

Reutilización de componentes de la plataforma ASIO

Juan Valer y Javier Ruiz

RIAM Intelearning Lab – GNOSS

juanvaler@gnoss.com y javierruiz@gnoss.com



HĒRCULES



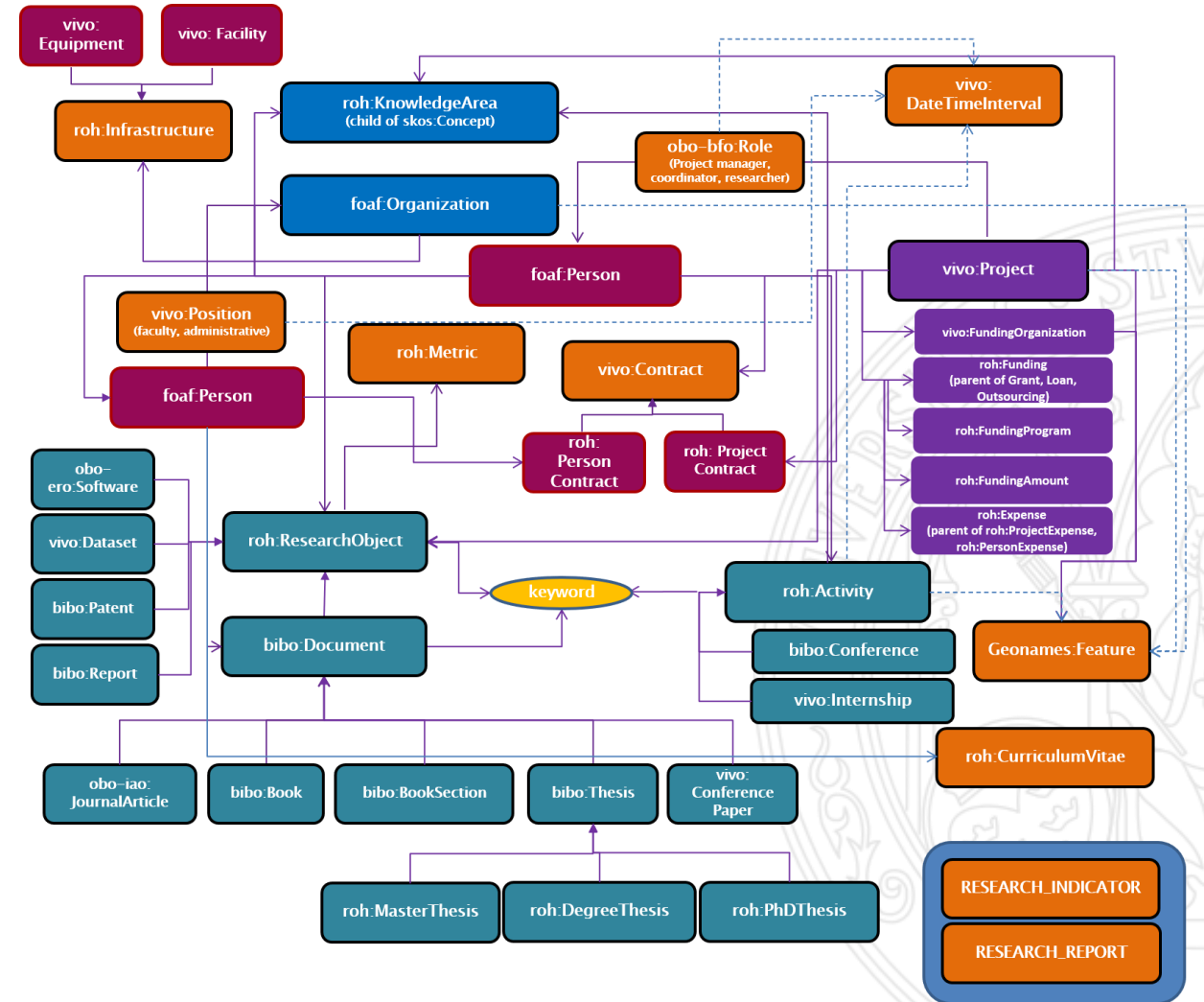
Introducción

- ☐ Definiciones
- ☐ Arquitectura de aplicaciones ASIO
- ☐ Cómo reutilizar los componentes ASIO como librerías
- ☐ Taller de reutilización



Definiciones: Ontología

- Descripción formal sobre un modelo de dominio concreto. En este caso, Investigadores, proyectos de investigación, etc.



