

OPC FOUNDATION CLIENT-SERVER SOLUTION

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

Contents

Requirements.....	2
Overview.....	3
Scope.....	5
RQ-01 MVC Hangfire Management Application.....	6
RQ-02 OPC Foundation Client-Server and DB Model Layer.....	8
RQ-03 OPC Foundation Core.....	10
Deployment.....	11
Software life cycle management.....	14
Annexes.....	15
Improvements and performance.....	15
Hangfire application always running on IIS.....	15
Jobs chain of responsibility pattern.....	15
Additional tools used for testing.....	15
UaExpert.....	15

Doc. Version	Date	Author	Description
1.0	2025-10-09	Iker Elg. Alzaa	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.

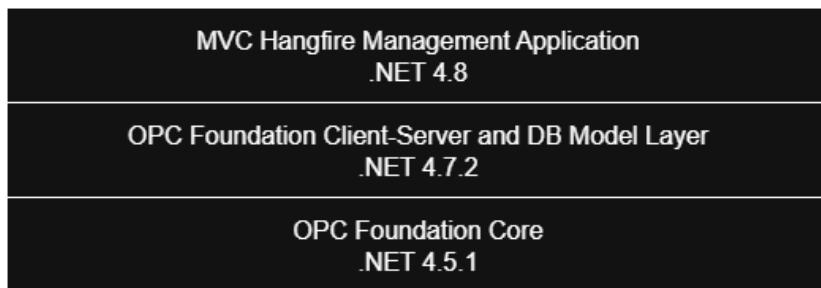
Overview

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.



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.

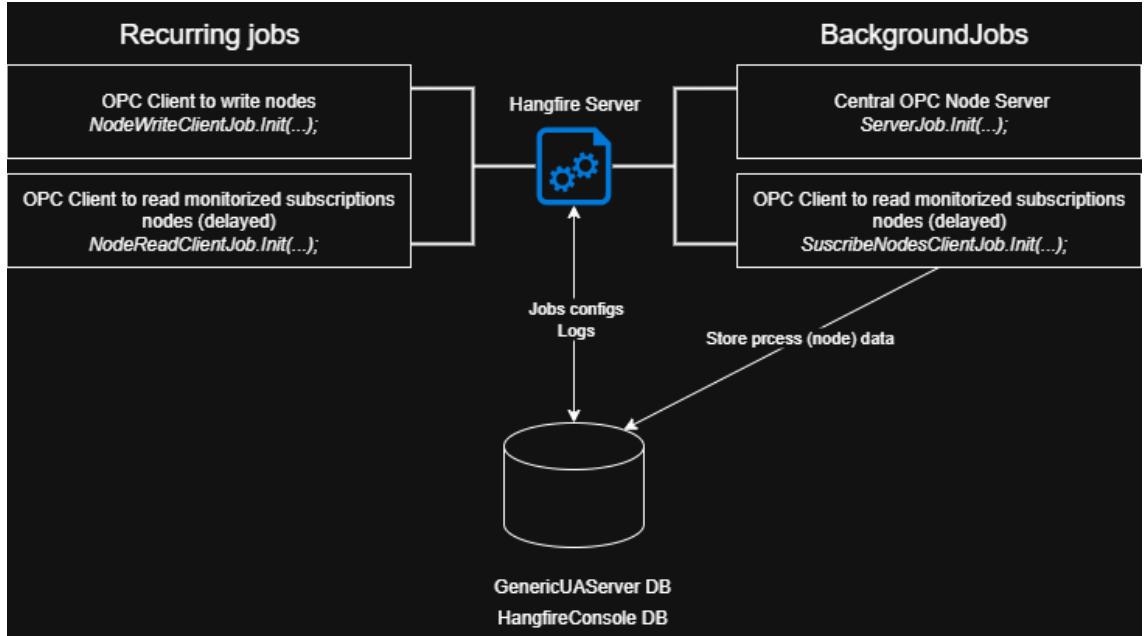
This latest version of the Hangfire server includes:

- 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 overview.



