OPC FOUNDATION CLIENT-SERVER SOLUTION

This document contains the I+D technical details of the MVC solution that integrates Hangfire and OPC Foundation

Contents

Doc. Version	Date	Author	Description
1.0	2025-10-17	https://github.com/kraftcoding	First version of the release

Requirements

The next table shows the requirements for the solution described.

Requirement	Description
RQ-001 | MVC application with Hangfire	MVC application integrating Hangfire (including a console and Windows Server project for Hangfire.
RQ-002 | OPC Foundation client-server and DB model layer	Custom client-server OPC libraries and jobs, as well as the model and database access.
RQ-003 | OPC Foundation core	Projects with the source code containing the SDK & Stack of OPC Foundation.

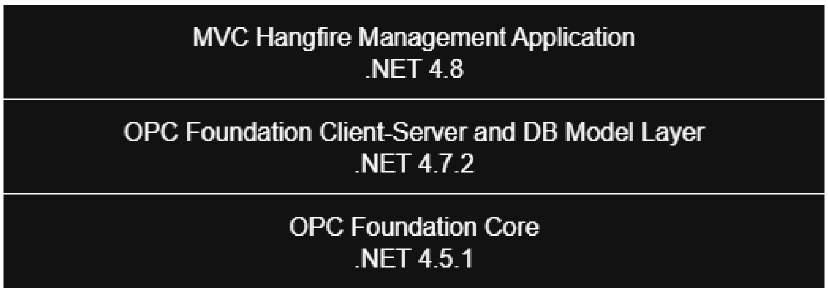
Overview

This document contains the technical details of the MVC application that integrates Hangfire 1.8.21 and OPC Foundation, running on IIS/SQL Server.

Solutions:

- **Hangfire-OPC-Labs_.NET-4.8:** Contains the MVC application with Hangfire (including a console and Windows Server project for Hangfire).
- **OPC-Foundation-Labs-Server-Client-.Net-4.8:** Containing the projects that make up the custom client-server OPC libraries and jobs, as well as the model and database access.
- **OPC-Foundation-Labs-.Net-4.8:** The OPC Foundation projects and core code (SDK & Stack).

The next image shows solution hierarchy and management relation.


MVC Hangfire Management Application
.NET 4.8
OPC Foundation Client-Server and DB Model Layer
.NET 4.7.2
OPC Foundation Core
.NET 4.5.1

The application provides the next capabilities:

- Start/stop a background job that runs an OPC server, where an XML configuration file tells it which nodes it should manage.
- Start/stop a background job that runs an OPC client (connecting to the server), where an XML configuration file tells it which nodes it should subscribe to in a monitored manner. As soon as it is notified of a change from the server, it dumps the new values for those nodes into the database on the server (with an "idProcess" flag).
- Add a recurring job that executes an OPC client (connecting to the server) where an XML configuration file indicates the server nodes to which it should write values.
- Add a recurring job that executes an OPC client (connecting to the server) where an XML configuration file indicates the server nodes from which it should read values.
- View logs in real time using a text buffer.

  NOTE: The XML configuration file used for jobs is the OPC one.

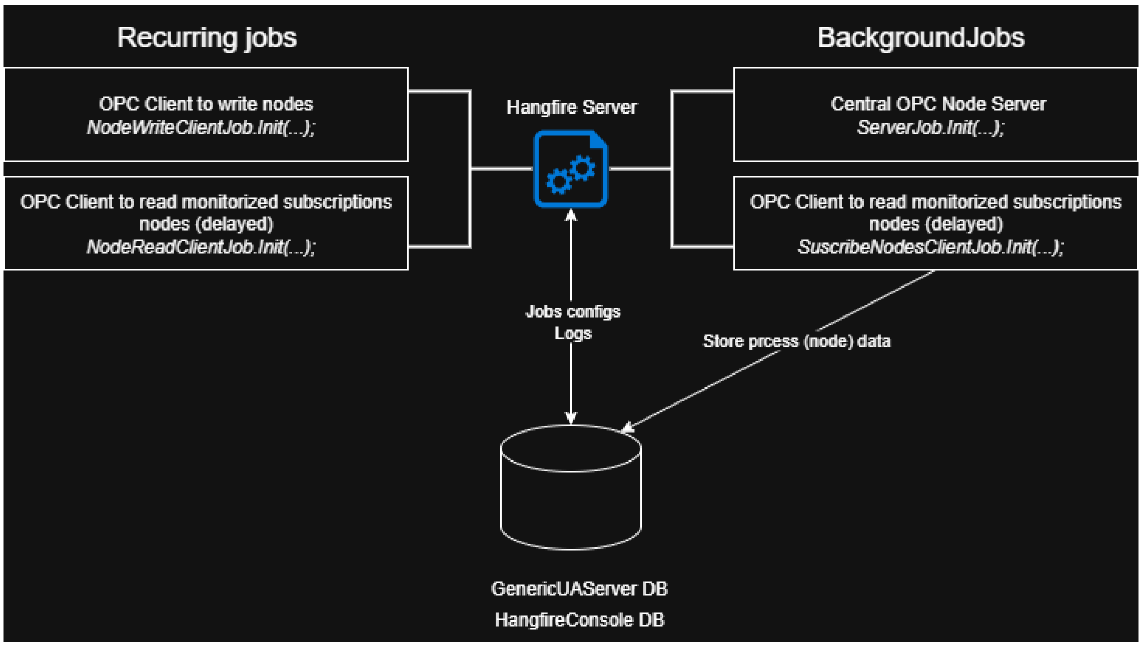
This latest version of the Hangfire server includes:

- Cancellation Tokens to stop jobs in the background.
- All jobs are implemented using the chain of responsibility pattern and can be intercepted using action filters

Basically, the solution creates four jobs:

- First, two BackgroundJob jobs: one for the OPC server (ServerJob) and the other for the client for monitored subscriptions (SubscribeNodesClientJob).
- Then, it creates two more RecurringJob jobs: one to simulate the infrastructure with random values (NodeWriteClientJob) and the other to read the values directly without subscriptions (NodeReadClientJob).

Here the diagram with the solution architecture overvie.


Recurring jobs
BackgroundJobs
OPC Client to write nodes
NodeWriteClientJob.Init(...);
OPC Client to read monitorized subscriptions nodes (delayed)
NodeReadClientJob.Init(...);
Hangfire Server
Central OPC Node Server
ServerJob.Init(...);
OPC Client to read monitorized subscriptions nodes (delayed)
SuscribeNodesClientJob.Init(...);
Jobs configs
Logs
Store prcess (node) data
GenericUAServer DB
HangfireConsole DB

OPC UA (Open Platform Communications Unified Architecture) is a cross-platform, machine-to-machine communication protocol for industrial automation that enables secure and reliable data exchange between different systems. It acts as a "universal translator" for devices and software, using a client-server architecture and built-in security to transfer data from factory floor machinery to higher-level systems like SCADA and ERP.

Key features

Platform-independent: OPC UA works across various operating systems, including Windows, Linux, and Android.

- Secure by design: It has built-in security features like data encryption, authentication, and access control, which are a major improvement over its predecessor.
- Client-server model: It uses a client-server architecture where clients can request data directly or subscribe to receive updates when specific conditions are met.
- Scalable and flexible: The protocol is scalable, meaning it can handle anything from simple status data to complex, plant-wide information, and is flexible enough to connect different systems.

- Open standard: Developed by the OPC Foundation, it is based on the international standard IEC62541, making it an open and vendor-neutral technology.

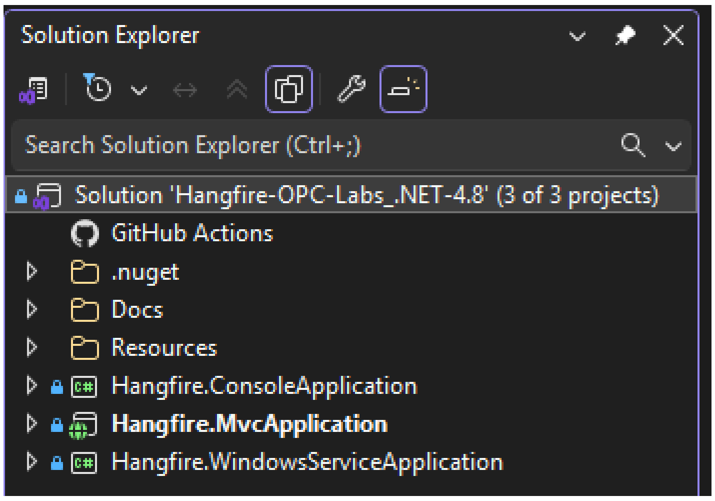
How it's used

- Interoperability: It connects diverse systems, such as PLCs, SCADA, MES, and ERP systems, allowing them to communicate and share data seamlessly.
- Data exchange: It facilitates communication from the control level of a factory all the way up to enterprise-level management systems.
- Real-time monitoring: It provides real-time and historical data access, which is critical for process control, monitoring, and troubleshooting in various industries like manufacturing and renewable energy.
- Device-to-device communication: It enables machine-to-machine communication, making it a core technology for the Industrial Internet of Things (IIoT).

