**Лабораторна робота №4**

**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 1**

Ознайомитися з кроками створення **ASP.NET Core Minimal API** | [Check out the](https://docs-mai.vercel.app/docs/developmement/minimal_api).

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

1. **Додати Swagger (OpenAPI)** для зручного тестування API через браузер:
2. Перевірити роботу API через **Swagger** або **Postman**.

**Завдання 2**

Ознайомитися з кроками створення **ASP.NET Core Web API** | [Check out the](https://docs-mai.vercel.app/docs/developmement/asp_net).

Створити **ASP.NET Core Web API**:

1. Створити **3–4 контролера** відповідно до Додатку А(наприклад: TodoController, UserController, CategoryController).
2. Створити відповідні моделі даних. Викорситати Enum, як поле класу. [Про Enum тут](https://docs-mai.vercel.app/docs/developmement/additionals/enum)
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 валідація через FluentValidation. Для одного з полів використати [Regular Expressions](https://docs-mai.vercel.app/docs/developmement/additionals/regex)
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:

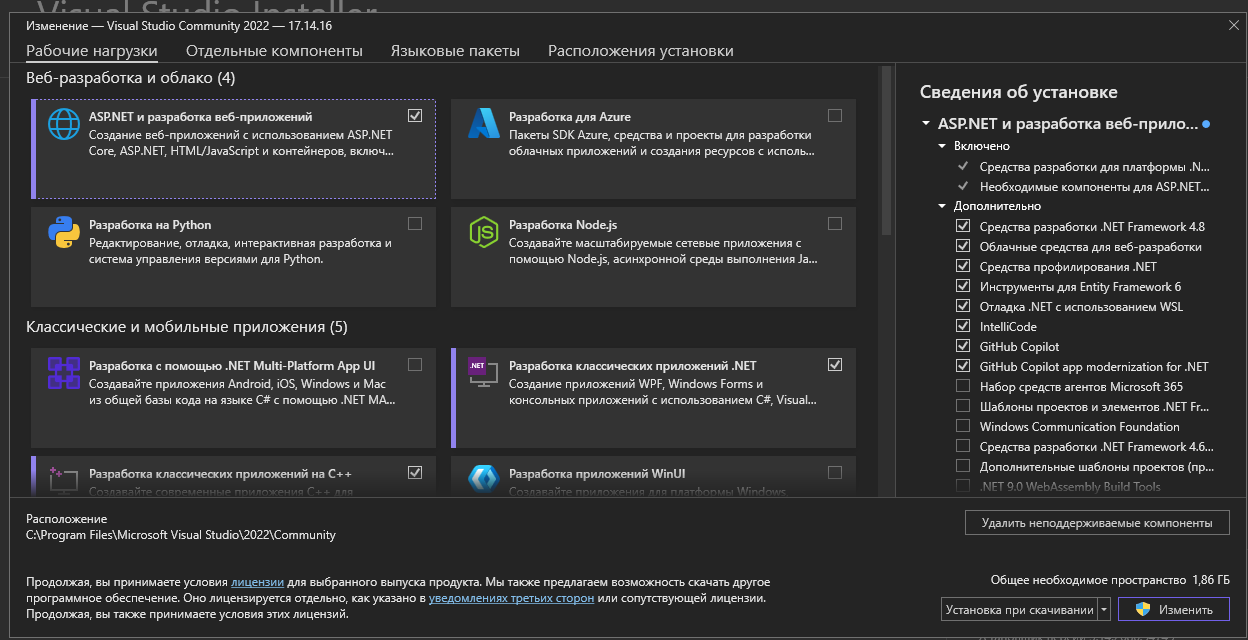
/Models  
/Controllers  
/Data (за потреби)  
/Validators  
Program.cs

1. Перевірити роботу API через **Swagger** або **Postman**.
2. Підготувати **звіт**, який містить:
   * короткий опис проєкту;
   * код Program.cs, моделей та ендпойнтів;
   * скріншоти виконання запитів (GET, POST, помилки валідації);

