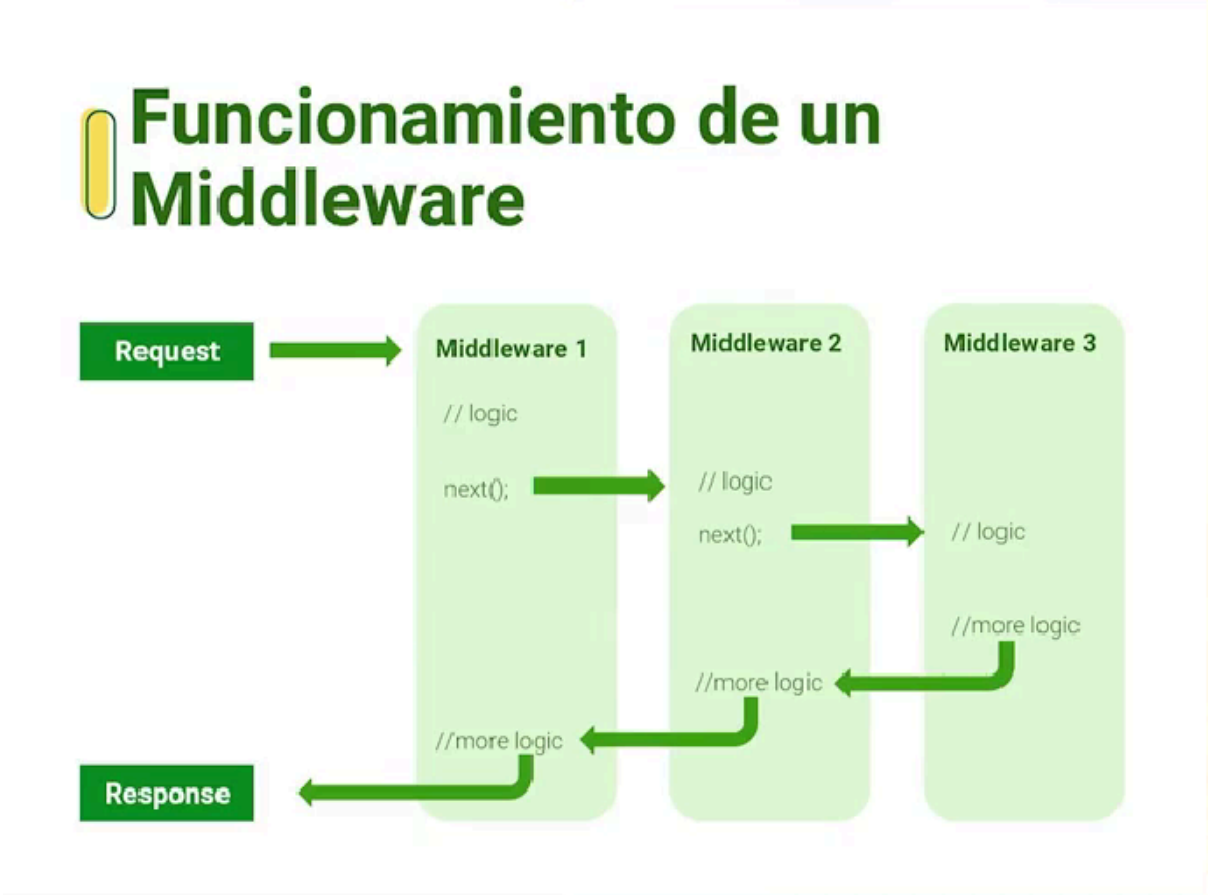
Qué son los middlewares?