Clean Architecture

Clean Architecture is an architecture pattern that is based on four layers: presentation, application, domain, and infrastructure. The presentation layer is responsible for handling user interactions, the application layer is responsible for handling the business logic, the domain layer is responsible for defining the business entities and their relationships, and the infrastructure layer is responsible for handling external concerns such as databases and web services.

Here the solution structure (Visual Studio 2022):



Solution 'Hangfire-OPC-Labs_.NET-4.8' (3 of 3 projects)
GitHub Actions
.nuget
Docs
Resources
Hangfire.ConsoleApplication
Hangfire.MvcApplication
Hangfire.WindowsServiceApplication

The key principles of Clean Architecture include the independence of the layers, the data flow from the outer layers to the inner layers, and the ability to swap out dependencies without affecting the rest of the system. Clean Architecture emphasizes the separation of concerns to make the codebase easier to maintain, test, and extend.

The project structure of Clean Architecture follows a structured layout that enables code reusability and testability. In the project structure, you would typically have a project for the presentation layer, a project for the application layer, a project for the domain layer, a project for the infrastructure layer, and a project for the tests. Each layer would have its own set of interfaces and implementations, with the application layer acting as the bridge between the other layers.

Scope

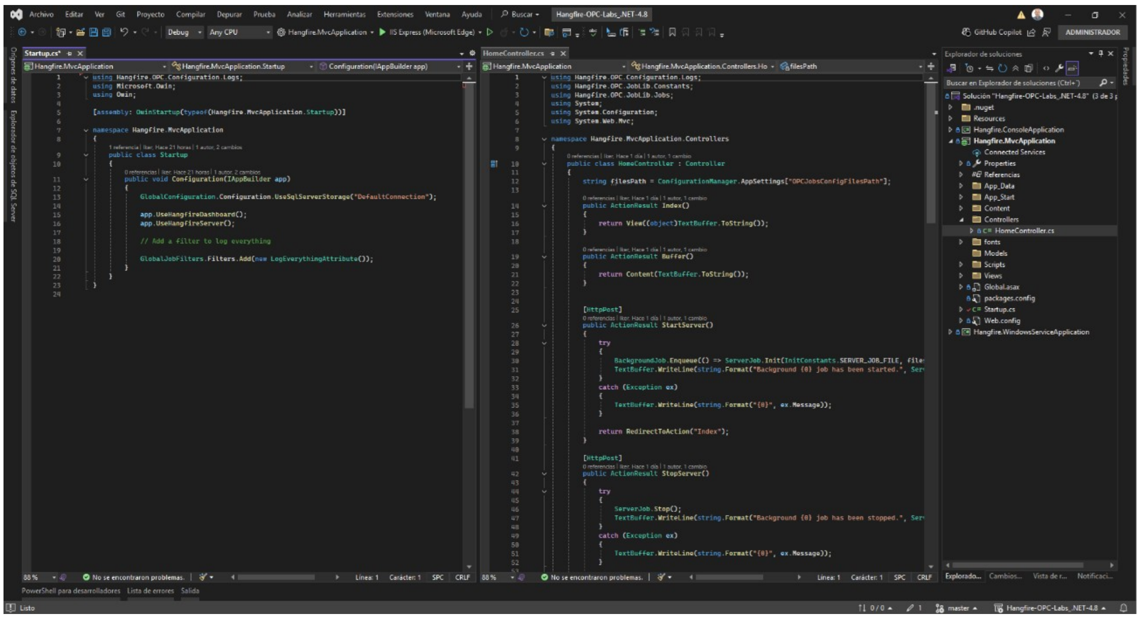
Hangfire is an open-source library for .NET applications that facilitates the execution of background jobs. It allows you to schedule jobs to run at a specific time, recurringly, or immediately, all without the need for external services, and with an administration panel to monitor their status.

OPC technologies are created to allow information to be easily and securely exchanged between diverse platforms from multiple vendors and to allow seamless integration of those platforms without costly, time-consuming software development. This frees engineering resources to do the more important work of running your business.

Using the MVC application, it is possible to provide a web-accessible UI for managing OPC jobs and viewing logs.

RQ-01 | MVC Hangfire Management Application

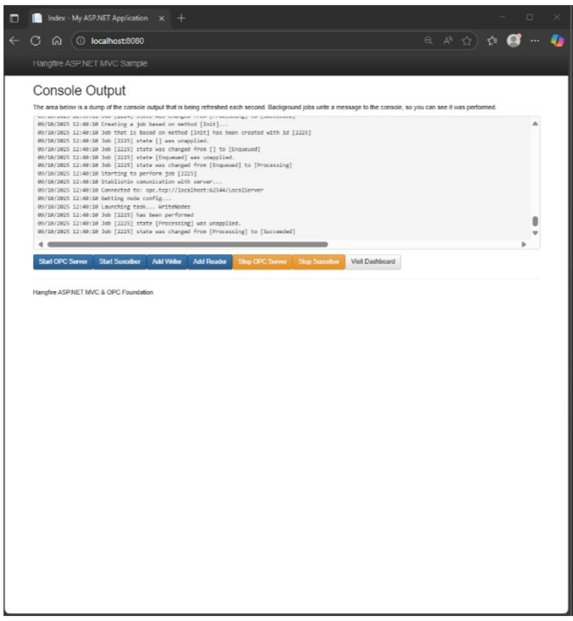
This solution contains a standard .NET 4.8 MVC project by which the Hangfire server is managed. It includes two projects, a console application for tests and a windows service project.


Startup.cs*
HomeController.cs
Hangfire-OPC-Labs_.NET-4.8
Debug
Any CPU
IIS Express (Microsoft Edge)
Explorador de soluciones
No se encontraron problemas.

```
using Hangfire.OPC.Configuration.Logs;
using Microsoft.Owin;
using Owin;

[assembly: OwinStartup(typeof(Hangfire.MvcApplication.Startup))]

namespace Hangfire.MvcApplication
{
    public class Startup
    {
        public void Configuration(IAppBuilder app)
        {
            GlobalConfiguration.Configuration.UseSqlServerStorage("DefaultConnection");

            app.UseHangfireDashboard();
            app.UseHangfireServer();

            // Add a filter to log everything

            GlobalJobFilters.Filters.Add(new LogEverythingAttribute());
        }
    }
}
```

```
using Hangfire.OPC.Configuration.Logs;
using Hangfire.OPC.JobLib.Constants;
using Hangfire.OPC.JobLib.Jobs;
using System;
using System.Configuration;
using System.Web.Mvc;

namespace Hangfire.MvcApplication.Controllers
{
    public class HomeController : Controller
    {
        string filesPath = ConfigurationManager.AppSettings["OPCJobsConfigFilesPath"];

        public ActionResult Index()
        {
            return View((object)TextBuffer.ToString());
        }

        public ActionResult Buffer()
        {
            return Content(TextBuffer.ToString());
        }


        [HttpPost]
        public ActionResult StartServer()
        {
            try
            {
                BackgroundJob.Enqueue(() => ServerJob.Init(InitConstants.SERVER_JOB_FILE, file
                TextBuffer.WriteLine(string.Format("Background {0} job has been started.", Ser
            }
            catch (Exception ex)
            {
                TextBuffer.WriteLine(string.Format("{0}", ex.Message));
            }

            return RedirectToAction("Index");
        }

        [HttpPost]
        public ActionResult StopServer()
        {
            try
            {
                ServerJob.Stop();
                TextBuffer.WriteLine(string.Format("Background {0} job has been stopped.", Ser
            }
            catch (Exception ex)
            {
                TextBuffer.WriteLine(string.Format("{0}", ex.Message));
            }
```

Here the screens of the application:

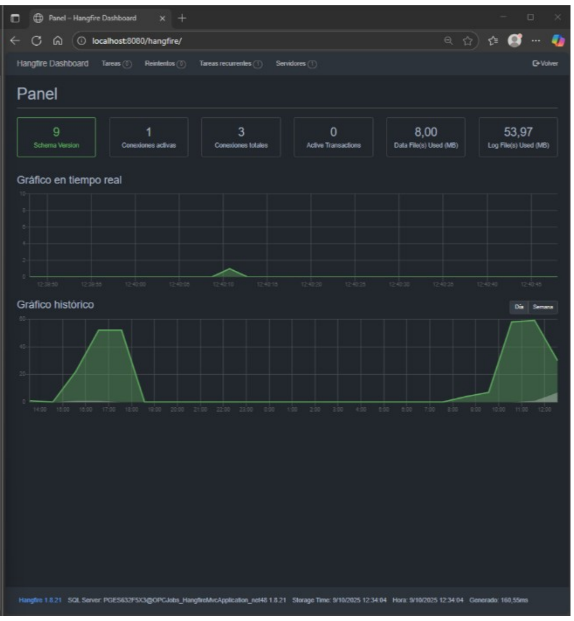


Console Output

The area below is a dump of the console output that is being refreshed each second. Background jobs write a message to the console, so you can see it was performed.

```
09/10/2025 12:40:10 Creating a job based on method [Init]...
09/10/2025 12:40:10 Job that is based on method [Init] has been created with id [2225]
09/10/2025 12:40:10 Job [2225] state [] was unapplied.
09/10/2025 12:40:10 Job [2225] state was changed from [] to [Enqueued]
09/10/2025 12:40:10 Job [2225] state [Enqueued] was unapplied.
09/10/2025 12:40:10 Job [2225] state was changed from [Enqueued] to [Processing]
09/10/2025 12:40:10 Starting to perform job [2225]
09/10/2025 12:40:10 Stablishin comunication with server...
09/10/2025 12:40:10 Connected to: opc.tcp://localhost:62544/LocalServer
09/10/2025 12:40:10 Getting node config...
09/10/2025 12:40:10 Launching task... WriteNodes
09/10/2025 12:40:10 Job [2225] has been performed
09/10/2025 12:40:10 Job [2225] state [Processing] was unapplied.
09/10/2025 12:40:10 Job [2225] state was changed from [Processing] to [Succeeded]
```

