XV Jornadas de Ingeniería Telemática.
JITEL 2021.
Universidad de A Coruña.

Actas de las XV Jornadas de Ingeniería Telemática (JITEL 2021), A Coruña (España), 27-29 de octubre de 2021.

GirolA: Una pasarela de servicios web

Antonio Delgado, Antonio Estepa.
Departamento de Ingeniería Telemática,
Universidad de Sevilla
C/ Camino de los Descubrimientos s/n
{aldelgado,aestepa}@us.es.

La integración entre múltiples aplicaciones con varios proveedores de servicios web puede resultar un problema complejo en organizaciones de mediano y gran tamaño donde el parque de aplicaciones y sistemas destino con los que es necesario integrarse se multiplica. Además, la variabilidad de mensajes y tecnologías implicadas en este tipo de entornos no hacen más que incrementar la complejidad de la situación.

En este artículo se presenta GirolA, una pasarela de servicios web diseñada con el objetivo de facilitar la integración de aplicaciones en entornos como el mencionado y que lleva siendo empleada con éxito durante más de un año. En dicho periodo se han integrado 19 servicios y se han realizado más de 2.7M de invocaciones a servicios web por lo que lo consideramos un caso de éxito.

***Palabras Clave*- pasarela de servicios, integración**

I. INTRODUCCIÓN

La integración de los sistemas de información es una de las necesidades básicas en las grandes organizaciones. Hoy es habitual que una aplicación se comunique con otras para consultar su información o solicitarles que realicen determinadas acciones. Afortunadamente, disponemos de tecnologías fuertemente asentadas que proporcionan los pilares para desarrollar las soluciones de integración que pudieran ser necesarias: SOAP, REST, UDDI, WSDL, etc. se han convertido en estándares de facto para esta labor [1, 2]. Sin embargo, satisfacer las necesidades de integración en una organización de gran tamaño, en especial del sector público, puede llegar a ser muy complejo. En casos extremos, un organismo puede manejar cientos de aplicaciones y cada una de ellas puede necesitar integrarse con varios sistemas destino que, a su vez, pueden emplear distintas tecnologías de integración obligando a nuestras aplicaciones a contemplar cada una de ellas. Incluso cuando todos los sistemas destino con los que tenga que interactuar una de nuestras aplicaciones empleen una misma tecnología de servicios web existirán distintos mensajes a intercambiar con ellos, con una gran variabilidad en su estructura y sintaxis. Todo ello conlleva un gran esfuerzo de desarrollo que debe mantenerse prácticamente en cada sistema de la organización. Por último, tampoco es extraño que en organizaciones de gran tamaño exista un buen número de aplicaciones heredadas con un mantenimiento muy complicado o incluso imposible de llevarse a cabo en algunos casos.

En este contexto podemos tomar algunas medidas para reducir la presión sobre los equipos de desarrollo. En algunos casos se puede optar por atacar la problemática de la integración desarrollando bibliotecas que abstraigan a nuestras aplicaciones de la tecnología y mensajes concretos a intercambiar con el sistema destino. Es una buena solución, pero no carente de algunos inconvenientes. En primer lugar, estas bibliotecas deben mantenerse actualizadas lo que deriva en evolutivos que deberán ser implementados en nuestras aplicaciones. Por ejemplo, si cierto campo que era opcional en una llamada pasa a ser obligatorio, la biblioteca deberá ser actualizada y también las aplicaciones que hagan uso de esa llamada. De hecho, no hace falta ir a un cambio funcional para ilustrar este problema. Cualquier nueva compilación de la biblioteca obligará a recompilar todas las aplicaciones que dependan de ella. Por lo tanto, las bibliotecas, aunque amortiguan el esfuerzo necesario para mantener nuestras aplicaciones integradas con los sistemas destino, no lo eliminan por completo. Desplazan el esfuerzo del desarrollo a un único punto, pero se mantiene un esfuerzo de desarrollo, aunque sea mínimo, en nuestras aplicaciones. En segundo lugar, no es infrecuente que existan aplicaciones en las que no se pueda hacer uso de las bibliotecas para resolver sus necesidades de integración debido a diferencias de versiones, uso de tecnologías no compatibles u otros problemas [3]. Por último, hay que destacar que esta solución será prácticamente imposible de aplicar en sistemas heredados ya que en muchos casos carecerán de un mantenimiento adecuado.