Definiciones: OWL

- Lenguaje de definición de ontologías para la Web (Ontology Web Language). Basado en XML

```
<owl:ObjectProperty rdf:about="http://purl.org/roh#hasKnowledgeArea">
  <owl:inverseOf rdf:resource="http://purl.org/roh#knowledgeAreaOf"/>
  <rdfs:range rdf:resource="http://www.w3.org/2004/02/skos/core#Concept"/>
  <rdfs:comment xml:lang="en">Some entity related to any Concept.</rdfs:comment>
  <rdfs:label xml:lang="en">has knowledge area</rdfs:label>
  <rdfs:label xml:lang="es">tiene area de conocimiento</rdfs:label>
</owl:ObjectProperty>
<!-- http://purl.org/roh#hasMetric -->
<owl:ObjectProperty rdf:about="http://purl.org/roh#hasMetric">
  <owl:inverseOf rdf:resource="http://purl.org/roh#metricOf"/>
  <rdfs:domain rdf:resource="http://purl.org/roh#ResearchObject"/>
  <rdfs:range rdf:resource="http://purl.org/roh#Metric"/>
  <rdfs:comment xml:lang="en">A Metric which quantifies a Research Object.</rdfs:comment>
  <rdfs:label xml:lang="en">has metric</rdfs:label>
  <rdfs:label xml:lang="es">tiene métrica</rdfs:label>
</owl:ObjectProperty>
```

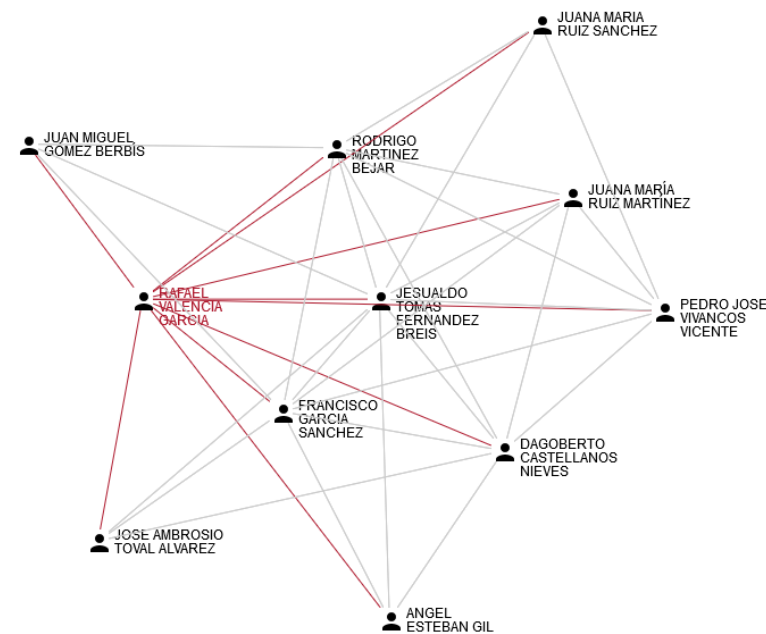
Definiciones: RDF

- Resource Description Framework. Lenguaje de definición de **instancias** de las ontología