Scope

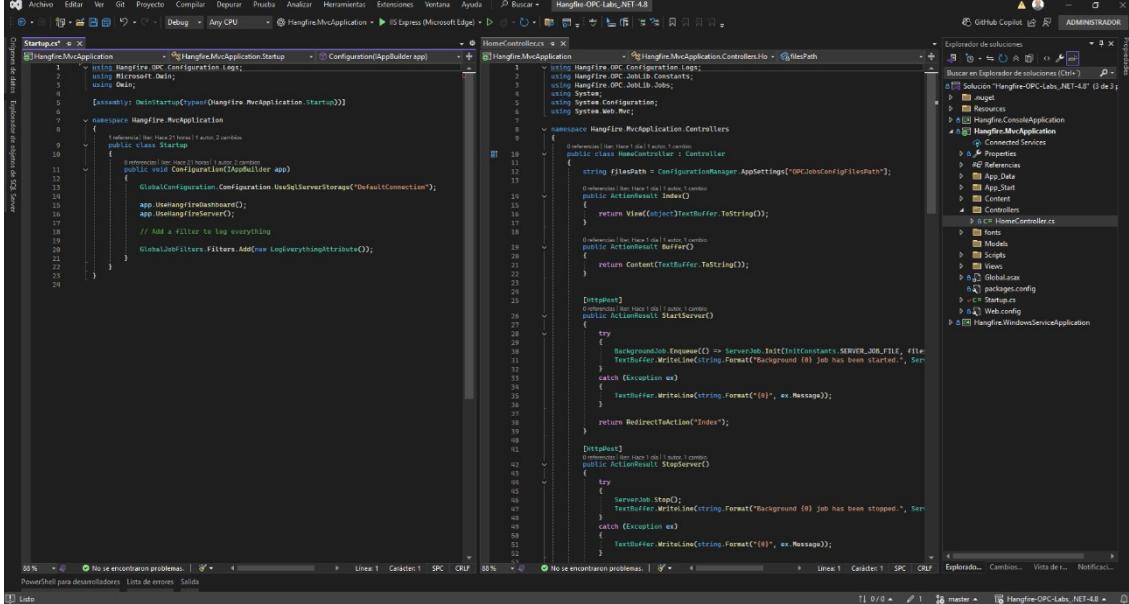
Hangfire is an open-source library for .NET applications that facilitates the execution of background jobs. It allows you to schedule jobs to run at a specific time, recurring, or immediately, all without the need for external services, and with an administration panel to monitor their status.

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.

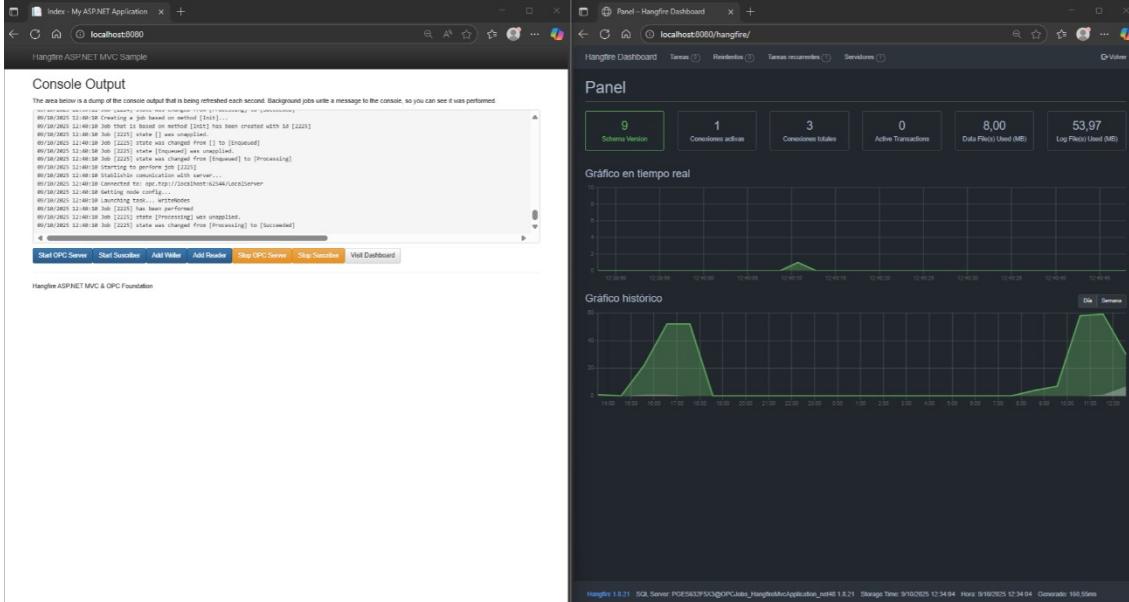
RQ-01 | MVC Hangfire Management Application

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.