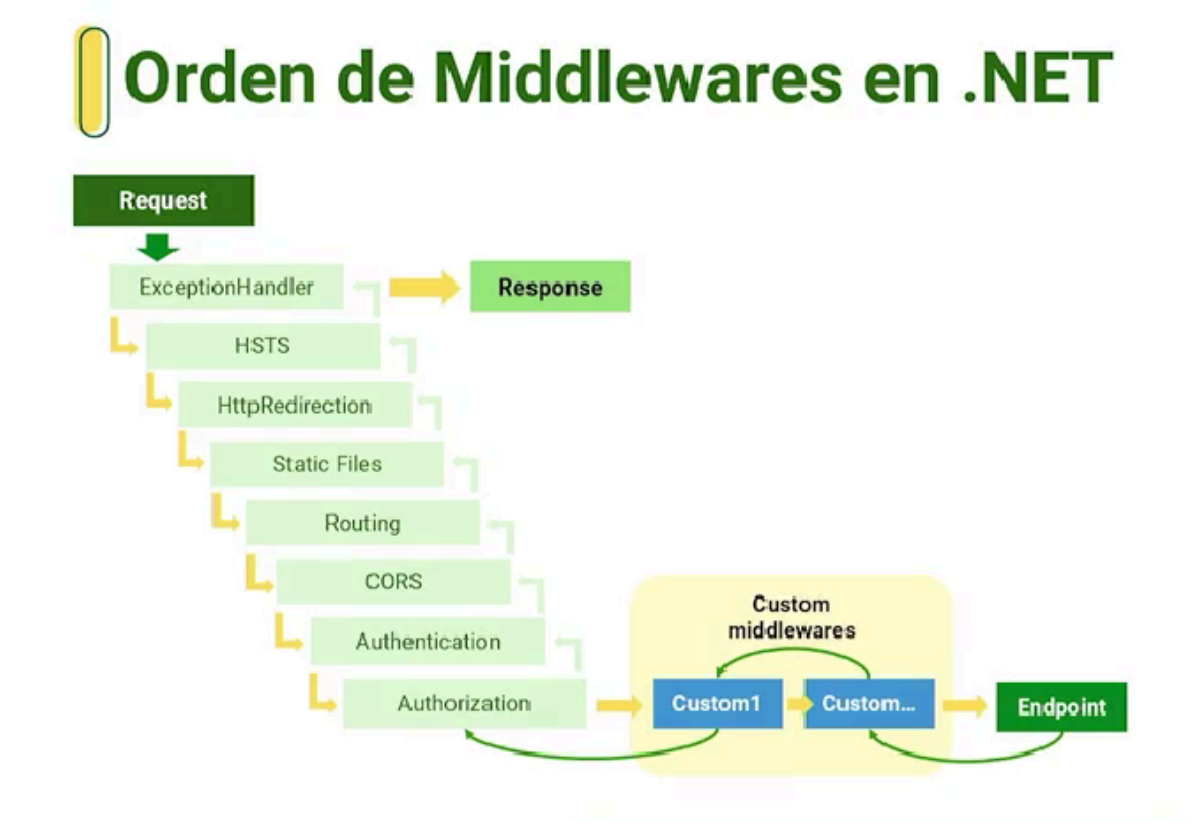
Los middlewares son componentes en una aplicación web que permiten realizar acciones específicas en cada solicitud que se recibe.

Los middlewares se ejecutan en el orden en que se agregaron en la cadena de solicitudes, y pueden elegir si pasar la solicitud al siguiente middleware en la cadena o si detener la cadena y proporcionar una respuesta.

- Es una serie de instrucciones de código que se agregan al ciclo de vida de una petición Http
- Provee una ejecución de peticiones a través de capas
- Facilitan la implementación de interceptores y filtros sobre las peticiones en un API



El orden en el que los middlewares son agregados en .NET es crítico, ya que puede afectar el comportamiento de la aplicación.



Ubicación

```
Program.cs
DemoTallerCF
22
23      var app = builder.Build();
24
25      // Configure the HTTP request pipeline.
26      if (app.Environment.IsDevelopment())
27      {
28          app.UseSwagger();
29          app.UseSwaggerUI();
30      }
31
32      app.UseHttpsRedirection();
33
34      app.UseAuthorization();
35
36      app.MapControllers();
37
38      app.Run();
39
```

Tipos de middleware

Middleware de terminal

El middleware de terminal es responsable de enviar la respuesta de vuelta al cliente. Es el componente de middleware final del proceso. El middleware de terminal se puede utilizar para modificar la respuesta saliente antes de que se envíe de vuelta al cliente.

