**Creating and Exploring ASP.NET Core Web API**

**Building a Simple ASP.NET Core Web API**

By the end of this lab, you will be able to set up a basic ASP.NET Core project, create a web API, and implement GET, POST, PUT, and DELETE endpoints. You will also learn how to use Postman to test your API.

**Step 1: Prepare for the Application**

You’ll create a small web API using ASP.NET Core. This API will manage a simple list of items with basic CRUD (Create, Read, Update, Delete) operations.

**Steps:**

1. Open Visual Studio Code.
2. Make sure you have the .NET SDK installed. If not, install the latest version from the [official .NET website](https://dotnet.microsoft.com/).
3. Open the terminal in Visual Studio Code (Ctrl + ~ for Windows/Linux or Cmd + ~ for Mac).
4. Create a new ASP.NET Core Web API project:
5. Navigate to the project directory.
6. Open the project in Visual Studio Code.

**Step 2: Setting Up the АРI Project**

You’ll now set up the basic project structure and configure the necessary components for your API.

**Steps:**

1. In the Program.cs file, remove any existing code and start with a clean slate.
2. Use the following basic structure to set up your API routes: var builder = WebApplication.CreateBuilder(args);

var app = builder.Build();

// Basic routes

app.MapGet("/", () => "Welcome to the Simple Web API!");

app.Run();

1. Save the file and run your project.
2. Open a web browser and check if the API is running.

**Step З: Implement CRUD Endpoints**

Create the necessary endpoints for managing a list of items (GET, POST, PUT, DELETE).

**Steps:**

1. Create a new folder named Models and add a file called Item.cs.
2. Define a basic model for the items.
3. In the Program.cs file, create an in-memory list to store items.
4. Implement the endpoints.
   1. GET all items:
   2. GET a specific item by ID:
   3. POST a new item:
   4. PUT to update an existing item:
   5. DELETE an item by ID:
5. Save the file and test the endpoints by running the project.

**Step 4: Testing the API with Postman**

Use Postman to test your API endpoints.

**Steps:**

1. Open Postman and create a new request.
2. Set the request type (GET, POST, PUT, DELETE) in the dropdown menu.
3. Enter the API URL in the request field (e.g., http://localhost:5000/items).
4. For POST and PUT requests, go to the "Body" tab, select "raw", and choose "JSON" as the format. Enter your JSON data, for example: {     "name": "New Item" }
5. Click "Send" to make the request.
6. Check the response in the lower section of Postman to ensure the API behaves as expected.

**Item.cs:**

##### **namespace** **ItemsApi.Models**;

##### **public** **class** **Item**

##### {

##### **public** **int** Id { **get**; **set**; }

##### **public** **string** Name { **get**; **set**; } = **string**.Empty;

##### }

**Program.cs:**

**using** **ItemsApi.Models**;

**using** **Microsoft.OpenApi.Models**;

**var** builder = WebApplication.CreateBuilder(args);

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", **new** OpenApiInfo { Title = "Items API", Version = "v1" });

});

**var** app = builder.Build();

**if** (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI(c =>

{

c.SwaggerEndpoint("/swagger/v1/swagger.json", "Items API v1");

});

}

app.UseHttpsRedirection();

**var** items = **new** List<Item>();

**var** nextId = **1**;

app.MapGet("/", () => "Welcome to the Simple Web API (NET 9)!");

**var** itemsApi = app.MapGroup("/items").WithOpenApi();

itemsApi.MapGet("/", () => items);

itemsApi.MapGet("/{id}", (**int** id) =>

{

**var** item = items.FirstOrDefault(i => i.Id == id);

**return** item **is** **not** **null** ? Results.Ok(item) : Results.NotFound();

});

itemsApi.MapPost("/", (Item newItem) =>

{

newItem.Id = nextId++;

items.Add(newItem);

**return** Results.Created($"/items/{newItem.Id}", newItem);

});

itemsApi.MapPut("/{id}", (**int** id, Item updatedItem) =>

{

**var** item = items.FirstOrDefault(i => i.Id == id);

**if** (item **is** **null**) **return** Results.NotFound();

item.Name = updatedItem.Name;

**return** Results.NoContent();

});

itemsApi.MapDelete("/{id}", (**int** id) =>

{

**var** item = items.FirstOrDefault(i => i.Id == id);

**if** (item **is** **null**) **return** Results.NotFound();

items.Remove(item);

**return** Results.NoContent();

});

app.Run();