The screenshot shows the Visual Studio IDE with two open files: `HangfireMvcApplication.cs` and `HomeController.cs`. The `HangfireMvcApplication.cs` file contains the startup configuration for the application, including the connection string and the configuration of the `GlobalConfiguration` class. The `HomeController.cs` file contains the implementation of the `HomeController`, which handles requests for the home page and manages background jobs. The code uses the `Hangfire` library to interact with the database and perform tasks.

Here the screens of the application:



The screenshot displays two browser windows. The left window shows the "Console Output" page of the Hangfire ASP.NET MVC Sample application, displaying log messages related to job creation and processing. The right window shows the "Panel - Hangfire Dashboard" page, which provides real-time and historical performance metrics for the Hangfire server, such as the number of active connections and the size of the log file.

The image displays two screenshots of the Hangfire Dashboard, both titled "Tareas en proceso" and "Tareas recurrentes".

Tareas en proceso (Left Screenshot):

- Header:** Tareas en proceso - Hangfire Dashboard | localhost:8080/hangfire/jobs/processsing
- Section:** Tareas en proceso
- Filters:** Programadas, Procesando, Completadas, Con errores, Eliminado, En espera.
- Table Headers:** Id, Servidor, Tarea, Iniciado.
- Data:**
 - #2211 POE5632FSX3 14636 ServJob.Inst hace 19 minutos
 - #2212 POE5632FSX3 14636 SuscribeModelClientJob.Inst hace 19 minutos
- Total elementos:** 2

Tareas recurrentes (Right Screenshot):

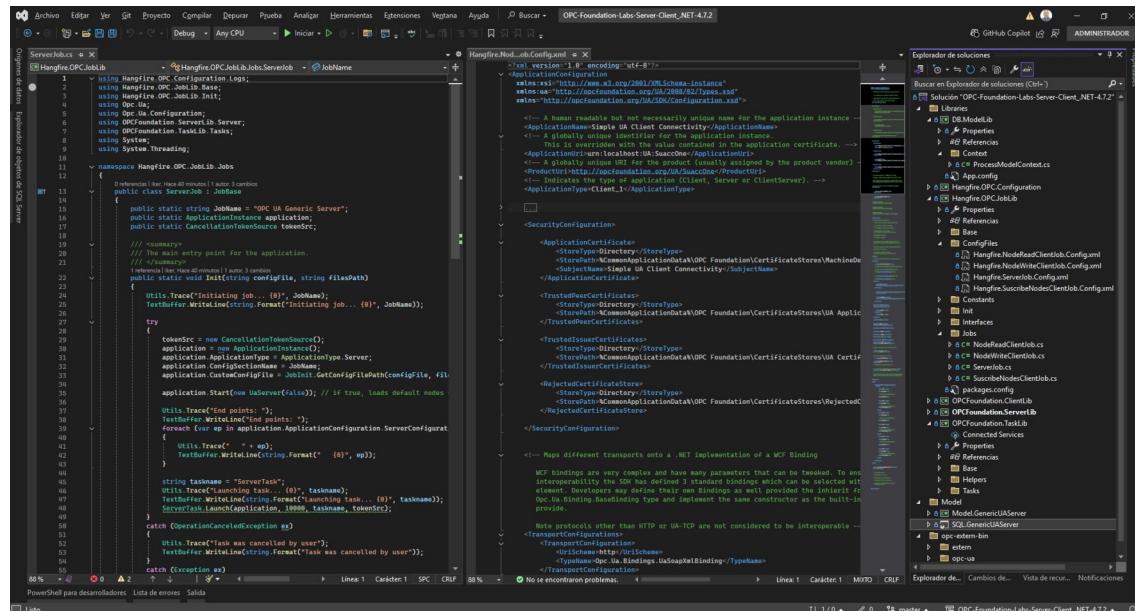
- Header:** Tareas recurrentes - Hangfire Dashboard | localhost:8080/hangfire/recuring
- Section:** Tareas recurrentes
- Filters:** Localizar tarea, Eliminar.
- Table Headers:** Id, Cron, Zona horaria, Tarea, Próxima ejecución, Última ejecución, Creado.
- Data:**
 - 3a8dd73-66d3-4b0c-8d45-127 67143462 UTC NodeWriteClientJob.Inst en unos segundos hace 19 minutos
- Total elementos:** 1