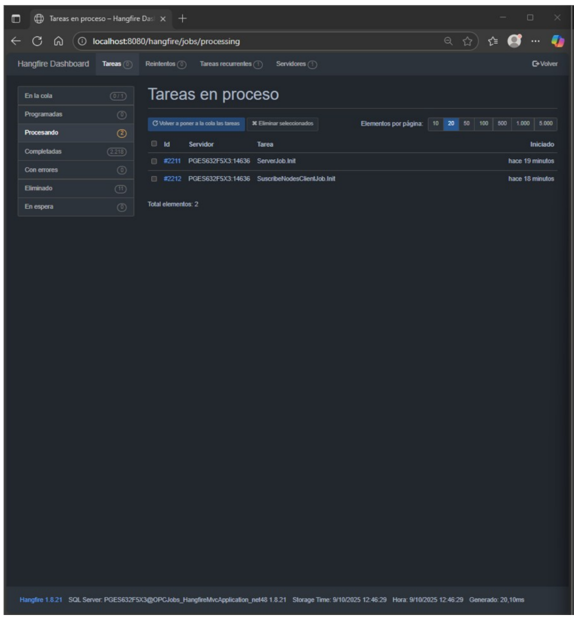
Hangfire ASP.NET MVC & OPC Foundation



Panel

9	1	3	0	8,00	53,97
Schema Version	Conexiones activas	Conexiones totales	Active Transactions	Data File(s) Used (MB)	Log File(s) Used (MB)

Hangfire 1.8.21 SQL Server: PGES632F5X3@OPCJobs_HangfireMvcApplication_net48 1.8.21 Storage Time: 9/10/2025 12:34:04 Hora: 9/10/2025 12:34:04 Generado: 160,55ms

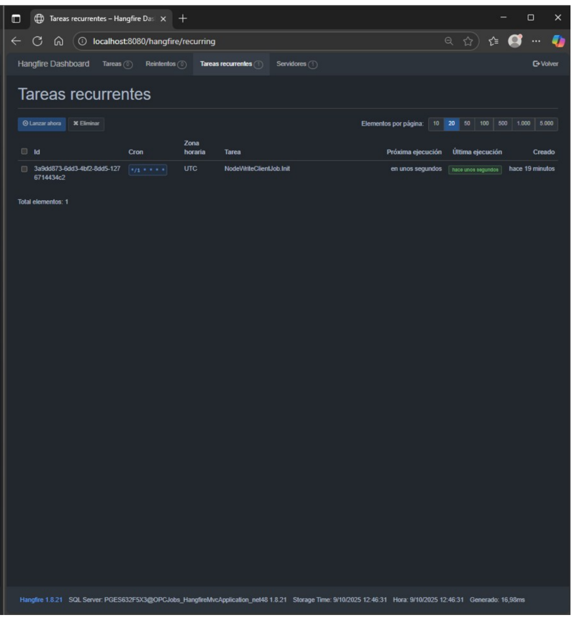


Tareas en proceso

Id	Servidor	Tarea	Iniciado
#2211	PGES632F5X3:14636	ServerJob.Init	hace 19 minutos
#2212	PGES632F5X3:14636	SuscribeNodesClientJob.Init	hace 18 minutos

Total elementos: 2

Hangfire 1.8.21 SQL Server: PGES632F5X3@OPCJobs_HangfireMvcApplication_net48 1.8.21 Storage Time: 9/10/2025 12:46:29 Hora: 9/10/2025 12:46:29 Generado: 20,10ms


Tareas recurrentes – Hangfire Das
localhost:8080/hangfire/recurring
Hangfire Dashboard
Tareas 0
Reintentos 0
Tareas recurrentes 1
Servidores 1
Volver
Lanzar ahora
Eliminar
Elementos por página: 10 20 50 100 500 1.000 5.000

Tareas recurrentes

Id	Cron	Zona horaria	Tarea	Próxima ejecución	Última ejecución	Creado
3a9dd873-6dd3-4bf2-8dd5-127 6714434c2	*/1 * * * *	UTC	NodeWriteClientJob.Init	en unos segundos	hace unos segundos	hace 19 minutos

Total elementos: 1

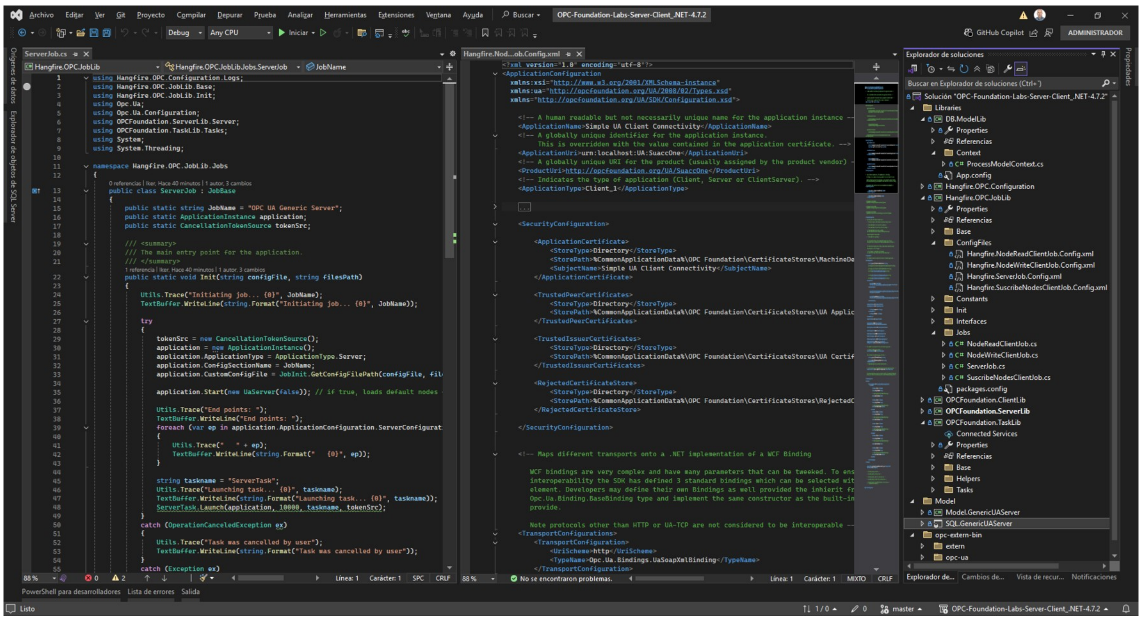
Hangfire 1.8.21 SQL Server: PGES632F5X3@OPCJobs_HangfireMvcApplication_net48 1.8.21 Storage Time: 9/10/2025 12:46:31 Hora: 9/10/2025 12:46:31 Generado: 16,98ms

RQ-02 | OPC Foundation Client-Server and DB Model Layer

This solution contains a standard .NET 4.7.2 projects regarding the client-server layer to interact with OPC Foundation together with the database and model projects to store data.