En este artículo se propone una solución alternativa para solventar algunos de los problemas comentados con anterioridad, GirolA, una pasarela o *gateway* de servicios

This work is licensed under a Creative Commons 4.0 International License (CC BY-NC-ND 4.0)

web [4]. Un sistema de este tipo permite desacoplar a las aplicaciones de las bibliotecas de integración. Si además el *gateway* admite la integración con los sistemas destino a través mecanismos alternativos (como el uso de ficheros) puede permitir resolver los problemas de integración de las aplicaciones heredadas. Además, veremos que podemos dotar a una plataforma de este tipo de muchas funcionalidades interesantes que pueden elevar el nivel de integración de nuestra organización de manera importante.

En las siguientes secciones de este artículo se describe de forma general el sistema desarrollado para continuar profundizando en su conocimiento a través del modelo de dominio, el de presentación (interfaz de usuario), el modelo de procesos y su interfaz de programación de aplicaciones (API). Una vez presentado el sistema se relata nuestra experiencia práctica con él durante más de un año de uso. Por último, cerrando el documento, se plantean algunas conclusiones obtenidas de nuestra experiencia particular.

II. Estado del arte

Un *gateway* o pasarela de servicios web es un artefacto software que actúa de intermediario en la comunicación entre un consumidor y un proveedor de servicios web. Las pasarelas de servicios web proporcionan un único punto de control, acceso y validación de solicitudes de servicios web, y permiten controlar qué servicios están disponibles para los distintos consumidores de los servicios web [5].

En la literatura podemos encontrar *gateways* de servicios web con distintos propósitos:

- Traducción de protocolos de servicios web o tecnologías de distribución de procesamiento. Probablemente la forma más sencilla de convertir un servicio SOAP a un servicio REST o viceversa sea usar una pasarela de servicios web (en estos casos también conocidos como un proxy de servicio). Estos sistemas realizan las traducciones entre los mundos SOAP y REST de las peticiones y las repuestas pertinentes. Un producto que realiza esta función es *Membrane Service Proxy* [6]. También existen publicaciones en las que se realizan propuestas en esta línea [7]. En algunos casos incluso se han propuesto sistemas más complejos que actúan como pasarela entre tecnologías distintas para procesamiento distribuido [8].
- Seguridad. La eclosión de los servicios web ha incrementado la exposición de recursos críticos generando un mayor esfuerzo para la securización de los sistemas que puede ser mitigado con el uso de *gateways* para servicios web [9]. Otras propuestas relativas a la seguridad se han centrado en evitar ataques. Por ejemplo, en [10] se presenta *Checkway*, un *gateway* para proteger servicios web de ataques de denegación de servicio. Por último, en [11] se propone el uso de una pasarela que controla el acceso a un nivel muy fino en las invocaciones a los servicios web.
- Un tercer grupo de aportaciones está orientado a resolver el problema de la integración con múltiples orígenes de datos manteniendo una consistencia en la información obtenida con el objeto de realizar posteriores análisis o estudios. En esta línea *TogoWS* [12] ofrece una solución en el dominio específico de la biotecnología.

Como hemos visto, la idea de emplear pasarelas de servicios web no es nueva y su uso ha sido propuesto con fines diversos a lo largo del tiempo. Sin embargo, no existen propuestas que aconsejen su empleo como solución al problema que ocasiona la integración en organizaciones con un alto número de aplicaciones. Tampoco hay estudios dirigidos al objetivo de reducir la carga de desarrollo que implica el que una aplicación tenga que integrarse con distintos sistemas cada uno con tecnologías y mensajes diferentes. Tampoco hay propuestas enfocadas en tratar de ofrecer una solución en este sentido para aplicaciones heredadas.

III. Descripción general del sistema

A grandes rasgos, nuestro objetivo es desarrollar un *gateway* de servicios web que simplifique la integración de las aplicaciones de una organización con los distintos sistemas destino proveedores de servicios web. Los principales requisitos que para nosotros debe cumplir el sistema se enumeran en la Tabla I.

Tabla I
Requisitos principales del sistema

Requisito	Descripción
1	El sistema debe permitir la integración con sistemas destino de distintas tecnologías (SOAP, REST, etc.), tipos (síncronos o asíncronos) y que empleen distintos métodos para la autenticación, (usuario + clave, certificados, etc.)
2	La pasarela debe interactuar con las aplicaciones de la organización, pero también con usuarios de distinto perfil
3	Las peticiones recibidas por el *gateway* (también denominadas lotes o trabajos) estarán formadas por un grupo de solicitudes individuales. Una petición estará compuesta por lo tanto al menos por una solicitud individual
4	El tratamiento de las peticiones recibidas por la pasarela será asíncrono
5	Los trabajos o lotes podrán comunicarse a la pasarela por diversos medios: ficheros, servicio web, base de datos, etc.
6	El sistema no debe limitarse a ser un mero intermediario en las peticiones que se le realicen para integrarse con algún sistema externo. Debe tratar la información obtenida de esos sistemas externos simplificando la respuesta para las aplicaciones de la organización
7	La sintaxis de las peticiones y sus respuestas deben mantener una estructura homogénea y ser independientes del servicio final invocado
8	El sistema debe almacenar información sobre todas las transacciones realizadas de forma que puedan utilizarse para auditorías u otros usos