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 classes to interact with the SDK / Stack
 - OPCFoundation.ServerLib → Opc server procedures and classes 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



The screenshot shows the SSMS interface with the following details:

- Toolbar:** Archivo, Editar, Ver, Consulta, Git, Proyecto, Herramientas, Extensiones, Ventana, Ayuda, Solución.
- Object Explorer:** Shows 'GenericUAServer' as the current database.
- Query Editor:** Title bar says 'SQLQuery1.sql...abalaEXT (68)*'. The query is:

```
1  SELECT id, idProcess, processStartDate, DateTimeFieldName6, StringFieldName6, DoubleFieldName6
2  FROM [GenericUAServer].[dbo].[ReadingsData]
3  ORDER BY id DESC
```
- Status Bar:** Shows '81%', 'No se encontraron problemas.', 'Línea: 4', 'Carácter: 3', 'TABULACIONES', 'CRLF'.
- Results Grid:** Contains 13 rows of data from the 'ReadingsData' table, with columns: id, idProcess, processStartDate, DateTimeFieldName6, StringFieldName6, DoubleFieldName6.
- Bottom Status:** 'Consulta ejecutada correctamente.'

```
<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

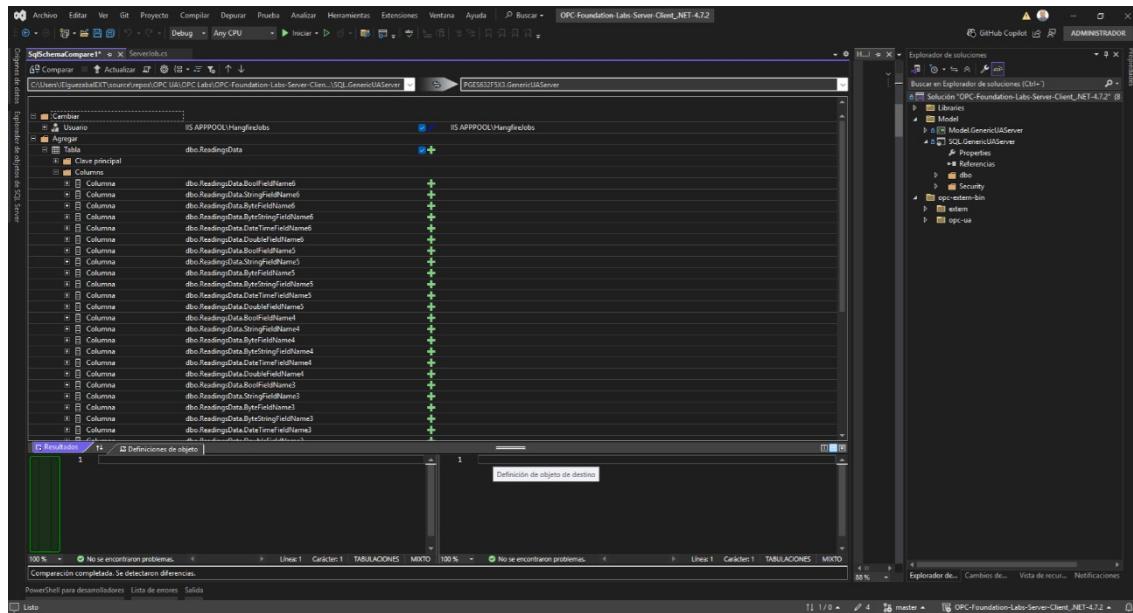
This solution contains a standard .NET 4.5.1 projects with the fundamental source code of OPC Foundation.

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

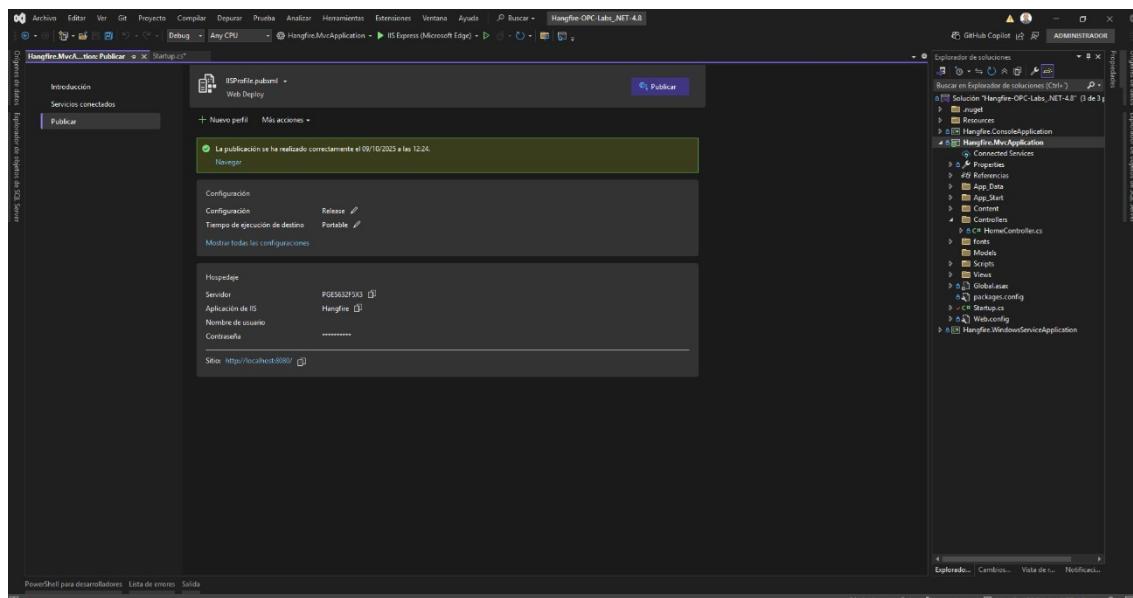