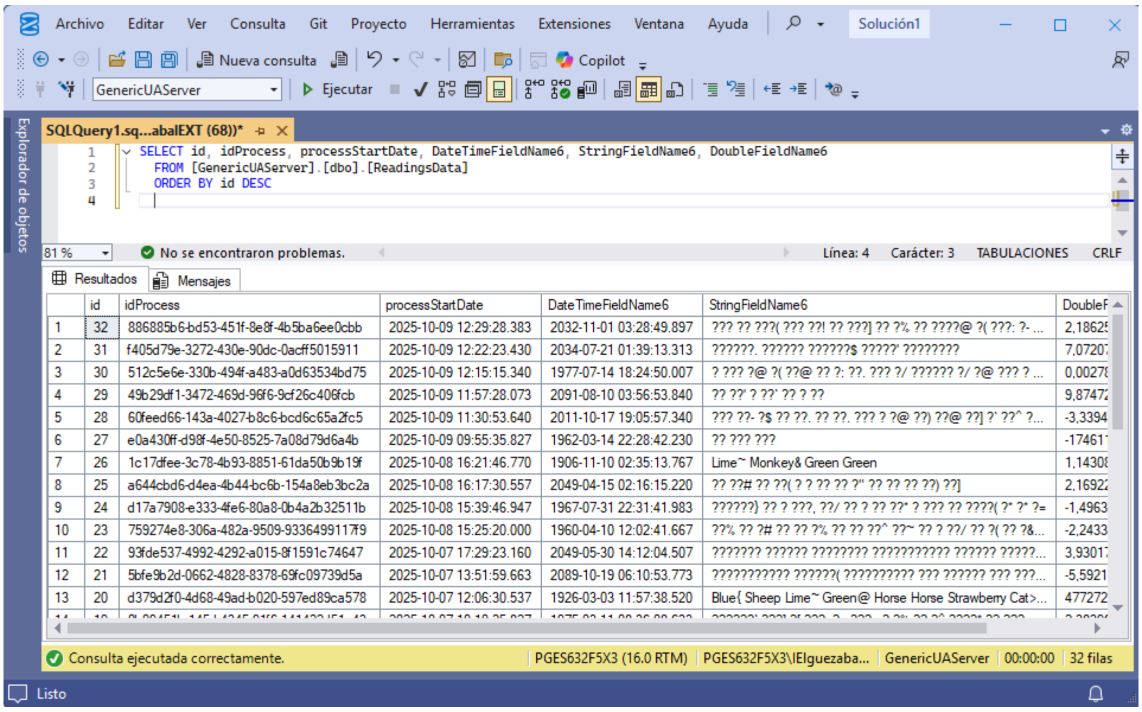
The solution is divided in the next modules or assemblies:

- Client-server layer
  - DB.ModelLib → Database interaction context (EntityFramework)
  - Hangfire.OPC.Configuration → Hangfire default Job configuration
  - Hangfire.OPC.JobLib → Job definitions managed by the Hangfire process
  - OPCFoundation.ClientLib → Opc client procedures and clases to interact with the SDK / Stack
  - OPCFoundation.ServerLib → Opc server procedures and clases to interact with the SDK / Stack
  - OPCFoundation.TaskLib → Tasks managed by the jobs
- Database and model
  - Model.GenericUAServer → GenericUAServer DB model to store the Job process and related data
    Note: Hangfire tables are created automatically
  - SQL.GenericUAServer→ Database to store node data



```csharp
using Hangfire.OPC.Configuration.Logs;
using Hangfire.OPC.JobLib.Base;
using Hangfire.OPC.JobLib.Init;
using Opc.Ua;
using Opc.Ua.Configuration;
using OPCFoundation.ServerLib.Server;
using OPCFoundation.TaskLib.Tasks;
using System;
using System.Threading;

namespace Hangfire.OPC.JobLib.Jobs
{
    0 referencias | Iker, Hace 40 minutos | 1 autor, 3 cambios
    public class ServerJob : JobBase
    {
        public static string JobName = "OPC UA Generic Server";
        public static ApplicationInstance application;
        public static CancellationTokenSource tokenSrc;

        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        1 referencia | Iker, Hace 40 minutos | 1 autor, 3 cambios
        public static void Init(string configFile, string filesPath)
        {
            Utils.Trace("Initiating job... {0}", JobName);
            TextBuffer.WriteLine(string.Format("Initiating job... {0}", JobName));

            try
            {
                tokenSrc = new CancellationTokenSource();
                application = new ApplicationInstance();
                application.ApplicationType = ApplicationType.Server;
                application.ConfigSectionName = JobName;
                application.CustomConfigFile = JobInit.GetConfigFilePath(configFile, fil

                application.Start(new UaServer(false)); // if true, loads default nodes

                Utils.Trace("End points: ");
                TextBuffer.WriteLine("End points: ");
                foreach (var ep in application.ApplicationConfiguration.ServerConfigurat
                {
                    Utils.Trace("    " + ep);
                    TextBuffer.WriteLine(string.Format("    {0}", ep));
                }

                string taskname = "ServerTask";
                Utils.Trace("Launching task... {0}", taskname);
                TextBuffer.WriteLine(string.Format("Launching task... {0}", taskname));
                ServerTask.Launch(application, 10000, taskname, tokenSrc);
            }
            catch (OperationCanceledException ex)
            {
                Utils.Trace("Task was cancelled by user");
                TextBuffer.WriteLine(string.Format("Task was cancelled by user"));
            }
            catch (Exception ex)
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<ApplicationConfiguration
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ua="http://opcfoundation.org/UA/2008/02/Types.xsd"
  xmlns="http://opcfoundation.org/UA/SDK/Configuration.xsd">

    <!-- A human readable but not necessarily unique name for the application instance -
    <ApplicationName>Simple UA Client Connectivity</ApplicationName>
    <!-- A globally unique identifier for the application instance.
         This is overridden with the value contained in the application certificate. -->
    <ApplicationUri>urn:localhost:UA:SuaccOne</ApplicationUri>
    <!-- A globally unique URI for the product (usually assigned by the product vendor) -
    <ProductUri>http://opcfoundation.org/UA/SuaccOne</ProductUri>
    <!-- Indicates the type of application (Client, Server or ClientServer). -->
    <ApplicationType>Client_1</ApplicationType>

    ...

    <SecurityConfiguration>

        <ApplicationCertificate>
            <StoreType>Directory</StoreType>
            <StorePath>%CommonApplicationData%\OPC Foundation\CertificateStores\MachineDe
            <SubjectName>Simple UA Client Connectivity</SubjectName>
        </ApplicationCertificate>

        <TrustedPeerCertificates>
            <StoreType>Directory</StoreType>
            <StorePath>%CommonApplicationData%\OPC Foundation\CertificateStores\UA Applic
        </TrustedPeerCertificates>

        <TrustedIssuerCertificates>
            <StoreType>Directory</StoreType>
            <StorePath>%CommonApplicationData%\OPC Foundation\CertificateStores\UA Certif
        </TrustedIssuerCertificates>

        <RejectedCertificateStore>
            <StoreType>Directory</StoreType>
            <StorePath>%CommonApplicationData%\OPC Foundation\CertificateStores\RejectedC
        </RejectedCertificateStore>

    </SecurityConfiguration>


    <!-- Maps different transports onto a .NET implementation of a WCF Binding

       WCF bindings are very complex and have many parameters that can be tweeked. To ens
       interoperability the SDK has defined 3 standard bindings which can be selected wit
       element. Developers may define their own Bindings as well provided the inhierit fr
       Opc.Ua.Binding.BaseBinding type and implement the same constructor as the built-in
       provide.

       Note protocols other than HTTP or UA-TCP are not considered to be interoperable --
    <TransportConfigurations>
        <TransportConfiguration>
            <UriScheme>http</UriScheme>
            <TypeName>Opc.Ua.Bindings.UaSoapXmlBinding</TypeName>
        </TransportConfiguration>
```



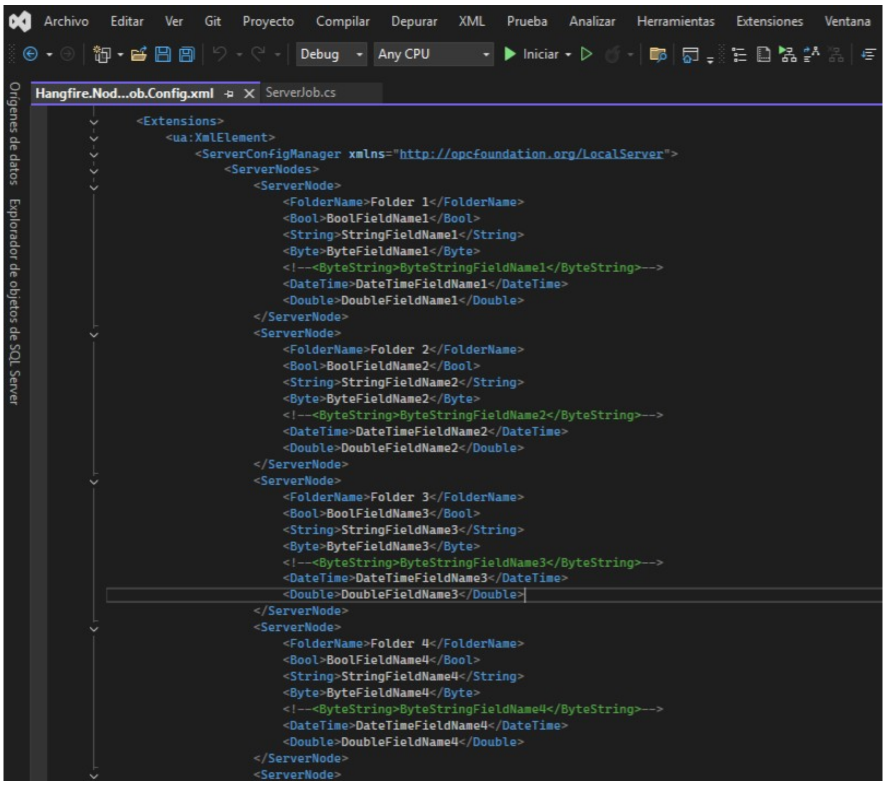
```sql
SELECT id, idProcess, processStartDate, DateTimeFieldName6, StringFieldName6, DoubleFieldName6
  FROM [GenericUAServer].[dbo].[ReadingsData]
  ORDER BY id DESC
```