This work is licensed under a Creative Commons 4.0 International License (CC BY-NC-ND 4.0)

En la figura 1, se muestra un diagrama de contexto del sistema. En esta figura se puede apreciar que las aplicaciones de la organización (APP1..APP3) disponen de tres vías para realizar peticiones de integración con sistemas remotos: a través de ficheros CSV empleando un servidor SFTP, empleando una API de servicios web o utilizando un conjunto de funciones definidas en la base de datos del sistema (requisitos 2 y 5). También vemos que los usuarios pueden interactuar con el sistema mediante una interfaz de usuario que les permite generar trabajos utilizando también ficheros en formato CSV (requisitos 2 y 5). La pasarela, es capaz de interactuar con sistemas externos de distinta naturaleza (requisito 1). En la figura aparecen GIRO (el sistema de Gestión Integral de Recursos Organizativos de la Junta de Andalucía), SCSP (Supresión de Certificados en Soporte Papel) [13] y otros como podrían ser los servicios web de la AEAT [14], pero la plataforma permite la integración sencilla con cualquier proveedor de servicios web. En el diagrama también se quiere poner de relieve que el *gateway* no es un mero "intermediario" de peticiones si no que tiene cierta capacidad de proceso y que es capaz de tratar las respuestas recibidas, traduciéndolas o simplificándolas para el solicitante y siendo capaz de generar hasta tres salidas diferentes para un lote de peticiones (requisito 6). Por último, también se quiere destacar que el sistema almacena en su base de datos información de todas las transacciones realizadas, así como trazas (log) de las peticiones efectuadas y las respuestas obtenidas de los sistemas destino (requisito 8).

De manera simplificada podemos decir que GirolA recibe lotes de trabajos por distintos medios y de distintos solicitantes: aplicaciones o humanos. Esos trabajos son encolados y se van ejecutando efectuando las invocaciones a los servicios web pertinentes en cada caso. La pasarela se encarga de componer las llamadas a los servicios web en la forma adecuada, autenticarse en el sistema destino en la manera en que sea necesario y obtener la respuesta del mismo ya sea de manera síncrona o asíncrona. Una vez obtenida la respuesta el sistema la almacena y tiene la capacidad de tratarla para ofrecer a los solicitantes una información más refinada. La respuesta puede ser tratada de tres formas distintas teniendo así la posibilidad de generar hasta tres resultados diferentes para un trabajo concreto.

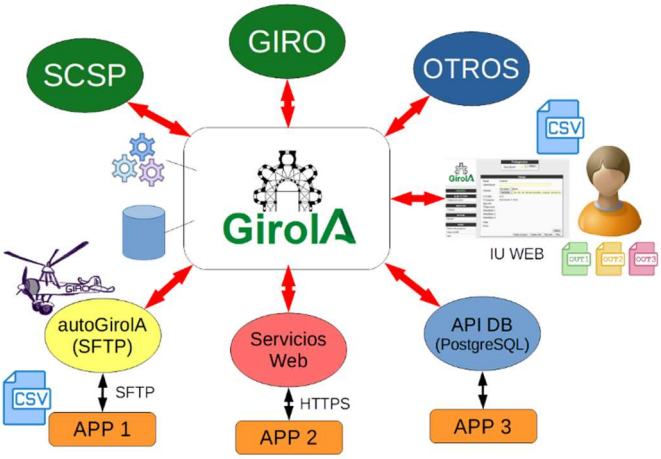

Fig. 1. Diagrama de contexto del sistema

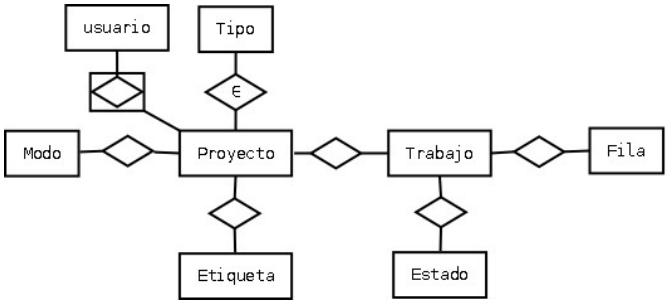

Fig. 2. Diagrama Entidad-Relación

IV. Modelo de dominio

El modelo de dominio de nuestro sistema (Fig. 2) recoge principalmente información sobre proyectos de integración, los trabajos y su contenido. A continuación, se describen las entidades más relevantes del modelo.