The screenshot shows the Microsoft Visual Studio IDE interface. The top menu bar includes Archivo, Editor, Ver, Proyectos, Compilar, Depurar, Prueba, Analizar, Herramientas, Extensiones, Ventana, Ayuda, and Búscar. The title bar indicates the project is "OPC Foundation Lab .Net 4.5.1". The left sidebar displays the "Explorador de soluciones" (Solution Explorer) with several projects listed under "OPC-Foundation-Labs-.Net-4.5.1": UA Server Library, SubscriptionManager, UA Client Controls, UA Client Library, UA Configuration Library, UA Service Library, UA Session, UA Subscription, UA SessionPubSub, UA Subscriptions, UA SessionSubscriptionManager, UA Core Library, and UA Security. The right sidebar shows the "Explorador de tipos" (Type Explorer). The main code editor window displays the "SubscriptionManager.cs" file, which contains C# code for managing subscriptions. The code includes imports for System, System.Collections.ObjectModel, System.Diagnostics, System, System.Threading, System.Globalization, System.Runtime.Serialization, System.Security.Cryptography.X509Certificates, System.Security.Principal, System.ServiceModel, System.ServiceModel.Description, System.Runtime.InteropServices, System.Threading.Tasks, System.IIS, System.Reflection, and System.Runtime Bindings. The class `SubscriptionManager` implements `IDisposable` and `ISubscriptionManager`. It has a constructor that takes an `IExternalObject` parameter. The `OnNewEvent` method is annotated with `/// <summary>`, `/// A generic session manager object for a server.