No se encontraron problemas.

	id	idProcess	processStartDate	DateTimeFieldName6	StringFieldName6	DoubleF
1	32	886885b6-bd53-451f-8e8f-4b5ba6ee0cbb	2025-10-09 12:29:28.383	2032-11-01 03:28:49.897	??? ?? ???(??? ??! ?? ???] ?? ?% ?? ????@ ?(???: ?- ...	2,18625
2	31	f405d79e-3272-430e-90dc-0acff5015911	2025-10-09 12:22:23.430	2034-07-21 01:39:13.313	??????. ?????? ??????$?????' ????????	7,07207
3	30	512c5e6e-330b-494f-a483-a0d63534bd75	2025-10-09 12:15:15.340	1977-07-14 18:24:50.007	? ??? ?@ ?(??@ ?? ?: ??. ??? ?/ ?????? ?/ ?@ ??? ? ...	0,00278
4	29	49b29df1-3472-469d-96f6-9cf26c406fcb	2025-10-09 11:57:28.073	2091-08-10 03:56:53.840	?? ??' ? ??` ?? ? ??	9,87472
5	28	60feed66-143a-4027-b8c6-bcd6c65a2fc5	2025-10-09 11:30:53.640	2011-10-17 19:05:57.340	??? ??- ?$?? ??. ?? ??. ??? ? ?@ ??) ??@ ??] ?` ??^ ?...	-3,3394
6	27	e0a430ff-d98f-4e50-8525-7a08d79d6a4b	2025-10-09 09:55:35.827	1962-03-14 22:28:42.230	?? ??? ???	-174611
7	26	1c17dfee-3c78-4b93-8851-61da50b9b19f	2025-10-08 16:21:46.770	1906-11-10 02:35:13.767	Lime~ Monkey& Green Green	1,14308
8	25	a644cbd6-d4ea-4b44-bc6b-154a8eb3bc2a	2025-10-08 16:17:30.557	2049-04-15 02:16:15.220	?? ??# ?? ??(? ? ?? ?? ?" ?? ?? ?? ??) ??]	2,16922
9	24	d17a7908-e333-4fe6-80a8-0b4a2b32511b	2025-10-08 15:39:46.947	1967-07-31 22:31:41.983	??????} ?? ? ???, ??/ ?? ? ?? ??* ? ??? ?? ????(?* ?* ?=	-1,4963
10	23	759274e8-306a-482a-9509-9336499117f9	2025-10-08 15:25:20.000	1960-04-10 12:02:41.667	??% ?? ?# ?? ?? ?% ?? ?? ??^ ??~ ?? ? ??/ ?? ?(?? ?&...	-2,2433
11	22	93fde537-4992-4292-a015-8f1591c74647	2025-10-07 17:29:23.160	2049-05-30 14:12:04.507	??????? ?????? ???????? ??????????? ?????? ?????...	3,93017
12	21	5bfe9b2d-0662-4828-8378-69fc09739d5a	2025-10-07 13:51:59.663	2089-10-19 06:10:53.773	??????????? ??????(?????????? ??? ?????? ??? ???...	-5,5921
13	20	d379d2f0-4d68-49ad-b020-597ed89ca578	2025-10-07 12:06:30.537	1926-03-03 11:57:38.520	Blue{ Sheep Lime~ Green@ Horse Horse Strawberry Cat>...	477272

Consulta ejecutada correctamente. | PGES632F5X3 (16.0 RTM) | PGES632F5X3\IElguezaba... | GenericUAServer | 00:00:00 | 32 filas

The OPC nodes are defined like this.

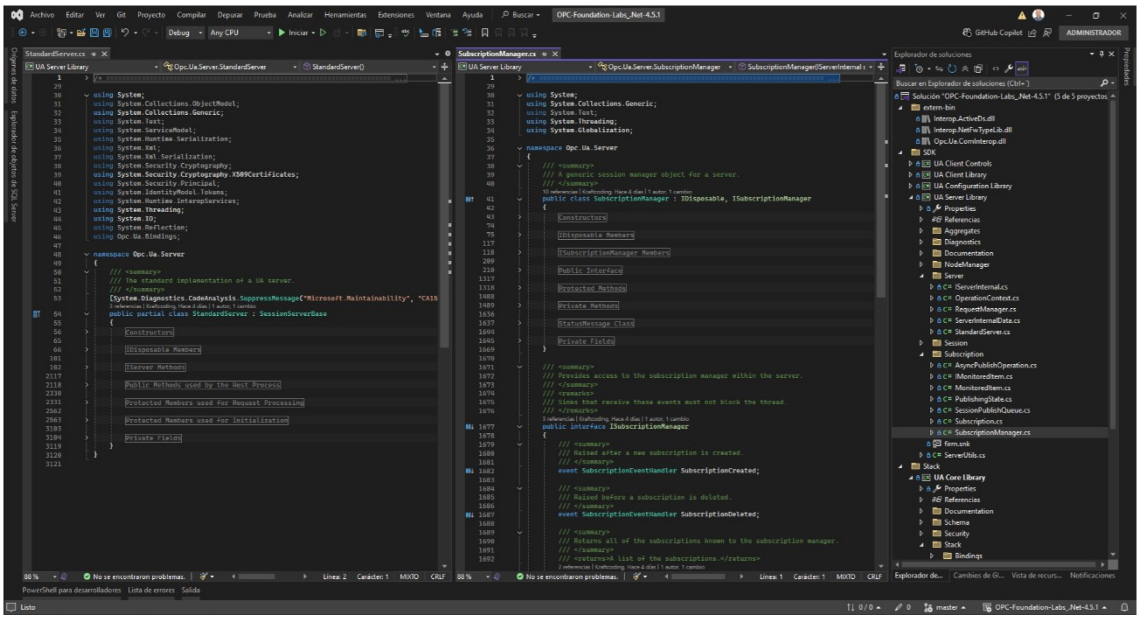


```xml
<Extensions>
    <ua:XmlElement>
        <ServerConfigManager xmlns="http://opcfoundation.org/LocalServer">
            <ServerNodes>
                <ServerNode>
                    <FolderName>Folder 1</FolderName>
                    <Bool>BoolFieldName1</Bool>
                    <String>StringFieldName1</String>
                    <Byte>ByteFieldName1</Byte>
                    <!--<ByteString>ByteStringFieldName1</ByteString>-->
                    <DateTime>DateTimeFieldName1</DateTime>
                    <Double>DoubleFieldName1</Double>
                </ServerNode>
                <ServerNode>
                    <FolderName>Folder 2</FolderName>
                    <Bool>BoolFieldName2</Bool>
                    <String>StringFieldName2</String>
                    <Byte>ByteFieldName2</Byte>
                    <!--<ByteString>ByteStringFieldName2</ByteString>-->
                    <DateTime>DateTimeFieldName2</DateTime>
                    <Double>DoubleFieldName2</Double>
                </ServerNode>
                <ServerNode>
                    <FolderName>Folder 3</FolderName>
                    <Bool>BoolFieldName3</Bool>
                    <String>StringFieldName3</String>
                    <Byte>ByteFieldName3</Byte>
                    <!--<ByteString>ByteStringFieldName3</ByteString>-->
                    <DateTime>DateTimeFieldName3</DateTime>
                    <Double>DoubleFieldName3</Double>
                </ServerNode>
                <ServerNode>
                    <FolderName>Folder 4</FolderName>
                    <Bool>BoolFieldName4</Bool>
                    <String>StringFieldName4</String>
                    <Byte>ByteFieldName4</Byte>
                    <!--<ByteString>ByteStringFieldName4</ByteString>-->
                    <DateTime>DateTimeFieldName4</DateTime>
                    <Double>DoubleFieldName4</Double>
                </ServerNode>
                <ServerNode>
```

RQ-03 | OPC Foundation Core

This solution contains a standard .NET 4.5.1 projects with the fundamental source code of OPC Foundation.

Contains the SDK and Stack, as we can see in the next image.