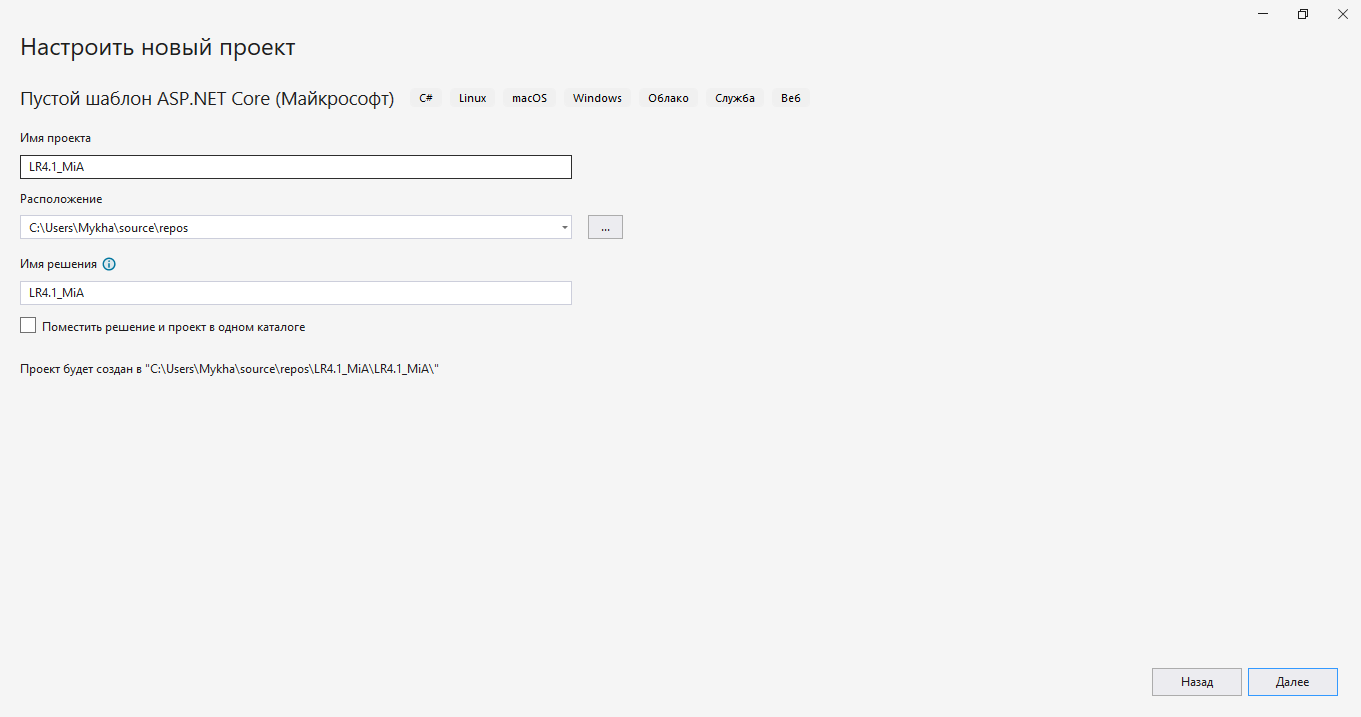
**Хід роботи**

**40.Онлайн-сервіс для обміну кулінарними рецептами з рейтингами**

1. Під час встановлення Visual Studio обираємо ASP.NET



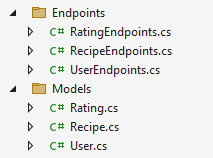
1. Перейдемо до створення проєкту. Для цього знаходимо проєкт ASP.NET Core empty.



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.

