Crear un archivo Log

Un logger es una herramienta o mecanismo que permite registrar y rastrear eventos o mensajes generados por una aplicación durante su ejecución. Estos registros, conocidos como logs, son útiles para depurar, monitorear y analizar el comportamiento de la aplicación, especialmente en entornos de desarrollo, prueba y producción.

FileLogger.cs

```
public class FileLogger
{
    private readonly string _filePath;

    1 referencia
    public FileLogger(string filePath)
    {
        _filePath = filePath;
    }

    1 referencia
    public void LogError(string message, Exception ex)
    {
        var logMessage = $"{DateTime.Now:yyyy-MM-dd HH:mm:ss} - ERROR: {message}\n" +
            $"Exception: {ex.Message}\nStack Trace: {ex.StackTrace}\n";
        WriteToFile(logMessage);
    }

    0 referencias
    public void LogInfo(string message)
    {
        var logMessage = $"{DateTime.Now:yyyy-MM-dd HH:mm:ss} - INFO: {message}\n";
        WriteToFile(logMessage);
    }

    2 referencias
    private void WriteToFile(string logMessage)
    {
        lock (this) // Evitar problemas con accesos concurrentes
        {
            File.AppendAllText(_filePath, logMessage);
        }
    }
```

Program.cs

```
var logFilePath = Path.Combine(
    AppDomain.CurrentDomain.BaseDirectory,      //Ruta base del proyecto una vez compilado
    "Logs",             //Nombre de la carpeta
    "api-errors.log");  //Nombre del archivo

Directory.CreateDirectory(Path.GetDirectoryName(logFilePath)); // Crear el directorio si no existe

builder.Services.AddSingleton(new FileLogger(logFilePath));
```

ErrorLoggingMiddleware.cs

```
public class ErrorLoggingMiddleware
{
    private readonly RequestDelegate _next;
    private readonly FileLogger _logger;

    0 referencias
    public ErrorLoggingMiddleware(RequestDelegate next,
                                  FileLogger logger)
    {
        _next = next;
        _logger = logger;
    }

    0 referencias
    public async Task Invoke(HttpContext context)
    {
        try
        {
            await _next(context); // Procesar la solicitud
        }
        catch (Exception ex)
        {
            // Registrar el error en el archivo
            _logger.LogError($"Error procesando la solicitud:" +
                $" {context.Request.Method} {context.Request.Path}", ex);

            // Enviar una respuesta de error al cliente
            context.Response.StatusCode = 500;
            await context.Response.WriteAsync("Ocurrió un error interno del servidor.");
        }
    }
}
```

Program.cs

```
app.UseAuthorization();

app.UseMiddleware<ErrorLoggingMiddleware>();

app.MapControllers();

app.Run();
```