
}
```

Cancel

Clear

Request

URL

```
url: <URL> \
  headers: { "Host": "192.168.1.100" } \
  <body> <body>
```

Response URL

```
https://192.168.1.100/users
```

Response headers

Code

Details

201

Response body

```
{
  "id": 1,
  "username": "john_doe",
  "email": "john.doe@example.com"
}
```

Download

Response footer

```
Content-Type: application/json; charset=utf-8
Date: Tue, 27 Oct 2020 10:14:11 GMT
Server: Apache/2.4.18
```

Response

Code

Details

Links

200

OK

No links

Запит на отримання конкретного користувача

GET /users/{id}

Parameters

Cancel

Name	Description
id *required	
integer(int32)	3
(path)	

Execute

Clear

Responses

Curl

```
curl -X 'GET' \
  'https://localhost:7060/users/3' \
  -H 'accept: */*'
```

Request URL

https://localhost:7060/users/3

Server response

Code	Details
200	<div>Response body</div> <div><pre>{ "id": 3, "username": "alice_wonder", "email": "alice@example.com" }</pre></div> <div>Response headers</div> <div><pre>content-type: application/json; charset=utf-8 date: Tue, 07 Oct 2025 18:16:42 GMT server: Kestrel</pre></div>

Response

Code	Description	Links
200	OK	No links

Запит на оновлення користувача

PUT /users/{id}

Parameters

Cancel

Reset

Name	Description
id *required	
integer(int32)	2
(path)	

Request body *required

application/json

Full Value | Syntax

```
{
  "id": 12,
  "username": "bob",
  "email": "bob@example.com"
}
```

Submit

Clear

Responses

Curl

```
curl -X 'PUT' \
  'https://localhost:7060/users/2' \
  -H 'content-type: application/json' \
  -d '{
    "id": 12,
    "username": "bob",
    "email": "bob@example.com"
  }'
```

Request URL

https://localhost:7060/users/2

Server response

Code	Details
200	<div>Response body</div> <div><pre>{ "id": 12, "username": "bob", "email": "bob@example.com" }</pre></div> <div>Response headers</div> <div><pre>content-type: application/json; charset=utf-8 date: Tue, 07 Oct 2025 18:17:00 GMT server: Kestrel</pre></div>

Response

Code	Description	Links
200	OK	No links

Змн.	Арк.	№ докум.	Підпис	Дата

Запит на видалення користувача

SELECT

/users/{id}

Cancel

Parameters

Name

Description

id

integer(1000)

1

Clear

Clear

Responses

200

```

{
  "id": "12345678",
  "username": "john.doe@example.com",
  "email": "john.doe@example.com"
}

```

201

```

{
  "id": "12345678",
  "username": "john.doe@example.com",
  "email": "john.doe@example.com"
}

```

204

```

{
  "id": "12345678",
  "username": "john.doe@example.com",
  "email": "john.doe@example.com"
}

```

Запит на отримання оцінок

[illegible]

Змн.	Арк.	№ докум.	Підпис	Дата

ЛР.ОК24.ПІ231.13.04

Арк.

13

Запит на додавання оцінки

POST /ratings

Cancel

Результат

No responses

Request body:

application/json

Raw Value

JSON

```
{  "id": 1,  "userId": 12,  "score": 5,  "comment": "Great!"}
```

Скопіювати

Clear

Відповісти

Code

Details

201

Created

Response body

Download

```
{  "id": 1,  "userId": 12,  "score": 5,  "comment": "Great!"}
```

Response headers

Content-Type: application/json; charset=utf-8

Date: Tue, 27 Oct 2020 10:27:48 GMT

Server: Python/3.6

Server: Docker

Response

Code

Details

200

OK

Links

No links

Запит на отримання конкретної оцінки

GET /ratings/{id}

Cancel

Результат

Header:

Details

Id:

1

Скопіювати

Clear

Відповісти

Code

Details

200

Response body

Download

```
{  "id": 1,  "userId": 12,  "score": 5,  "comment": "Great!"}
```

Response headers

Content-Type: application/json; charset=utf-8

Date: Tue, 27 Oct 2020 10:27:48 GMT

Server: Python/3.6

Server: Docker

Response

Code

Details

200

OK

Links

No links

Запит на оновлення оцінки

PUT/ratings/{id}

Результат

Id: integer(32x12) 5

Request body: application/json

```
{
  "id": 42,
  "reviewId": 88,
  "score": 12,
  "score": 4,
  "review": "test test"
}
```

Скажи

Clear

Вихід

Code

```
curl -X PUT \
  https://localhost:7000/ratings/5 \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 42,
    "reviewId": 88,
    "score": 12,
    "score": 4,
    "review": "test test"
  }'
```

Response: 200

Response body:

```
{
  "id": 42,
  "reviewId": 88,
  "score": 12,
  "score": 4,
  "review": "test test"
}
```

Response headers:

```
Content-Type: application/json; charset=utf-8
Date: Tue, 27 Oct 2025 18:58:12 GMT
Server: GoGoGo
```

Response:

Code	Description	Link
200	OK	No link

Запит на видалення оцінки

DELETE/ratings/{id}

Результат

Id: integer(32x12) 5

Request body: application/json

Скажи

Clear

Вихід

Code

```
curl -X DELETE \
  https://localhost:7000/ratings/5 \
  -H 'accept: */*' \
  -d '{
    "id": 42,
    "reviewId": 88,
    "score": 12,
    "score": 4,
    "review": "test test"
  }'
```

Response: 204

Response body:

```
{
  "id": 42,
  "reviewId": 88,
  "score": 12,
  "score": 4,
  "review": "test test"
}
```

Response headers:

```
Content-Type: application/json; charset=utf-8
Date: Tue, 27 Oct 2025 18:58:12 GMT
Server: GoGoGo
```

Response:

Code	Description	Link
200	OK	No link

					ЛР.ОК24.ПІ231.13.04	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		15