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" },

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.IsNullOrWhiteSpace(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!" },

};

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";

});

}

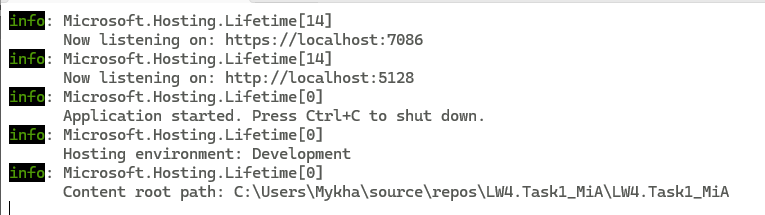
app.MapRecipeEndpoints();

app.MapUserEndpoints();

app.MapRatingEndpoints();

app.Run();

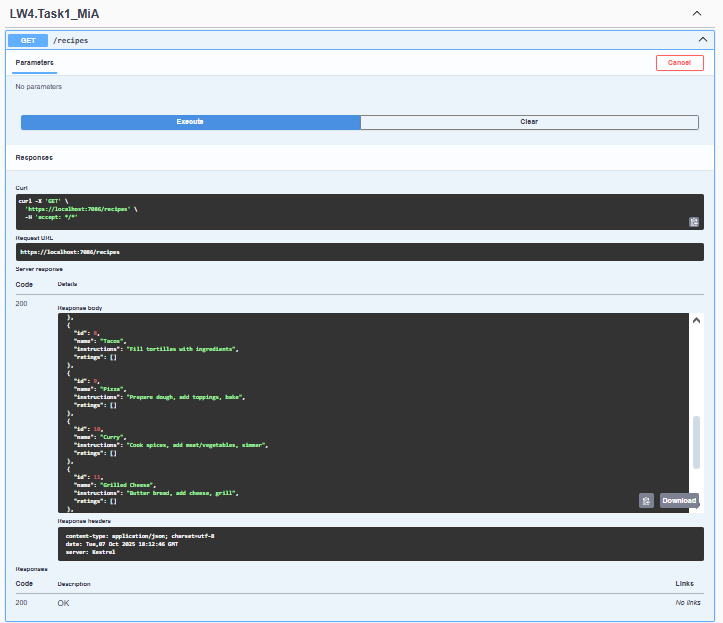
**Результат роботи:**

****

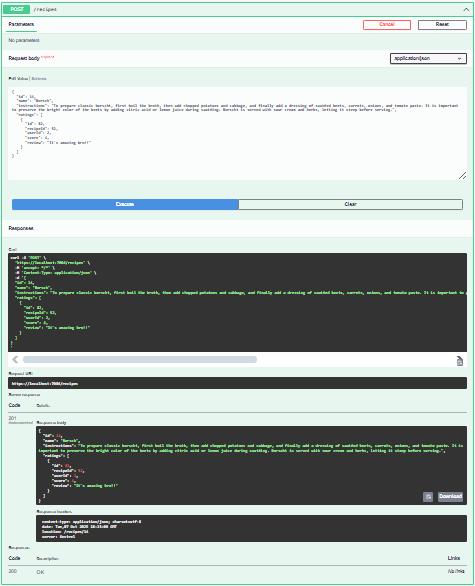
Програма показує те, що нам доступна вся інформація по посиланню.

