MINIMAL APIS

GROUP YOUR ENDPOINTS

FOR BETTER

READABILITY

@romain-od

www.code-review.tech

ENDPOINT BEFORE .NET7

No possibility to group similar endpoints, the only way was to add prefixes for every endpoint.

```
app.MapGet("api/items", () => "Get all items");
app.MapGet("api/items/{id}", (int id) => "Get item");
app.MapPost("api/items", () => "Add item");
app.MapPut("api/items/{id}", (int id) => "Update item");
app.MapDelete("api/items/{id}", (int id) => "Delete item");
app.MapGet("api/users", () => "Get all users");
app.MapGet("api/users/{id}", (int id) => "Get user");
app.MapPost("api/users", () => "Add user");
app.MapPut("api/users/{id}", (int id) => "Update user");
app.MapDelete("api/users/{id}", (int id) => "Delete user");
```

@romain-od

www.code-review.tech

.NET7 Introduce MapGroup()

```
var items = app.MapGroup("api/items/");
items.MapGet("", () => "Get all items");
items.MapGet("{id}", (int id) => "Get item");
items.MapPost("", () => "Add item");
items.MapPut("{id}", (int id) => "Update item");
items.MapDelete("{id}", (int id) => "Delete item");

var users = app.MapGroup("api/users/");
users.MapGet("api/users", () => "Get all users");
users.MapGet("{id}", (int id) => "Get user");
users.MapPost("", () => "Add user");
users.MapPut("{id}", (int id) => "Update user");
users.MapDelete("{id}", (int id) => "Delete user");
```

@romain-od

www.code-review.tech

That great, now we can apply constraints and rules on a bunch of endpoints.

But the list can grow fast.

To improve readability let's create extension method of `RouteGroupBuilder` and put our group building out of program.cs

@romain-od

www.code-review.tech

EXTENSION METHODS

TO READABILITY

```
public static class MyGroups
{
    public static RouteGroupBuilder GroupItems(this RouteGroupBuilder group)
    {
        group.MapGet("", () => "Get all items");
        group.MapGet("{id}", (int id) => "Get item");
        group.MapPost("", () => "Add item");
        group.MapPut("{id}", (int id) => "Update item");
        group.MapDelete("{id}", (int id) => "Delete item");
        return group;
    }

    public static RouteGroupBuilder GroupUsers(this RouteGroupBuilder group)
    {
        group.MapGet("", () => "Get all users");
        group.MapGet("{id}", (int id) => "Get user");
        group.MapPost("", () => "Add user");
        group.MapPut("{id}", (int id) => "Update user");
        group.MapDelete("{id}", (int id) => "Delete user");
        return group;
    }
}
```

@romain-od

www.code-review.tech

FINAL PROGRAM.CS

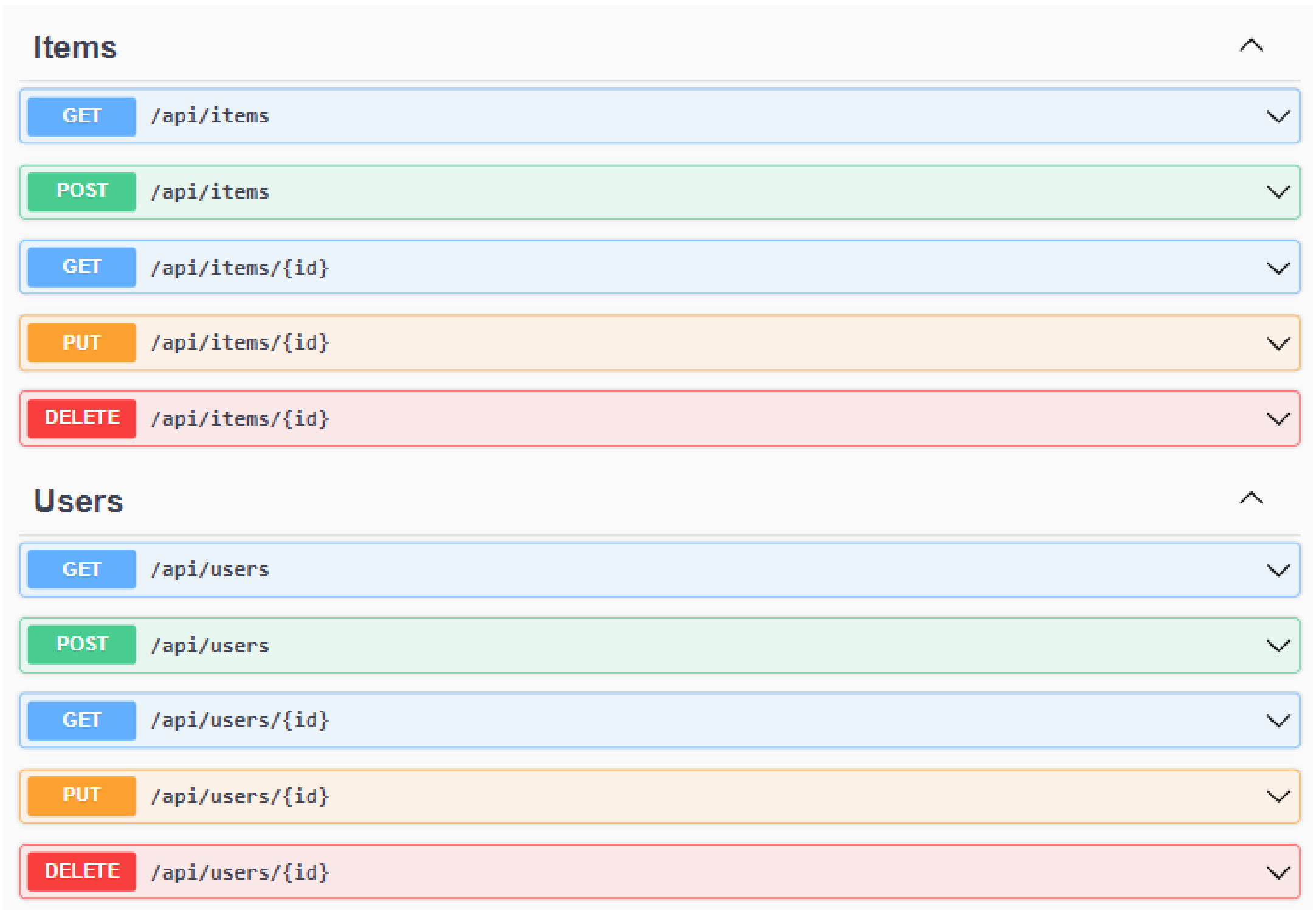
```
var builder = WebApplication.CreateBuilder(args);

builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

var app = builder.Build();

var items = app.MapGroup("api/items/")
               .GroupItems()
               .WithTags("Items");;

var users = app.MapGroup("api/users/")
               .GroupUsers()
               .WithTags("Users");

app.UseSwagger();
app.UseSwaggerUI();
app.Run();
```

@romain-od

www.code-review.tech

SWAGGER

As you can spot above, I use the `WithTags()` extensions methods for clearer documentation



@romain-od

www.code-review.tech

THANKS FOR READING

SHARE YOUR THOUGHTS

FOLLOW ME FOR MORE SHARING

@romain-od

www.code-review.tech