Proyecto. Recoge los datos de lo que en GirolA se denomina un proyecto de integración. Un proyecto de integración define una forma determinada de interactuar con alguno de los servicios web soportados por GirolA. Los atributos principales de esta entidad son el servicio con el que se desea trabajar, el entorno al que atacará el proyecto (pruebas, preproducción, producción), usuario y clave de acceso, etc. También son atributos de esta entidad una serie de ficheros que definen cómo se ha de generar la llamada al servicio web en cuestión y el tratamiento que se desea realizar de la respuesta obtenida. De todos ellos podemos destacar los siguientes:

- `callh`, `callb` y `callt`: cabecera, cuerpo y cola de la llamada. Se trata de ficheros de texto que actúan a modo de plantilla para componer el XML o JSON que se empleará para realizar la llamada al servicio web.
- `dbproc`, `proc1`, `proc2` y `proc3`: especifican una transformación xslt o un filtro jq [15] para realizar tratamientos de la respuesta obtenida. El primero de estos ficheros es obligatorio y se encarga de extraer de la respuesta la información necesaria para alimentar la base de datos de GirolA. Los otros tres son opcionales y posibilitan especificar hasta tres tratamientos alternativos de la respuesta para generar distintos ficheros de salida.

Tipo. Los proyectos de integración se clasifican según su tipo. Esta entidad sólo dispone de una clave primaria secuencial y una descripción.

Trabajo. Asociados a un proyecto de integración se pueden generar infinidad de trabajos. Un trabajo tiene una descripción que debe ser única, un usuario creador, un usuario ejecutor, fechas de creación y ejecución, etc.

Estado. Cada trabajo tiene asociado un estado que tiene relación con la ejecución del mismo. Así un trabajo

This work is licensed under a Creative Commons 4.0 International License (CC BY-NC-ND 4.0)

puede estar pendiente de ejecución, pendiente de respuesta asíncrona, finalizado con éxito, finalizado con error, etc.

Fila. Esta entidad define cada uno de datos individuales que componen un trabajo o lote. Para cada fila se pueden informar hasta 20 columnas que recogen datos que pueden ser empleados tanto en la invocación del servicio web como en la generación de resultados.

Etiqueta. Una etiqueta sirve para definir un comportamiento en cuanto a la ejecución y planificación de los trabajos asociados a un proyecto e incluso si fuera necesario establecer prioridades distintas en el tratamiento de los mismos. Los trabajos no etiquetados siguen la política general: se encolan por orden de llegada y posteriormente se van extrayendo de la cola y se ejecutan en paralelo hasta un máximo de seis procesos. Una etiqueta puede determinar el uso de colas particulares o planificaciones distintas para la ejecución ciertos trabajos. Actualmente están definidas en el sistema las siguientes etiquetas:

- `1_POR_MINUTO`: Esta etiqueta define una cola en la que los trabajos se extraen cada minuto y se ejecutan de uno en uno. Está destinada trabajos que emplean servicios web muy pesados en los que una invocación puede llegar a ocupar 1.5MB
- `3_EN_PARALELO`: Esta etiqueta define una cola en la que los trabajos son ejecutados como máximo en grupos de tres.
- `NOCTURNO`: Define una planificación similar a la que se emplea por defecto pero en la que la ejecución de los trabajos se permite únicamente entre las 00:00 y las 06:00.

Modo. El modo define el comportamiento del proyecto de integración. Actualmente están definidos los siguientes modos:

- LITIS: Llamada Individual con Tratamiento Individual (servicio destino Síncrono)
- LCTIS: Llamada Colectiva con Tratamiento Individual (servicio destino Síncrono)
- LCTCS: Llamada Colectiva con Tratamiento Común (servicio destino Síncrono)
- LCTUS: Llamada Colectiva con Tratamiento Único (servicio destino Síncrono)
- LCTIA: Llamada Colectiva con Tratamiento Individual (servicio destino Asíncrono)
- LCTCA: Llamada Colectiva con Tratamiento Común (servicio destino Asíncrono)
- LCTUA: Llamada Colectiva con Tratamiento Único (servicio destino Asíncrono)

La llamada individual (LI) indica que por cada fila del trabajo o lote se debe realizar una invocación al servicio web especificado en el proyecto de integración. Por el contrario, una llamada colectiva (LC) implica que hay que componer una única invocación que contenga información sobre cada una de las filas que componen el lote.