**Запити**

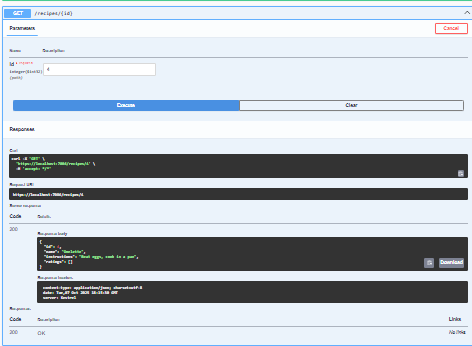
**Запит на отримання рецептів**



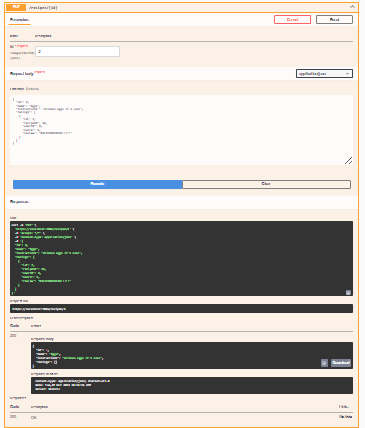
**Запит на публікацію рецепта**



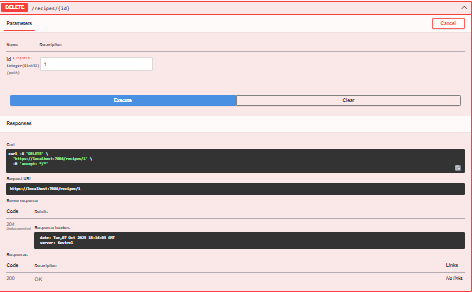
**Запит на отримання конкретного рецепта**

