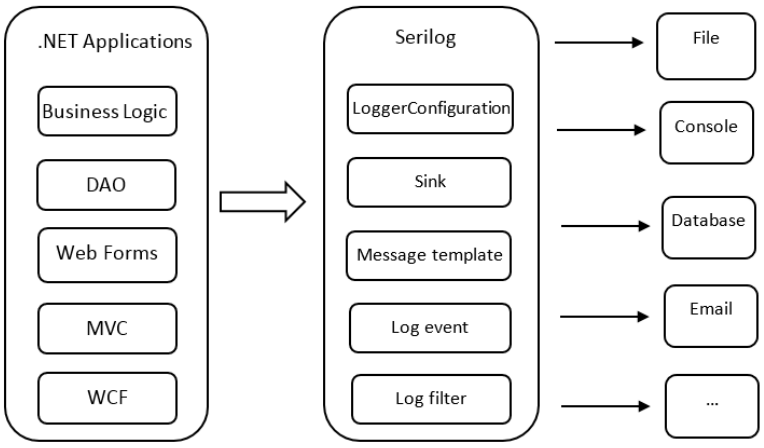
Serilog

¿Qué es?

Un Log es un registro de información sobre la ejecución de un programa con fines de depurar y rastrear. El registro generalmente implica escribir mensajes de texto para registrar archivos o enviar datos a aplicaciones de monitoreo. La información registrada proporciona una buena comprensión sobre datos procesados, advertencias, errores y más. En la práctica, el registro permite a los desarrolladores y probadores identificar problemas de software, monitorear sistemas en vivo y verificar salidas.

Serilog nos proporciona Los mensajes de registro se conservan como datos estructurados (JSON) que se pueden escribir en forma de documento para una variedad de proveedores de resultados.

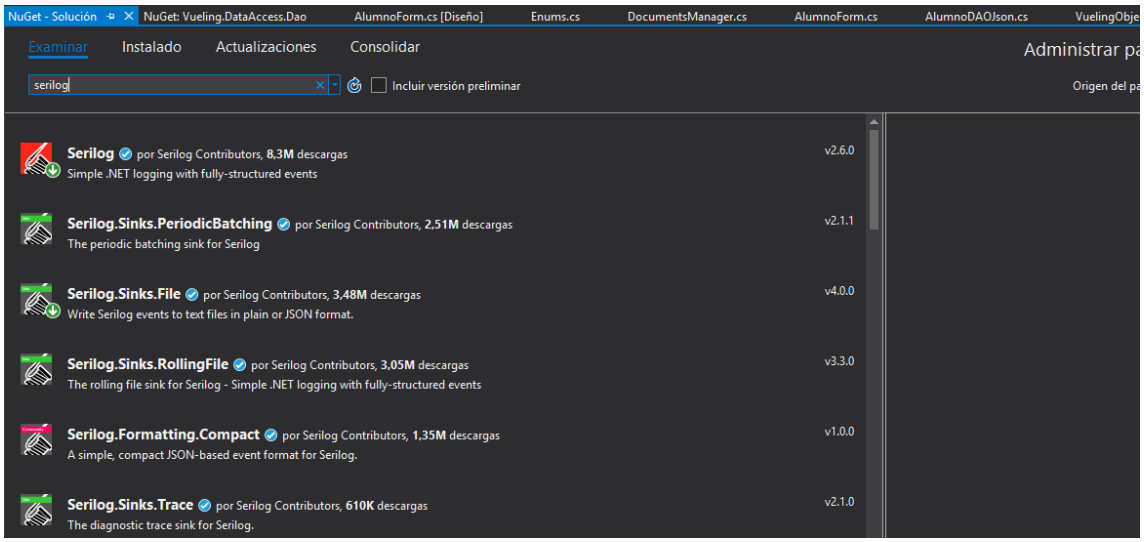
Estructura



Instalación:

1. **Servidor de paquetes Nuget.**

Botón derecho en nuestro proyecto->Administrar paquete de nugets y buscamos serilog.



2.Consola de proyecto.

Menu>herramientas>Administrador de paquetes Nuget> Consola de Administracion de paquetes.

>Install-package Serilog.

Configuración de Logger.

Para configurar nuestro log tenemos que crear un objeto LoggerConfiguration.

A continuación, podemos configuraros los Sinks. Esto son las salidas del log. Para ello deberemos bajarnos también sus respectivos paquetes. Un ejemplo sería el siguiente:

```
var logger = new LoggerConfiguration()
    .WriteTo.ColoredConsole()
    .WriteTo.RollingFile(@"C:\Log-{Date}.txt")
    .CreateLogger();
```

En este ejemplo vemos que crea nuestro objeto loggerconfiguration y define que las salidas (WriteTo) serán en consola y en un texto con el nombre log y la fecha actual.

-Tipos de mensaje:

En serilog por defecto encontramos varios tipos de mensajes:

```
log.Debug("Debugging message");
log.Information("Information message");
log.Warning("Warning message");
log.Error("Error message");
log.Fatal("Fatal message");
```

Verbose	Rastreo de información y minucias de depuración; generalmente solo se enciende en situaciones inusuales
Debug	Volcado de control interno y volcados de estado de diagnóstico para facilitar la localización precisa de problemas reconocidos
Information	Eventos de interés o que tengan relevancia para observadores externos; el nivel de registro mínimo habilitado predeterminado
Warning	Indicadores de posibles problemas o degradación del servicio / funcionalidad
Error	Indicando una falla dentro de la aplicación o sistema conectado
Fatal	Errores críticos que causan falla completa de la aplicación

:

```
public static int Main(string[] args)
{
    //Build Config
    var currentEnv = Environment.GetEnvironmentVariable("ASPNETCORE_ENVIRONMENT");
    var configuration = new ConfigurationBuilder()
        .AddJsonFile("appsettings.json")
        .AddJsonFile($"appsettings.{currentEnv}.json", optional: true)
        .AddEnvironmentVariables()
        .Build();

    //Configure Logger
    Log.Logger = new LoggerConfiguration()
        .ReadFrom.Configuration(configuration)
        .CreateLogger();

    try
    {
        Log.Information("Starting web host");
        BuildWebHost(args).Run();
        return 0;
    }
    catch (Exception ex)
    {
        Log.Fatal(ex, "Web Host terminated unexpectedly");
        return 1;
    }
    finally
    {
        Log.CloseAndFlush();
    }
}
```