Respecto a los tratamientos de la respuesta proporcionada por el servicio web se distinguen tres tipos:

1. Tratamiento individual (TI): la respuesta obtenida se trata individualmente para cada fila que compone el lote.
2. Tratamiento común (TC): la respuesta obtenida es común para todas las filas que componen el lote y por lo tanto los valores respuesta para cada fila del lote son idénticos
3. Tratamiento único (TU): la respuesta obtenida es común para todas las filas del lote, sin embargo, no se va a generar un resultado individualizado, el resultado será único para el lote.

El último carácter indica el modo de ejecución del servicio web destino que puede ser síncrono (S) o asíncrono (A).

V. Modelo de presentación

La interfaz de usuario del sistema ha sido desarrollada empleando WAINE [16], un entorno de desarrollo de interfaces de usuario basado en modelos. En ella, se definen cuatro roles: administrador, usuario avanzado, usuario regular y consulta. En la Tabla II se especifican las unidades de interacción accesibles a cada rol.

En los siguientes apartados se describen las unidades de interacción de mayor relevancia (destacadas en negrita en la Tabla II)

A. Gestión de proyectos de integración

Esta interfaz de usuario (Fig. 3) permite a los administradores gestionar proyectos de integración, así como asignar la visibilidad sobre el proyecto a los distintos usuarios (zona inferior de la unidad de interacción). En la pestaña información se cumplimentan los campos principales de la entidad. En las pestañas *Ficheros Obligatorios*, *Ficheros Opcionales*, *Procesamiento Extra 1*, *Procesamiento Extra 2* y *Procesamiento Asíncrono* se adjuntan los ficheros que fueran necesarios para definir las invocaciones a los servicios web y especificar los distintos tratamientos de la respuesta.

Tabla II
Rol-funcionalidad

	Admin	U.avanzado	Usuario	Consulta
Gest. Proyectos de integración	✓			
Gest. Categorías	✓			
Gest. Etiquetas	✓			
Gest. Trabajos	✓	✓	✓	
Trabajos en curso	✓	✓	✓	
Cons. Últimos trabajos realizados	✓	✓		
Cons. Trabajos por proyecto	✓	✓		
Búsqueda avanzada	✓	✓		
Búsqueda resultado	✓	✓		
Estad. Trabajos por categoría	✓			
Estad. Trabajos anuales	✓			
Gestión de usuarios	✓			
Búsqueda de resultados reducida			✓	✓

This work is licensed under a Creative Commons 4.0 International License (CC BY-NC-ND 4.0)

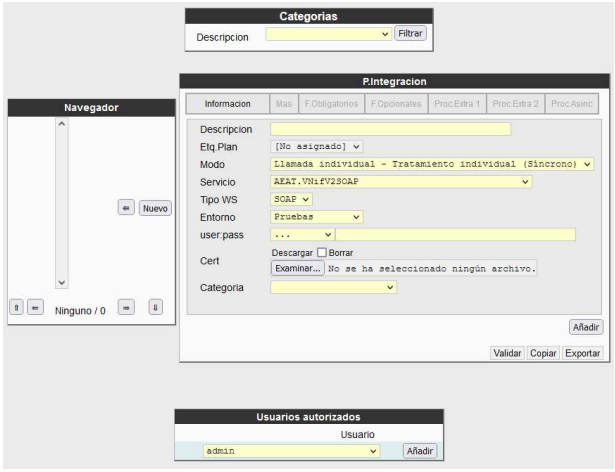

Fig. 3. Gestión de proyectos de integración

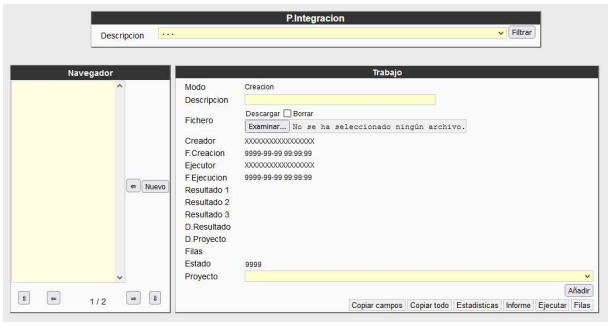

Fig. 4. Gestión de trabajos

B. *Gestión de trabajos*

Esta unidad de interacción (Fig. 4) permite el alta, modificación y eliminación de trabajos sobre los proyectos de integración a los que el usuario tiene acceso. También permite realizar diversas acciones como copiar el trabajo, extraer estadísticas de los resultados, generar un informe sobre la ejecución del mismo, ejecutar el trabajo y por último ver el detalle de las filas que lo componen.