****

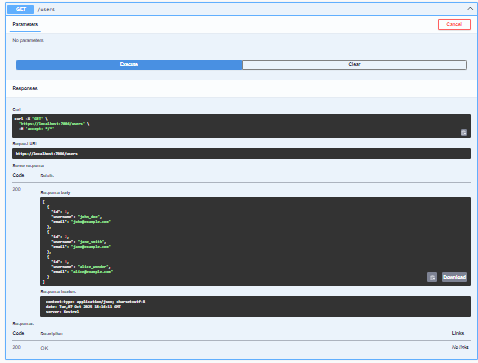
**Запит на оновлення рецепта**

****

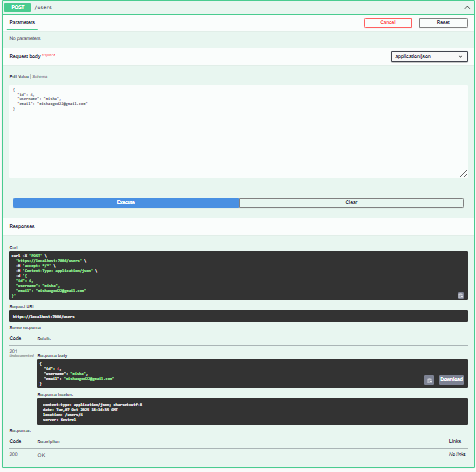
**Запит на видалення рецепта**

****

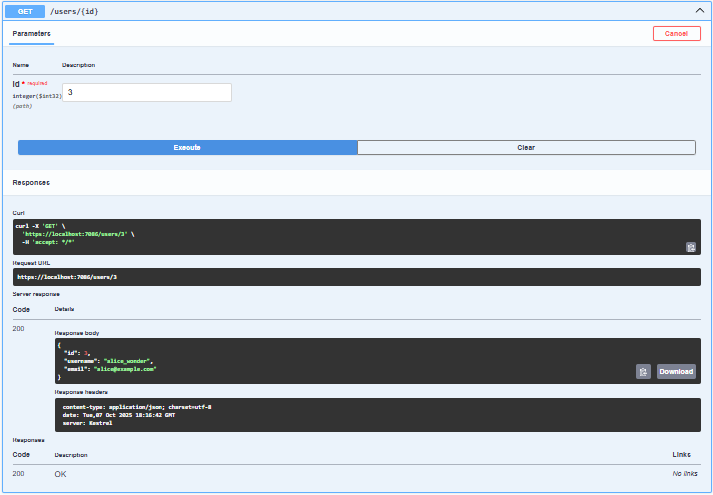
**Запит на отримання користувачів**

****

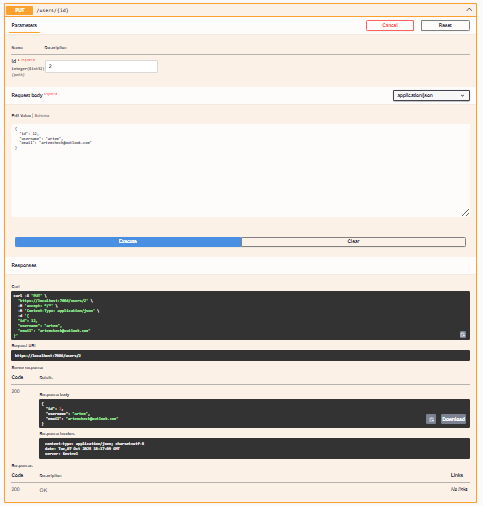
**Запит на додавання користувача**

****

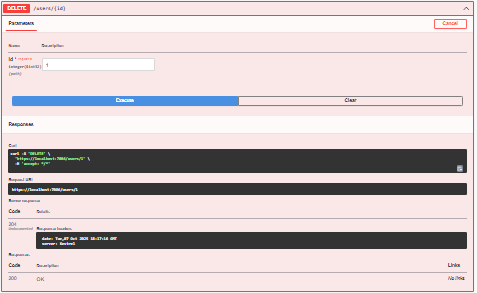
**Запит на отримання конкретного користувача**

****

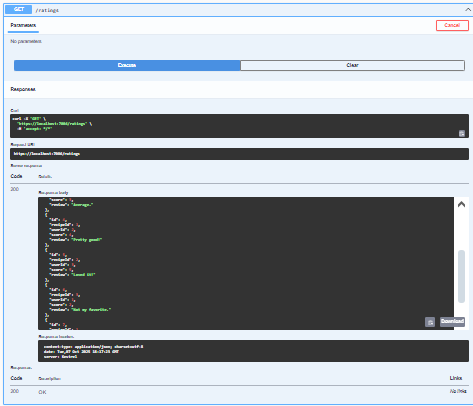
**Запит на оновлення користувача**

****

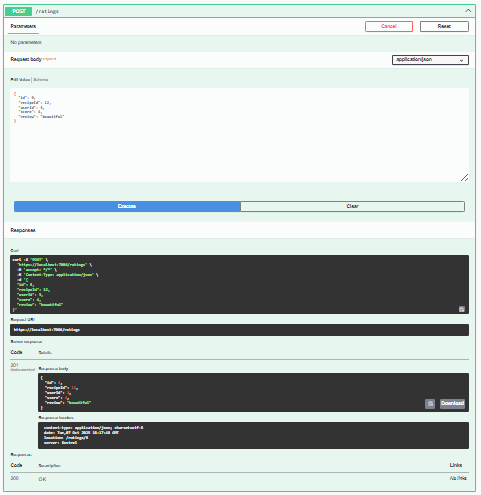
**Запит на видалення користувача**

****

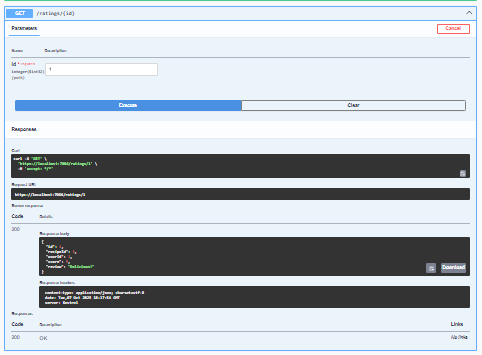
**Запит на отримання оцінок**

****

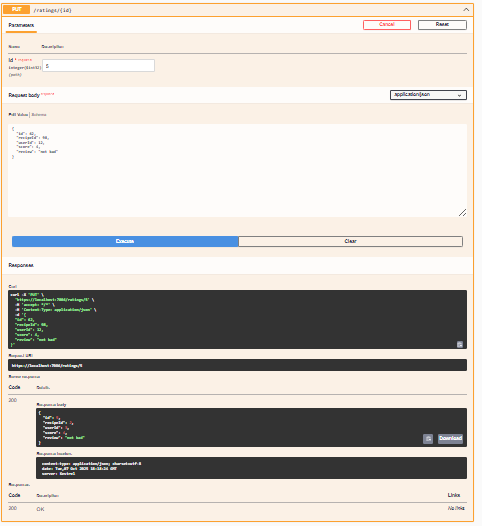
**Запит на додавання оцінки**

****

**Запит на отримання конкретної оцінки**

****

**Запит на оновлення оцінки**



**Запит на видалення оцінки**