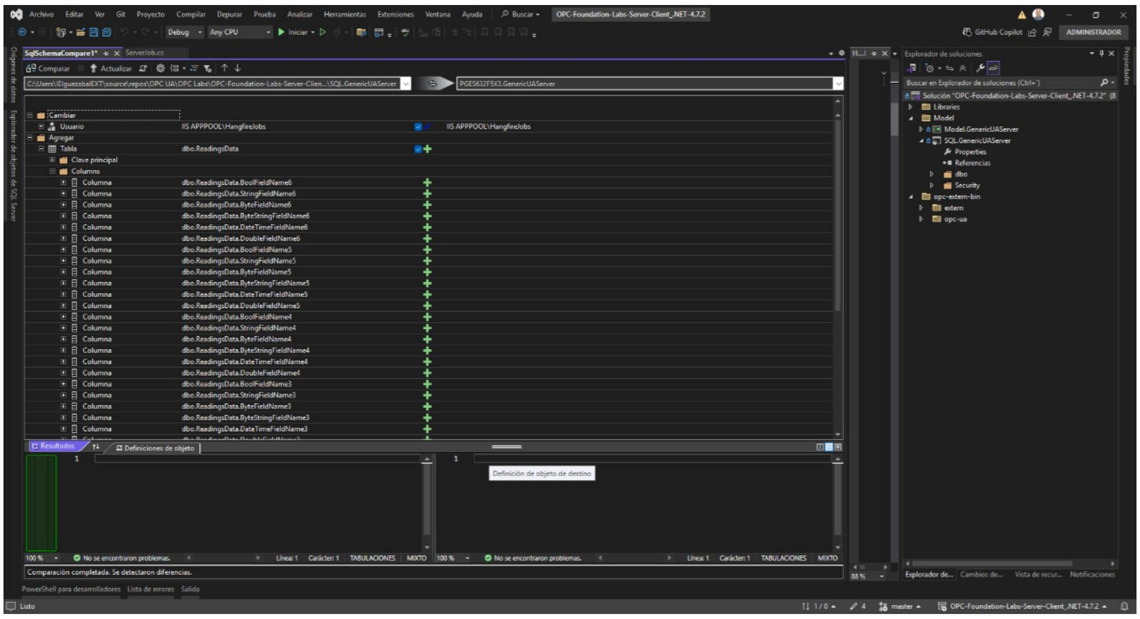
```csharp
/* ...

using System;
using System.Collections.ObjectModel;
using System.Collections.Generic;
using System.Text;
using System.ServiceModel;
using System.Runtime.Serialization;
using System.Xml;
using System.Xml.Serialization;
using System.Security.Cryptography;
using System.Security.Cryptography.X509Certificates;
using System.Security.Principal;
using System.IdentityModel.Tokens;
using System.Runtime.InteropServices;
using System.Threading;
using System.IO;
using System.Reflection;
using Opc.Ua.Bindings;

namespace Opc.Ua.Server
{
    /// <summary>
    /// The standard implementation of a UA server.
    /// </summary>
    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Maintainability", "CA15
    3 referencias | Kraftcoding, Hace 4 días | 1 autor, 1 cambio
    public partial class StandardServer : SessionServerBase
    {
        Constructors

        IDisposable Members

        IServer Methods

        Public Methods used by the Host Process

        Protected Members used for Request Processing

        Protected Members used for Initialization

        Private Fields
    }
}
```

```csharp
/* ...

using System;
using System.Collections.Generic;
using System.Text;
using System.Threading;
using System.Globalization;

namespace Opc.Ua.Server
{
    /// <summary>
    /// A generic session manager object for a server.
    /// </summary>
    10 referencias | Kraftcoding, Hace 4 días | 1 autor, 1 cambio
    public class SubscriptionManager : IDisposable, ISubscriptionManager
    {
        Constructors

        IDisposable Members

        ISubscriptionManager Members

        Public Interface

        Protected Methods

        Private Methods

        StatusMessage Class

        Private Fields
    }

    /// <summary>
    /// Provides access to the subscription manager within the server.
    /// </summary>
    /// <remarks>
    /// Sinks that receive these events must not block the thread.
    /// </remarks>
    3 referencias | Kraftcoding, Hace 4 días | 1 autor, 1 cambio
    public interface ISubscriptionManager
    {
        /// <summary>
        /// Raised after a new subscription is created.
        /// </summary>
        event SubscriptionEventHandler SubscriptionCreated;

        /// <summary>
        /// Raised before a subscription is deleted.
        /// </summary>
        event SubscriptionEventHandler SubscriptionDeleted;

        /// <summary>
        /// Returns all of the subscriptions known to the subscription manager.
        /// </summary>
        /// <returns>A list of the subscriptions.</returns>
        2 referencias | Kraftcoding, Hace 4 días | 1 autor, 1 cambio
```

```
Explorador de soluciones
Buscar en Explorador de soluciones (Ctrl+`)
Solución "OPC-Foundation-Labs_.Net-4.5.1" (5 de 5 proyectos
  extern-bin
    Interop.ActiveDs.dll
    Interop.NetFwTypeLib.dll
    Opc.Ua.ComInterop.dll
  SDK
    UA Client Controls
    UA Client Library
    UA Configuration Library
    UA Server Library
      Properties
      Referencias
      Aggregates
      Diagnostics
      Documentation
      NodeManager
      Server
        IServerInternal.cs
        OperationContext.cs
        RequestManager.cs
        ServerInternalData.cs
        StandardServer.cs
      Session
      Subscription
        AsyncPublishOperation.cs
        IMonitoredItem.cs
        MonitoredItem.cs
        PublishingState.cs
        SessionPublishQueue.cs
        Subscription.cs
        SubscriptionManager.cs
      firm.snk
      ServerUtils.cs
  Stack
    UA Core Library
      Properties
      Referencias
      Documentation
      Schema
      Security
      Stack
        Bindings
```

Deployment

To deploy the solution first launch the compare schema.


SqlSchemaCompare1*

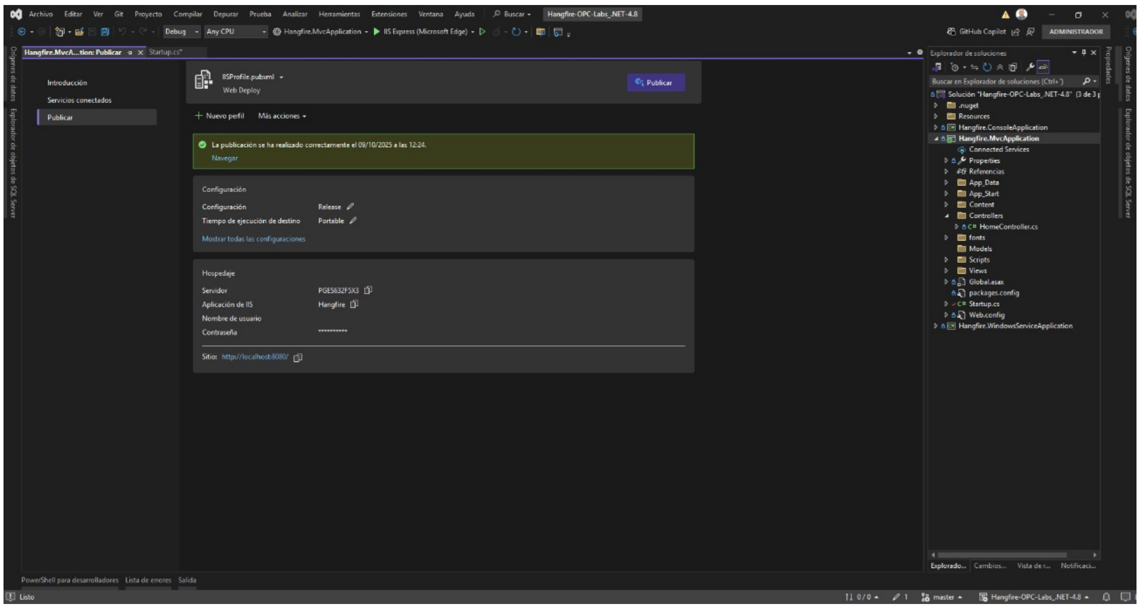
C:\Users\IElguezabalEXT\source\repos\OPC UA\OPC Labs\OPC-Foundation-Labs-Server-Clien...\SQL.GenericUAServer → PGES632F5X3.GenericUAServer

Cambiar			
Usuario	IIS APPPOOL\HangfireJobs	✓	IIS APPPOOL\HangfireJobs
Agregar			
Tabla	dbo.ReadingsData	✓ +	
Clave principal			
Columns			
Columna	dbo.ReadingsData.BoolFieldName6	+	
Columna	dbo.ReadingsData.StringFieldName6	+	
Columna	dbo.ReadingsData.ByteFieldName6	+	
Columna	dbo.ReadingsData.ByteStringFieldName6	+	
Columna	dbo.ReadingsData.DateTimeFieldName6	+	
Columna	dbo.ReadingsData.DoubleFieldName6	+	
Columna	dbo.ReadingsData.BoolFieldName5	+	
Columna	dbo.ReadingsData.StringFieldName5	+	
Columna	dbo.ReadingsData.ByteFieldName5	+	
Columna	dbo.ReadingsData.ByteStringFieldName5	+	
Columna	dbo.ReadingsData.DateTimeFieldName5	+	
Columna	dbo.ReadingsData.DoubleFieldName5	+	
Columna	dbo.ReadingsData.BoolFieldName4	+	
Columna	dbo.ReadingsData.StringFieldName4	+	
Columna	dbo.ReadingsData.ByteFieldName4	+	
Columna	dbo.ReadingsData.ByteStringFieldName4	+	
Columna	dbo.ReadingsData.DateTimeFieldName4	+	
Columna	dbo.ReadingsData.DoubleFieldName4	+	
Columna	dbo.ReadingsData.BoolFieldName3	+	
Columna	dbo.ReadingsData.StringFieldName3	+	
Columna	dbo.ReadingsData.ByteFieldName3	+	
Columna	dbo.ReadingsData.ByteStringFieldName3	+	
Columna	dbo.ReadingsData.DateTimeFieldName3	+	

No se encontraron problemas.

