Swagger implementation con netcore:

Tutorial online:

https://docs.microsoft.com/it-it/aspnet/core/tutorials/getting-started-with-nswag?view=aspnetcore-3.1&tabs=visual-studio

Passaggi:

Installare swagger

PowerShell Copia

```
Install-Package NSwag.AspNetCore
```

Aggiungere il servizio in startup:

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddDbContext<TodoContext>(opt =>
        opt.UseInMemoryDatabase("TodoList"));
    services.AddMvc();

    // Register the Swagger services
    services.AddSwaggerDocument();
}
```

C# Copia

```
public void Configure(IApplicationBuilder app)
{
    app.UseStaticFiles();

    // Register the Swagger generator and the Swagger UI middlewares
    app.UseOpenApi();
    app.UseSwaggerUi3();

    app.UseMvc();
}
```

Per creare una personalizzazione della documentazione api aggiungere a startup questo:

```
services.AddSwaggerDocument(config =>
{
    config.PostProcess = document =>
    {
        document.Info.Version = "v1";
        document.Info.Title = "ToDo API";
        document.Info.Description = "A simple ASP.NET Core web API";
        document.Info.TermsOfService = "None";
        document.Info.Contact = new NSwag.OpenApiContact
        {
            Name = "Shayne Boyer",
            Email = string.Empty,
            Url = "https://twitter.com/spboyer"
        };
        document.Info.License = new NSwag.OpenApiLicense
```

```
        {
            Name = "Use under LICX",
            Url = "https://example.com/license"
        };
    };
});
```

per abilitare i commenti aggiungere al file prj la seguente sezione:

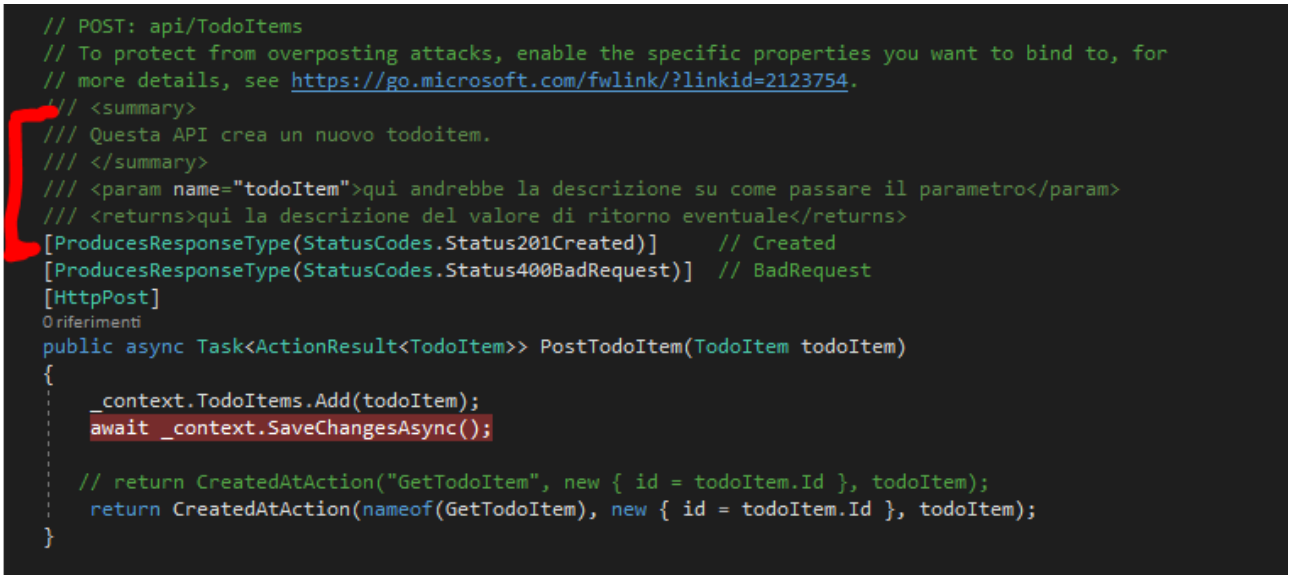
```
<PropertyGroup> <GenerateDocumentationFile>true</GenerateDocumentationFile>
  <NoWarn>$(NoWarn);1591</NoWarn>
</PropertyGroup>
```

Per far gestire dall'ambiente le risposte di ritorno (stati) in base a cio che succede durante l'esecuzione dell'api usare gli attributi ad esempio su un post:

```
[ProducesResponseType(StatusCodes.Status201Created)]     // Created
[ProducesResponseType(StatusCodes.Status400BadRequest)]  // BadRequest
```

Questo aggiunge due stati come argomento di uscita oltre a quello di default 200 aggiunto automaticamente se la classe è decorata con [ApiController]

Per aggiungere commenti personali alle API, ai parametri in ingresso ed ai valori di ritorno commentare direttamente il codice:



```
// POST: api/TodoItems
// To protect from overposting attacks, enable the specific properties you want to bind to, for
// more details, see https://go.microsoft.com/fwlink/?linkid=2123754.
/// <summary>
/// Questa API crea un nuovo todoitem.
/// </summary>
/// <param name="todoItem">qui andrebbe la descrizione su come passare il parametro</param>
/// <returns>qui la descrizione del valore di ritorno eventuale</returns>
[ProducesResponseType(StatusCodes.Status201Created)]     // Created
[ProducesResponseType(StatusCodes.Status400BadRequest)]  // BadRequest
[HttpPost]
0 riferimenti
public async Task<ActionResult<TodoItem>> PostTodoItem(TodoItem todoItem)
{
    _context.TodoItems.Add(todoItem);
    await _context.SaveChangesAsync();

    // return CreatedAtAction("GetTodoItem", new { id = todoItem.Id }, todoItem);
    return CreatedAtAction(nameof(GetTodoItem), new { id = todoItem.Id }, todoItem);
}
```