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/">
  <rdf:Description rdf:about="http://graph.um.es/res/researcher/0000-0001-8055-6823">
    <foaf:name>Diego López de Ipiña</foaf:name>
  </rdf:Description>
</rdf:RDF>
```

Definiciones: RDF Store

- Almacén RDF o base de datos de tipo Grafo.
- La información se almacena en formato triple: Sujeto – predicado – objeto



`<http://graph.um.es/res/researchobject/10.3390/en13071700> dc:title "Socio-Economic Effect on ICT-Based Persuasive Interventions Towards Energy Efficiency in Tertiary Buildings"`
`<http://graph.um.es/res/researchobject/10.3390/en13071700> dc:author <http://graph.um.es/res/researcher/0000-0001-8055-6823>`

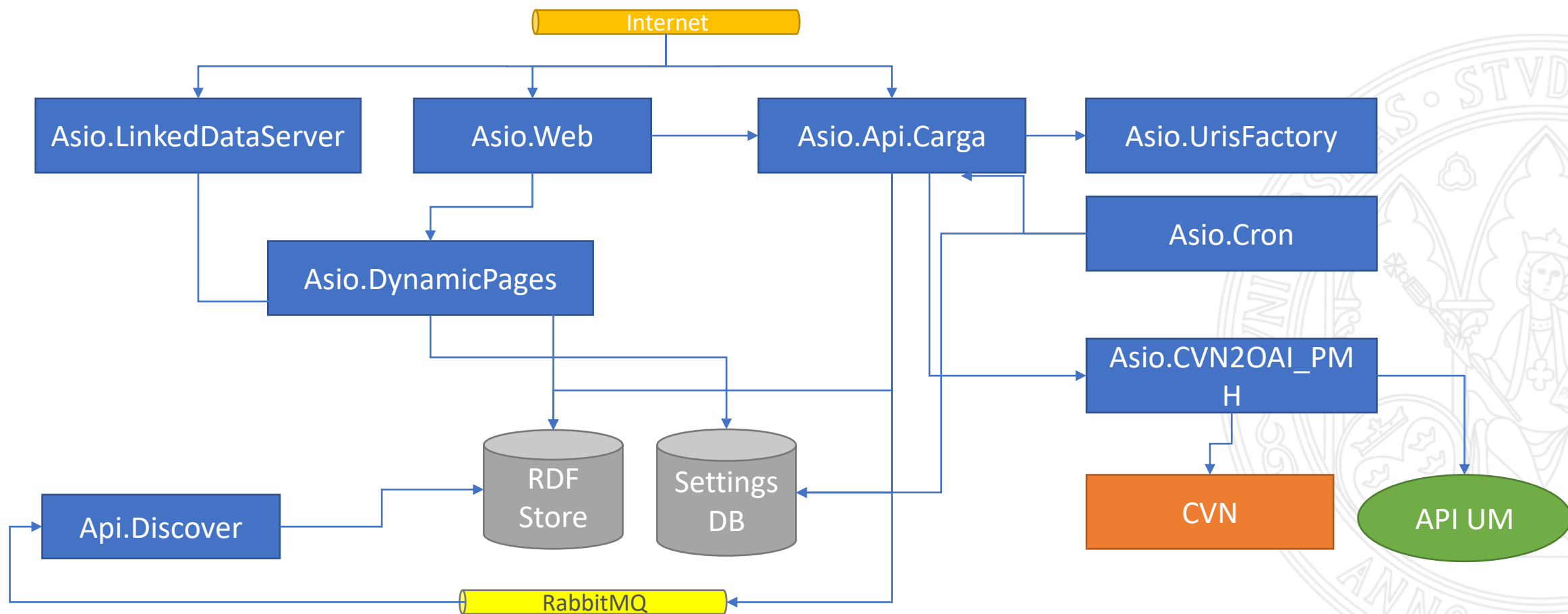
Requisitos funcionales

- Crear un grafo de conocimiento para los proyectos de investigación que se realizan en las universidades pertenecientes a la CRUE
- Identificar unívocamente a cada entidad, sin generar duplicidades
- Tener un sistema de descubrimiento y reconciliación de entidades
- Crear un sistema automatizado de integración de nuevas fuentes de datos, basado en OAI-PMH
- Ofrecer un interfaz Web que permita a los usuarios navegar por el grafo de conocimiento (Linked Data Server)
- Crear un interfaz Web que permita administrar todas las funcionalidades desarrolladas

Requisitos funcionales

- Crear un benchmark de bases de datos RDF (RDF Stores), que permita establecer a cada universidad sus propios criterios para decidir que RDF Store elegir
- Por lo tanto, el sistema debe ser totalmente agnóstico del sistema RDF que se use.
- Crear una interfaz por la cuál se puedan añadir nuevas páginas de manera dinámica, consultando cualquiera de los API del ASIO
- Generar una carga inicial, con los datos de la Universidad de Murcia
- Crear un sistema de validación de entidades, que verifique que éstas siempre cumplen la ontología ROH.

Arquitectura ASIO



Aplicaciones ASIO: Asio.Api.Carga

Es un servicio web que contienen 4 controladores, utilizados cada uno de ellos para su propio propósito:

- etlController: Contiene los procesos ETL (Extract, Transform and Load) necesarios para la carga de datos.
- repositoryController: Contiene los procesos necesarios para la gestión de los repositorios OAI-PMH (creación, modificación, eliminación...).
- syncController: Contiene los procesos necesarios para la ejecución de las sincronizaciones.
- ValidationController: Contiene los procesos necesarios para la gestión de las validaciones (creación, modificación, eliminación...). La carpeta Validaciones contiene información sobre los shapes SHACL definidos para validar.

Aplicaciones ASIO: Asio.Api.Discover

El API Descubrimiento ofrece unas funciones que son parte del proceso de carga. Estas funciones se dividen en 3 grupos:

- Reconciliación de entidades, que evita la duplicación de entidades y puede añadir datos desde otros nodos ASIO, a través de las entidades incorporadas en el nodo Unidata.
- Descubrimiento de enlaces, que genera enlaces hacia datasets externos (incluidos los de otros datasets ASIO a través del nodo Unidata), puede incorporar datos en ASIO y ofrece información de ayuda en la reconciliación de entidades.
- Detección de equivalencias, que genera equivalencias semánticas.

Aplicaciones ASIO: Asio.DynamicPages

Este módulo se usa para cargar páginas HTML y servir las a través de la web a modo de un gestor de contenidos, en el que los usuarios pueden subir páginas html con una ruta determinada y luego acceder a esas páginas a través de la web. Contiene un único controlador con cuatro acciones:

- Get Page: Que devuelve la lista de páginas sin el html.
- Get Page: Que devuelve una página dada su ruta.
- Post Page: Carga una nueva página o modifica una página existente
- Delete Page: Elimina una página dando su identificador

Aplicaciones ASIO: Asio.UrisFactory

Este módulo es el encargado de unificar las URIs de todas las entidades existentes en ASIO. Este Api contiene dos controladores.

- Factory: contiene el siguiente método para la generación de uris
- Schema: ofrece métodos para administrar la configuración de Uris

Aplicaciones ASIO: Asio.UrisFactory. Ejemplo

Quiero una URI para un **Investigador** cuyo identificador es **1234**

- Su URI debe ser: <http://data.um.es/res/researcher/1234>

Aplicaciones ASIO: GESTOR DOCUMENTACION. Ejemplo