Comparación completada. Se detectaron diferencias.

Then publish the MVC application.


Hangfire.MvcA...tion: Publicar

IISProfile.pubxml
Web Deploy

La publicación se ha realizado correctamente el 09/10/2025 a las 12:24.

Configuración

Configuración	Release
Tiempo de ejecución de destino	Portable

Hospedaje

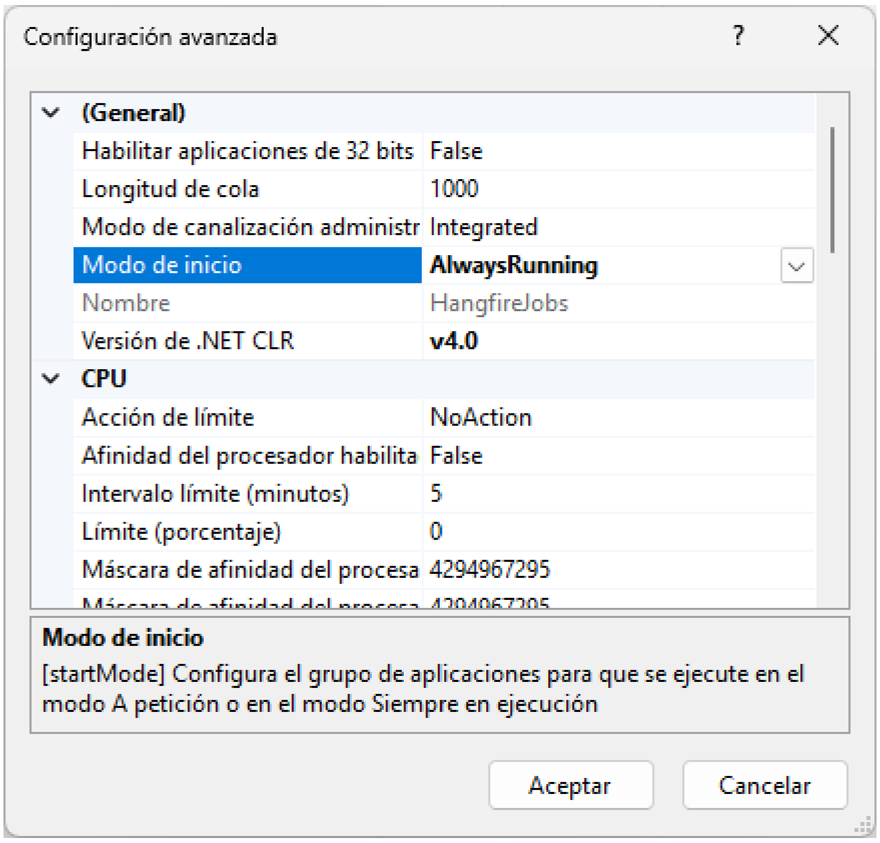
Servidor	PGES632F5X3
Aplicación de IIS	Hangfire
Nombre de usuario	
Contraseña	**********

Sitio: http://localhost:8080/

It is needed to give permissions to the IIS app-pool process to use TCP.

- netsh http add urlacl url=http://+:62543/ user=Everyone
- netsh http add urlacl url=http://+:62543/LocalServer/ user=IIS APPPOOL\HangfireJobs

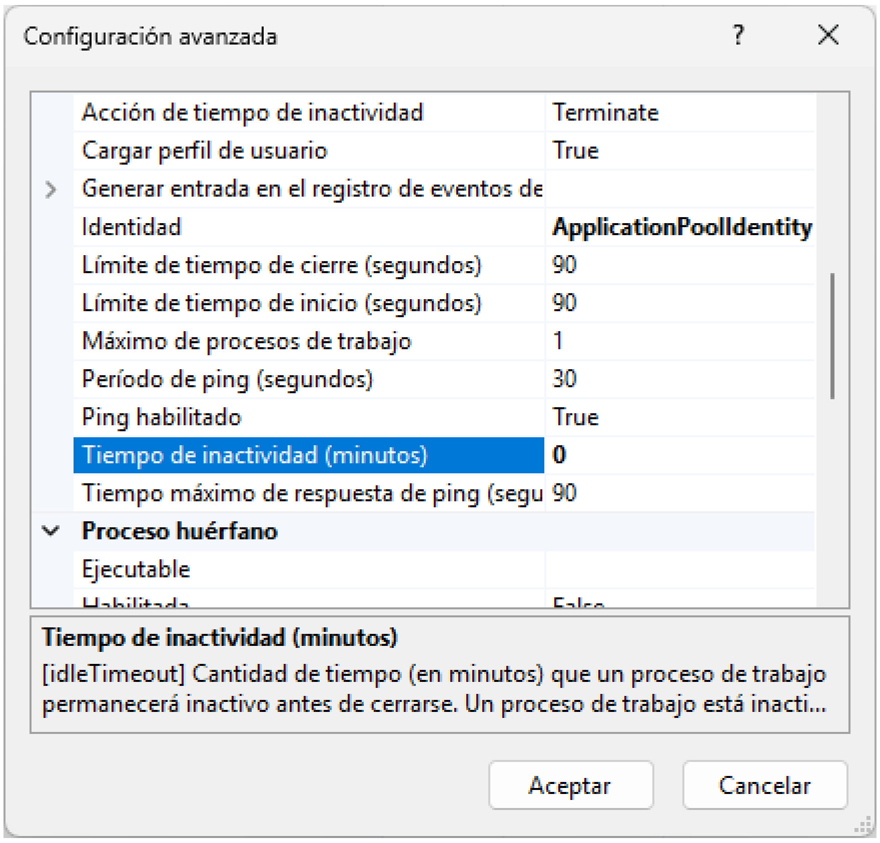
IIS and app-pool configuration should be like this.



Configuración avanzada

(General)	
Habilitar aplicaciones de 32 bits	False
Longitud de cola	1000
Modo de canalización administr	Integrated
Modo de inicio	**AlwaysRunning**
Nombre	HangfireJobs
Versión de .NET CLR	**v4.0**
CPU	
Acción de límite	NoAction
Afinidad del procesador habilita	False
Intervalo límite (minutos)	5
Límite (porcentaje)	0
Máscara de afinidad del procesa	4294967295

Modo de inicio
[startMode] Configura el grupo de aplicaciones para que se ejecute en el modo A petición o en el modo Siempre en ejecución



Configuración avanzada

Acción de tiempo de inactividad	Terminate
Cargar perfil de usuario	True
Generar entrada en el registro de eventos de	
Identidad	**ApplicationPoolIdentity**
Límite de tiempo de cierre (segundos)	90
Límite de tiempo de inicio (segundos)	90
Máximo de procesos de trabajo	1
Período de ping (segundos)	30
Ping habilitado	True
Tiempo de inactividad (minutos)	**0**
Tiempo máximo de respuesta de ping (segu	90
Proceso huérfano	
Ejecutable	

Tiempo de inactividad (minutos)
[idleTimeout] Cantidad de tiempo (en minutos) que un proceso de trabajo permanecerá inactivo antes de cerrarse. Un proceso de trabajo está inacti...

Application Pool Set "Preload Enabled". To set the application pool to "Preload Enabled" in IIS, follow these steps:

- Open the applicationHost.config file located in the %WINDIR%\system32\inetsrv\config\applicationHost.config directory.
- Locate the application pool setting and set preloadEnabled to true.
- Restart IIS to apply the changes.
- Optionally, configure warm-up requests to ensure all necessary components are loaded during the initialization phase.

This configuration allows IIS to send a "fake" request to the application when the associated application pool starts up, ensuring that the application is warmed up before any real traffic is received. This is particularly useful for applications with heavy startup processes, such as those requiring multiple service initializations or database connections.

Software life cycle management

TO DO.

Annexes

Improvements and performance

Hangfire application always running on IIS

Look up if there is any way to ensure that Hangfire application is always running on IIS.