Algunos ejemplos de middleware de terminal incluyen:

- Middleware de archivos estáticos: este middleware se utiliza para servir archivos estáticos como CSS, JavaScript e imágenes.
- Middleware de servidor de archivos: este middleware se utiliza para servir archivos desde un directorio específico.
- Middleware MVC: este middleware se utiliza para gestionar solicitudes de puntos finales MVC.

Middleware no terminal

El middleware no terminal es cualquier componente de middleware que no sea el componente final del proceso. El middleware no terminal se puede utilizar para modificar las solicitudes entrantes y las respuestas salientes.

Algunos ejemplos de middleware no terminal incluyen:

- Middleware de autenticación: este middleware se utiliza para autenticar a los usuarios.
- Middleware de autorización: este middleware se utiliza para autorizar a los usuarios a acceder a determinados recursos.
- Middleware de compresión de respuesta: este middleware se utiliza para comprimir la respuesta antes de enviarla de vuelta al cliente.
- Middleware de registro de solicitudes: este middleware se utiliza para registrar solicitudes.
- Middleware de enrutamiento: este middleware se utiliza para enrutar solicitudes al punto final apropiado.

Middleware personalizado

Además de los componentes de middleware integrados que están disponibles en .NET Core, también puede crear sus propios componentes de middleware personalizados. La creación de middleware personalizado le permite agregar funcionalidad a su aplicación que sea específica para sus necesidades.

Crear un middleware en .NET

Para crear un middleware en .NET lo único que tenemos que hacer es una clase normal, con la excepción de dos reglas

a. El constructor debe recibir RequestDelegate.
b. Un método llamado Invoke que reciba HttpContext como parámetro.

```
public class EjemploMiddleware
{
    private readonly RequestDelegate _next;

    public EjemploMiddleware(RequestDelegate next)
    {
        _next = next;
    }

    public async Task Invoke(HttpContext context)
    {
      await _next(context);
    }
}
```

Y en la clase podemos hacer lo que queramos, incluso podemos acceder al **contendor de dependenicas** a través de `context.RequestServices.GetService<T>()` o a través de la **inyeccion de dependencias** normal a través del constructor.

Ahora añadimos lógica, en el ejemplo es sencillo, loguear el tiempo que tarda la request, pero lo podemos complicar tanto como queramos.

```
public class EjemploMiddleware
{
    private readonly RequestDelegate _next;  //Continua con el
siguiente middleware
    private readonly ILogger _logger;


    public EjemploMiddleware(RequestDelegate next, ILoggerFactory
loggerFactory)
    {
        _next = next;
        _logger =
loggerFactory.CreateLogger(typeof(EjemploMiddleware));
    }



    public async Task Invoke(HttpContext context)
    {
        //Este id es para simular un ID que viene del FrontEnd/Container
       //Con el cual mantenemos "trace" de toda la acción del usuario
       //Antes de la request

        Guid traceId = Guid.NewGuid();
        _logger.LogDebug($"Request {traceId} iniciada");
        Stopwatch stopWatch = new Stopwatch();
        stopWatch.Start(); //Indica cuánto toma procesar la
solicitud


        await _next(context);

        //Despues de la request
        stopWatch.Stop();

        TimeSpan ts = stopWatch.Elapsed;
        string elapsedTime =
String.Format("{0:00}:{1:00}:{2:00}.{3:00}",
```

```
            ts.Hours, ts.Minutes, ts.Seconds, ts.Milliseconds / 10);
            _logger.LogDebug($"La request {traceId} ha llevado
{elapsedTime} ");
        }
}
```

Para activar nuestro middleware debemos hacerlo en el método `Configure` dentro de `startup.cs` y para ello únicamente incluiremos el siguiente código.

```
app.UseMiddleware<EjemploMiddleware>();
```