C. *Trabajos en curso*

Presenta para cada trabajo en ejecución una serie de campos que dan una idea de su velocidad de ejecución y su evolución esperada (ver Fig.5). Entre ellos aparecen el usuario ejecutor, la fecha de inicio, el número de peticiones por minuto (ppm), la duración estimada del trabajo (de) y la hora estimada de fin (hef). También permite ver el detalle de las filas que componen el trabajo, obtener estadísticas del mismo y acceder a la carpeta donde se almacenan las peticiones y respuestas de cada invocación al proveedor de servicios web realizada en el proyecto.

Fig. 5 Trabajos en curso

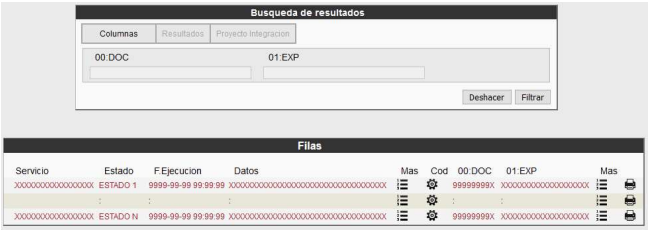

Fig. 6. Búsqueda de resultados reducida

D. *Búsqueda de resultados reducida*

Para finalizar esta sección presentamos la interfaz de usuario "búsqueda de resultados reducida" (Fig. 6). Esta unidad de interacción permite localizar resultados en función de la información que se aportó para la invocación al servicio.

VI. PROCESOS

En esta sección se presentan los procesos más relevantes del sistema. En la figura 7 podemos ver un diagrama que ilustra los distintos procesos y las dependencias existentes entre ellos, así como las relaciones que guardan con otros artefactos. A continuación, se describe brevemente cada proceso.

girola.action: Es la acción lanzada por el botón Ejecutar de la interfaz de usuario gestión de trabajos (ver Fig. 3). Este proceso toma el fichero CSV asociado al trabajo y lanza una ejecución manual del mismo invocando al proceso girola.man

girola.auto: Este proceso da soporte a lo que se ha denominado autogirola. Realiza un proceso similar al que realiza girola.action, salvo que el trabajo manual es lanzado por la creación de una carpeta en un directorio determinado que contiene una relación de ficheros que deben ser procesados por la pasarela.

girola.man: Se encarga de la ejecución de trabajos que vienen definidos por un fichero CSV. Este proceso da de alta el trabajo con sus correspondientes filas en la base de datos del sistema e inmediatamente invoca a girola.exec para que realice la ejecución del mismo.

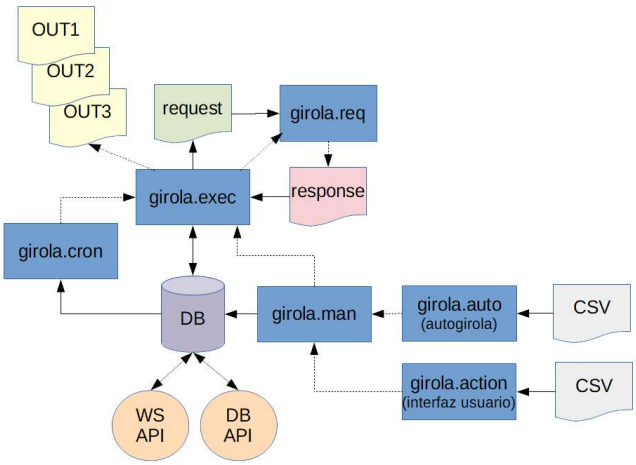

Fig. 7. Representación de los procesos

This work is licensed under a Creative Commons 4.0 International License (CC BY-NC-ND 4.0)

girola.cron: Realiza una tarea similar a la de girola.man pero para aquellos trabajos que han sido generados a través de la API de la base de datos o de la API de servicios web de GirolA. Empleando este proceso en combinación con las etiquetas de los proyectos de integración (ver sección IV) se pueden definir distintas planificaciones de ejecución para los trabajos. El funcionamiento básico de girola.cron consiste en tomar el trabajo más antiguo aún no ejecutado para una etiqueta dada y lanzar su ejecución invocando a girola.exec según una planificación establecida en el fichero crontab [17] e identificada por la etiqueta asignada al proyecto.

girola.exec: Es el núcleo de la ejecución de procesos. Este proceso toma cada fila del trabajo a ejecutar, monta las llamadas SOAP o REST, realiza la llamada al servicio web pertinente invocando a girola.req, trata las respuestas para generar los ficheros de salida necesarios y almacena los resultados en la base de datos de GirolA. Durante el proceso genera un log del trabajo y almacena evidencias de la ejecución en el sistema de ficheros.

girola.req: es el encargado de realizar llamadas a los distintos servicios web, abstrayendo a otros procesos de asuntos propios de la llamada como pueden ser el *host* involucrado, el *endpoint* a utilizar, la autenticación o firma de la llamada si fuera necesaria, etc.