```
@model ApiCargaWebInterface.ViewModels.CmsDataViewModel
@using CsvHelper
@using System.Globalization
@using System.IO
@using System.Text
@{
    Layout = "_Layout";
    ViewData["BodyClass"] = "fichaRecurso";
    \tstring result = Model.Results.FirstOrDefault();
    byte[] byteArray = Encoding.UTF8.GetBytes(result);
    MemoryStream stream = new MemoryStream(byteArray);
    var csvReader = new CsvReader(new StreamReader(stream), CultureInfo.InvariantCulture);
    var records = csvReader.GetRecords<PruebaSparql>();
}
```

```
@*<% sparql
select ?type count(?s) as ?count
where
{
    ?s rdf:type ?type
}
group by ?type
/%>*<
```

```
<div class="row">
    <div class="col col-12 col-lg-12 col-contenido">
        <div class="mainCol">
```

Aplicaciones ASIO: GESTOR DOCUMENTACION. Ejemplo

```
<h1>Número de entidades por tipo</h1>
<div class="contenido">
  <div class="grupo grupo-descripcion">
    <div class="items tabla">
      <div class="cabecera">
        <div class="columna">
          <p>Type</p>
        </div>
        <div class="columna">
          <p>Count</p>
        </div>
      </div>
      @foreach (var fila in records)
      {
        <div class="fila">
          <div class="columna principal">
            <p>@fila.type</p>
          </div>
          <div class="columna">
            <p>@fila.count</p>
          </div>
        </div>
      }
    </div>
  </div>
</div>
```

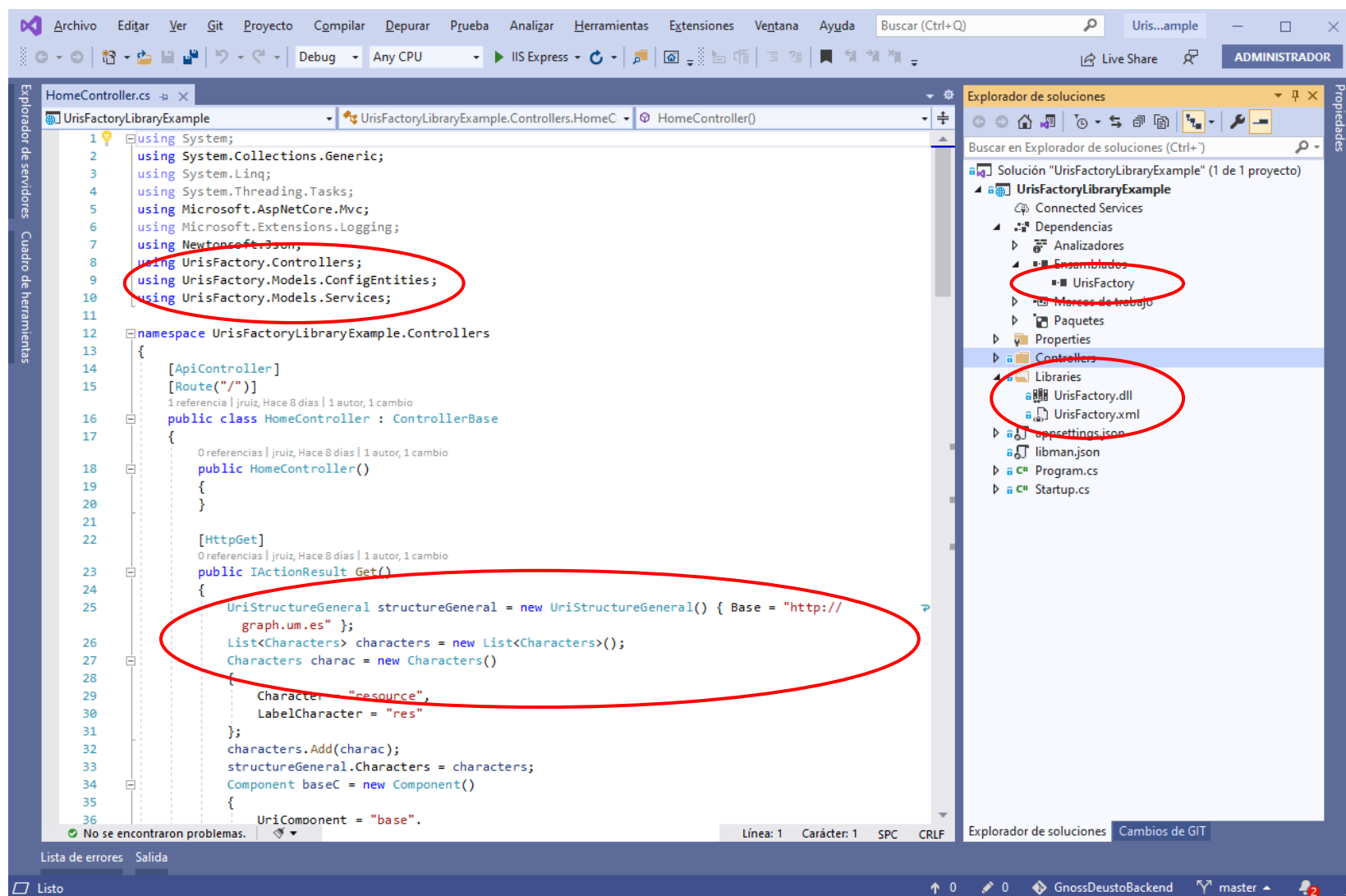
Como reutilizar los componentes ASIO como librerías

Desde Visual Studio o Visual Studio Code, haciendo referencia al proyecto o la librería. Hay tres formas posibles:

- Hacer referencia al código fuente: Descargar el proyecto desde GitHub y añadirlo en nuestra solución como un proyecto más
- Hacer referencia a la DLL: Descargar el paquete, compilarlo, copiar la DLL en nuestra aplicación y hacer una referencia a ella.