`, and `/// <param name="e"> An event containing the new data to write. </param>`. The `OnNewEvent` method calls `OnNewEventCore` and returns a `Task<EventArgs>`. The `OnNewEventCore` method is annotated with `/// <summary>`, `/// Allow access to the subscription manager within the server.`, `/// <param name="e"> An event that receive these events must not block the thread.`, and `/// <param name="cancelToken"> A cancellation token used to cancel the operation.`. It returns a `Task<EventArgs>`. The `OnSubscriptionCreated` event is annotated with `/// <summary>` and `/// Raised after a new subscription is created.`. The `OnSubscriptionDeleted` event is annotated with `/// <summary>` and `/// Raised before a subscription is deleted.`. The `OnSubscriptionChanged` event is annotated with `/// <summary>` and `/// Returns all the subscriptions known to the subscription manager.`. The `GetSubscriptions` method is annotated with `/// <summary>` and `/// Returns a list of the subscriptions. </returns>`.

Deployment

To deploy the solution first launch the compare schema.



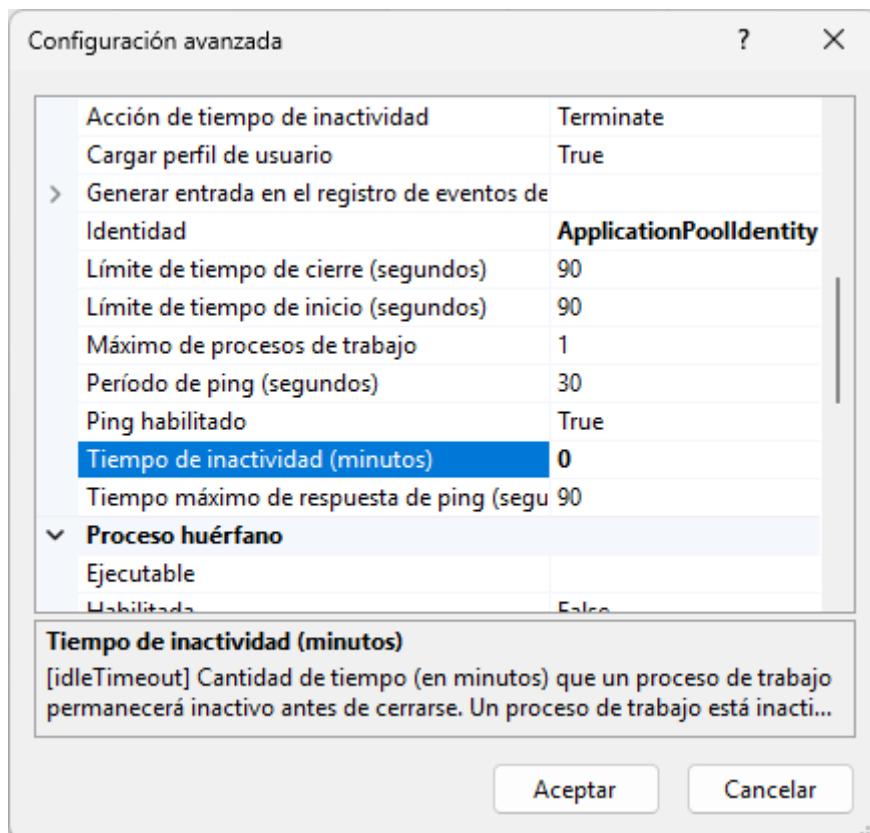
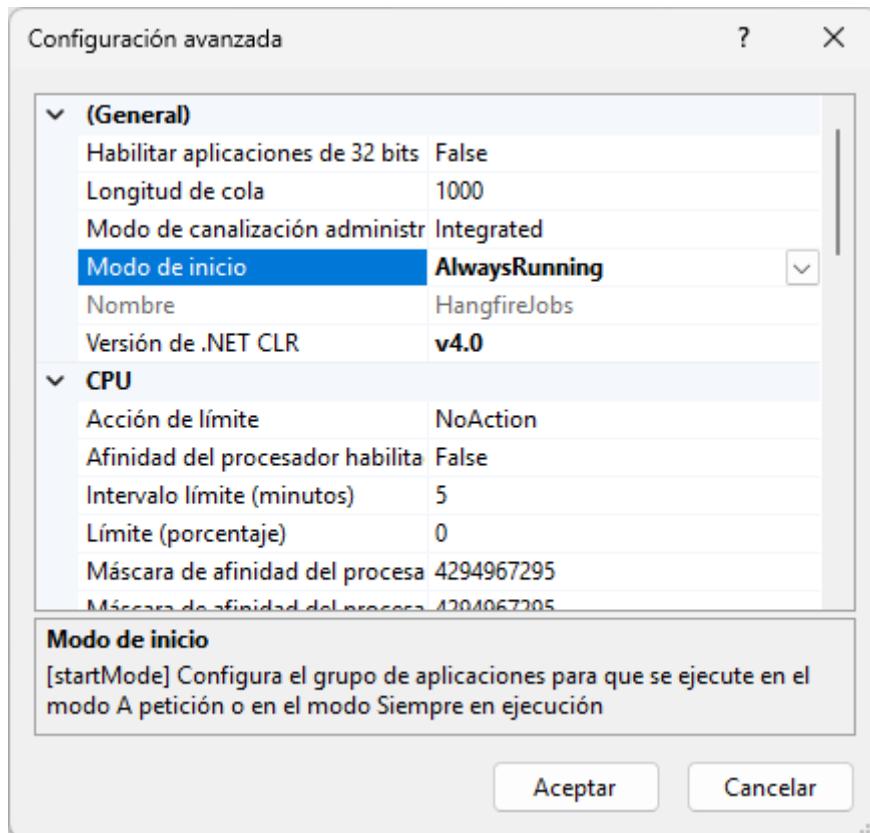
Then publish the MVC application.



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.



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.

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.