VII. INTERFAZ DE PROGRAMACIÓN DE APLICACIONES

No se podría finalizar esta exposición del sistema sin presentar el conjunto de funciones que permiten a las aplicaciones de la organización integrarse con GirolA. De manera abstracta la API tanto de servicios web (actualmente SOAP) como la soportada por la base de datos (ver Fig. 1) está compuesta por las siguientes funciones:

`1. ID addLDRow(IN cuser,IN proj,IN lddescr,IN col00,IN,..,IN col19)`. Añade una fila con las columnas proporcionadas (col00 .. col19) a un lote o carga identificado por lddescr. En la llamada se informan también el usuario o aplicación que realiza la inserción (cuser) y el proyecto de integración que se debe emplear (proj). Durante la primera inserción de una fila en el lote, se crea el mismo. Se pueden seguir añadiendo filas al lote mientras no se le haya asignado tipo al lote utilizando la función `setLoadType`. Si la función tuvo éxito devuelve el identificador de la fila insertada. A continuación, se muestra un ejemplo de la invocación a esta función empleando el API en base de datos y el API SOAP.

API en base de datos

Llamada

```
select addLDRow('AplicacionX', 65, 'LOTE-1',
'99999999K', 'ES0123456789012345678901');
```

Respuesta

```
addldrow
----------
         6
```

API SOAP

Llamada

```
<soapenv:Envelope xmlns:soapenv=
"http://schemas.xmlsoap.org/soap/envelope"
xmlns:girola="http://juntadeandalucia.es/GIROLA">
<soapenv:Header/>
   <soapenv:Body>
      <ADDLDROWREQ>
         <HEAD>
            <APP>AplicacionX</APP>
         </HEAD>
        <PARAMS>
         <PROJ>65</PROJ>
         <DESCR>LOTE-1</DESCR>
         <COLS>
           <ITEM>99999999K</ITEM>
           <ITEM>ES0123456789012345678901</ITEM>
         </COLS>
        </PARAMS>
      </ADDLDROWREQ>
   </soapenv:Body>
</soapenv:Envelope>
```

Respuesta

```
<?xml version="1.0"?>
<SOAP:Envelope xmlns:SOAP=
    "http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP:Header/>
   <SOAP:Body>
     <ADDLDROWRESP>
     <HEAD>
      <ERROR>000</ERROR>
      <ERRDESCR>Ejecucion sin errores</ERRDESCR>
     </HEAD>
     <RESULT>
       <ROWID>6</ROWID>
     </RESULT>
     </ADDLDROWRESP>
   </SOAP:Body>
</SOAP:Envelope>
```

`2. ID delLDRow(IN cuser, IN id)`. Elimina una fila de un lote de trabajo. La llamada recibe el usuario o aplicación que realiza la inserción (cuser) y el identificador de fila a eliminar (id). Esta eliminación sólo se puede realizar cuando el trabajo no ha sido ejecutado aún. Si tuvo éxito devuelve el identificador de la fila eliminada.

`3. getLDRowResult(IN cuser,IN id,OUT err,OUT stat,OUT ts,OUT data)`. Obtiene el conjunto de resultados básicos para la fila identificada por el identificador id.

`4. getLDRowFullResult(IN cuser,IN id,OUT err,OUT stat,OUT ts,OUT data,OUT r00, .. ,OUT r19)`. Obtiene todos los resultados para una fila, los básicos (los mismos que se obtienen con una llamada a `getLDRowResult`) y otros que pueden haber sido definidos por el proyecto de integración. El proyecto de integración puede definir hasta 20 resultados para una llamada.

`5. Boolean isLoadFinished(IN cuser, IN d)`. Dado un usuario creador y un lote la función devuelve verdadero si el trabajo está finalizado y falso en caso contrario.

`6. ID setLoadType(IN cuser, IN d, IN t)`. Establece el tipo de trabajo para un lote determinado, M: Manual, A: Automático. Una vez asignado no se pueden añadir más filas al trabajo. Si el estado se asigna a automático el sistema lanzará el trabajo de forma atendiendo a la planificación establecida.