This next article contains a detailed explanation about related topics:

- How to Make Sure Your ASP.NET Core Keep Running on IIS - ASP.NET Hosting Tips & Guides

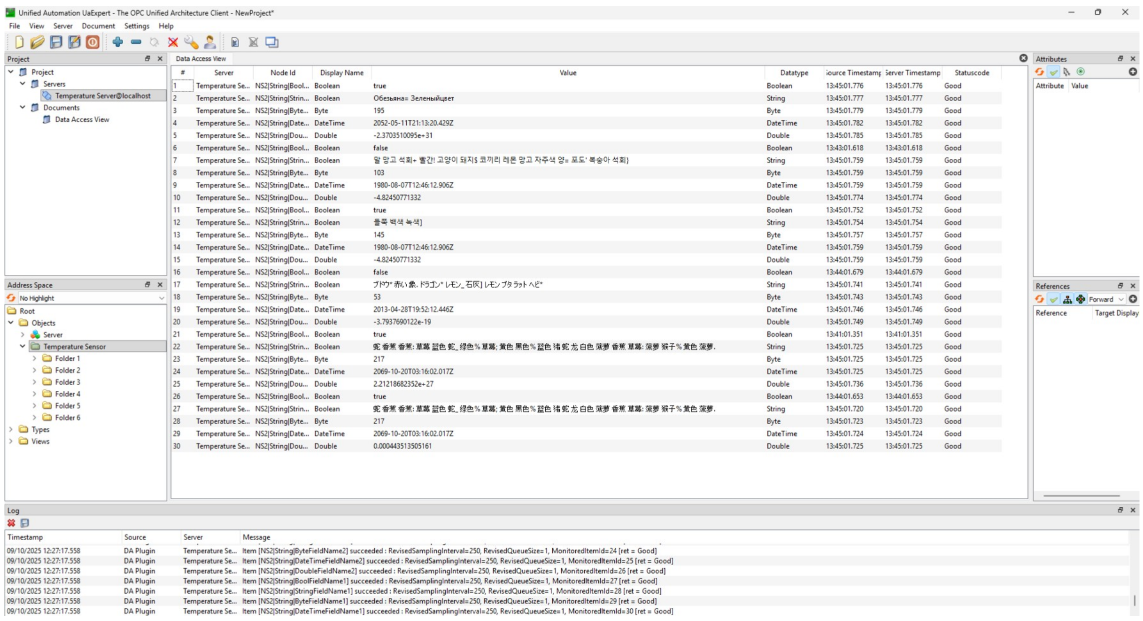
Jobs chain of responsibility pattern

Must be look up if further integration could be attained between the OPC Jobs and Hangfire chain of responsibility patterns.

Additional tools used for testing

UaExpert

For initial steps of development and testing UaExpert client application was used.



#	Server	Node Id	Display Name	Value	Datatype	Source Timestamp	Server Timestamp	Statuscode
1	Temperature Se...	NS2|String|Bool...	Boolean	true	Boolean	13:45:01.776	13:45:01.776	Good
2	Temperature Se...	NS2|String|Strin...	Boolean	Обезьяна= Зеленыйцвет	String	13:45:01.777	13:45:01.777	Good
3	Temperature Se...	NS2|String|Byte...	Byte	195	Byte	13:45:01.779	13:45:01.779	Good
4	Temperature Se...	NS2|String|Date...	DateTime	2052-05-11T21:13:20.429Z	DateTime	13:45:01.782	13:45:01.782	Good
5	Temperature Se...	NS2|String|Dou...	Double	-2.3703510095e+31	Double	13:45:01.785	13:45:01.785	Good
6	Temperature Se...	NS2|String|Bool...	Boolean	false	Boolean	13:43:01.618	13:43:01.618	Good
7	Temperature Se...	NS2|String|Strin...	Boolean	말 망고 석회+ 빨간! 고양이 돼지$ 코끼리 레몬 망고 자주색 양= 포도' 복숭아 석회}	String	13:45:01.759	13:45:01.759	Good
8	Temperature Se...	NS2|String|Byte...	Byte	103	Byte	13:45:01.759	13:45:01.759	Good
9	Temperature Se...	NS2|String|Date...	DateTime	1980-08-07T12:46:12.906Z	DateTime	13:45:01.759	13:45:01.759	Good
10	Temperature Se...	NS2|String|Dou...	Double	-4.82450771332	Double	13:45:01.774	13:45:01.774	Good
11	Temperature Se...	NS2|String|Bool...	Boolean	true	Boolean	13:45:01.752	13:45:01.752	Good
12	Temperature Se...	NS2|String|Strin...	Boolean	들쭉 백색 녹색]	String	13:45:01.754	13:45:01.754	Good
13	Temperature Se...	NS2|String|Byte...	Byte	145	Byte	13:45:01.757	13:45:01.757	Good
14	Temperature Se...	NS2|String|Date...	DateTime	1980-08-07T12:46:12.906Z	DateTime	13:45:01.759	13:45:01.759	Good
15	Temperature Se...	NS2|String|Dou...	Double	-4.82450771332	Double	13:45:01.759	13:45:01.759	Good
16	Temperature Se...	NS2|String|Bool...	Boolean	false	Boolean	13:44:01.679	13:44:01.679	Good
17	Temperature Se...	NS2|String|Strin...	Boolean	ブドウ* 赤い 象. ドラゴン* レモン_石灰] レモン ブタ ラット ヘビ*	String	13:45:01.741	13:45:01.741	Good
18	Temperature Se...	NS2|String|Byte...	Byte	53	Byte	13:45:01.743	13:45:01.743	Good
19	Temperature Se...	NS2|String|Date...	DateTime	2013-04-28T19:52:12.446Z	DateTime	13:45:01.746	13:45:01.746	Good
20	Temperature Se...	NS2|String|Dou...	Double	-3.7937690122e-19	Double	13:45:01.749	13:45:01.749	Good
21	Temperature Se...	NS2|String|Bool...	Boolean	true	Boolean	13:41:01.351	13:41:01.351	Good
22	Temperature Se...	NS2|String|Strin...	Boolean	蛇 香蕉 香蕉: 草莓 藍色 蛇_绿色%草莓; 黃色 黑色%藍色 猪 蛇 龙 白色 菠萝 香蕉 草莓: 菠萝 猴子%黃色 菠萝.	String	13:45:01.725	13:45:01.725	Good
23	Temperature Se...	NS2|String|Byte...	Byte	217	Byte	13:45:01.725	13:45:01.725	Good
24	Temperature Se...	NS2|String|Date...	DateTime	2069-10-20T03:16:02.017Z	DateTime	13:45:01.725	13:45:01.725	Good
25	Temperature Se...	NS2|String|Dou...	Double	2.21218682352e+27	Double	13:45:01.736	13:45:01.736	Good
26	Temperature Se...	NS2|String|Bool...	Boolean	true	Boolean	13:44:01.653	13:44:01.653	Good
27	Temperature Se...	NS2|String|Strin...	Boolean	蛇 香蕉 香蕉: 草莓 藍色 蛇_绿色%草莓; 黃色 黑色%藍色 猪 蛇 龙 白色 菠萝 香蕉 草莓: 菠萝 猴子%黃色 菠萝.	String	13:45:01.720	13:45:01.720	Good
28	Temperature Se...	NS2|String|Byte...	Byte	217	Byte	13:45:01.723	13:45:01.723	Good
29	Temperature Se...	NS2|String|Date...	DateTime	2069-10-20T03:16:02.017Z	DateTime	13:45:01.724	13:45:01.724	Good
30	Temperature Se...	NS2|String|Dou...	Double	0.000443513505161	Double	13:45:01.725	13:45:01.725	Good

```
Timestamp                Source      Server             Message
09/10/2025 12:27:17.558  DA Plugin   Temperature Se...  Item [NS2|String|ByteFieldName2] succeeded : RevisedSamplingInterval=250, RevisedQueueSize=1, MonitoredItemId=24 [ret = Good]
09/10/2025 12:27:17.558  DA Plugin   Temperature Se...  Item [NS2|String|DateTimeFieldName2] succeeded : RevisedSamplingInterval=250, RevisedQueueSize=1, MonitoredItemId=25 [ret = Good]
09/10/2025 12:27:17.558  DA Plugin   Temperature Se...  Item [NS2|String|DoubleFieldName2] succeeded : RevisedSamplingInterval=250, RevisedQueueSize=1, MonitoredItemId=26 [ret = Good]
09/10/2025 12:27:17.558  DA Plugin   Temperature Se...  Item [NS2|String|BoolFieldName1] succeeded : RevisedSamplingInterval=250, RevisedQueueSize=1, MonitoredItemId=27 [ret = Good]
09/10/2025 12:27:17.558  DA Plugin   Temperature Se...  Item [NS2|String|StringFieldName1] succeeded : RevisedSamplingInterval=250, RevisedQueueSize=1, MonitoredItemId=28 [ret = Good]
09/10/2025 12:27:17.558  DA Plugin   Temperature Se...  Item [NS2|String|ByteFieldName1] succeeded : RevisedSamplingInterval=250, RevisedQueueSize=1, MonitoredItemId=29 [ret = Good]
09/10/2025 12:27:17.558  DA Plugin   Temperature Se...  Item [NS2|String|DateTimeFieldName1] succeeded : RevisedSamplingInterval=250, RevisedQueueSize=1, MonitoredItemId=30 [ret = Good]
```