- Subir un paquete a nuget y hacer referencia a él, a través de nuget.org

FONDO EUROPEO DE DESARROLLO REGIONAL (FEDER)

Una manera de hacer Europa



The screenshot displays a Visual Studio IDE with a C# project named 'UriFactoryLibraryExample'. The main editor shows the 'HomeController.cs' file, which includes several using statements and a class definition. The 'UriFactory' namespace is highlighted in the code. The solution explorer on the right shows the project structure, with 'UriFactory' and 'UriFactory.dll' highlighted. The status bar at the bottom indicates 'No se encontraron problemas.' (No problems found).

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Threading.Tasks;
5 using Microsoft.AspNetCore.Mvc;
6 using Microsoft.Extensions.Logging;
7 using Newtonsoft.Json;
8 using UriFactory.Controllers;
9 using UriFactory.Models.ConfigEntities;
10 using UriFactory.Models.Services;
11
12 namespace UriFactoryLibraryExample.Controllers
13 {
14     [ApiController]
15     [Route("/")]
16     public class HomeController : ControllerBase
17     {
18         public HomeController()
19         {
20         }
21
22         [HttpGet]
23         public IActionResult Get()
24         {
25             UriStructureGeneral structureGeneral = new UriStructureGeneral() { Base = "http://graph.um.es" };
26             List<Characters> characters = new List<Characters>();
27             Characters charac = new Characters()
28             {
29                 Character = "resource",
30                 LabelCharacter = "res"
31             };
32             characters.Add(charac);
33             structureGeneral.Characters = characters;
34             Component baseC = new Component()
35             {
36                 UriComponent = "base".
```

FONDO EUROPEO DE DESARROLLO REGIONAL (FEDER)

Una manera de hacer Europa

¡Gracias!



HERCULES