This work is licensed under a Creative Commons 4.0 International License (CC BY-NC-ND 4.0)

Tabla III
LÍNEAS DE CÓDIGO

Fichero	Descripción	Líneas
Girola.asl	Código XML con la especificación de los modelos de usuario, diálogo y presentación	1805
Girola.sql	Código SQL correspondiente al modelo de dominio	624
Girola.action	Script correspondiente al proceso girola.action	63
Girola.auto	Script correspondiente al proceso girola.auto	84
Girola.cron	Script correspondiente al proceso girola.cron	54
Girola.exec	Script correspondiente al proceso girola.exec	719
Girola.man	Script correspondiente al proceso girola.man	61
Girola.req	Script correspondiente al proceso girola.req	114
	Total	3524

`7. syncLDRowResult(IN cuser,IN proj,IN lddescr,IN col00,..,IN col19,OUT err,OUT stat,OUT ts,OUT data,OUT r00,.., OUT r19)`. Realiza una llamada síncrona para una única fila con las columnas proporcionadas y devuelve los resultados. Esta función proporciona la única manera de realizar un trabajo síncrono en GirolA. Sólo está disponible en el API de servicios web (SOAP).

El API de la pasarela define un conjunto de mensajes que mantienen una estructura homogénea y que son independientes del servicio final invocado como se indicaba en el requisito 7 de la Tabla I. De hecho, en la mayoría de casos una aplicación sólo necesita emplear las funciones 1, 3, 5 y 6. Este conjunto mínimo es suficiente para dar de alta trabajos y obtener sus resultados básicos.

VIII. EXPERIENCIA CON GIROLA

GirolA comenzó a ser construido en abril de 2020 y actualmente su desarrollo está prácticamente finalizado. El proyecto tiene un total de 3524 líneas de código distribuidas como se muestra en la Tabla III.

Las distintas funcionalidades soportadas se han ido incorporando al sistema atendiendo a las necesidades más prioritarias de la organización. En paralelo a su desarrollo se han ido definiendo diversos proyectos de integración y ejecutando trabajos sobre ellos. Actualmente hay definidos en el sistema 115 proyectos de integración sobre los que se han ejecutado unos 5.628 trabajos que contenían 2.784.732 filas (1.720.755 en 2020 y 1.063.977 hasta el momento de escribir este artículo en junio de 2.021). En la figura 8 se muestra un diagrama en el que se detalla la distribución de los trabajos y el número de filas a lo largo del tiempo.

Tabla IV
TOP 5 SERVICIOS WEB MÁS UTILIZADOS

Servicio web	Trabajos	Peticiones	T. medio
Validación de cuenta	895	1136388	1270
Consulta de deudas	446	985686	2210
Alta de beneficiario	335	302139	901
Alta de subvención	2130	195631	92
Validar subvención	1676	144548	86

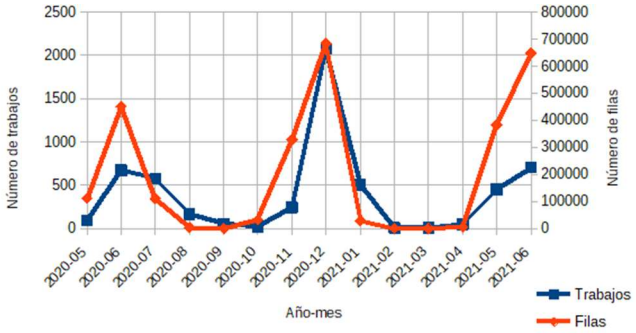

Fig. 8. Trabajos realizados y filas involucradas

GirolA soporta hasta el momento la integración con 19 servicios de todo tipo: síncronos, asíncronos, SOAP, REST, etc. El top 5 de los servicios web más empleados se detalla en la Tabla IV. En esta tabla se muestra el número de trabajos para cada servicio web, el número de filas que afectadas en cada uno de ellos y el tamaño medio de los trabajos. Se aprecia cómo los dos últimos tienen un tamaño medio de lote mucho menor. Ello es debido a la naturaleza de estos servicios. Los servicios de alta y validación de subvenciones son de tipo LCTCS y LCTUS respectivamente (ver sección IV). Se trata por lo tanto de servicios de llamada colectiva que en estos casos estaban limitados a unos 100 integrantes por llamada aproximadamente.

El número de usuarios por perfil existentes y activos en la aplicación se detallan en la Tabla V. Como se puede apreciar a la fecha de la redacción de este artículo el número de usuarios avanzados se ha reducido notablemente. Sólo quedan activos 3. Esto es debido a que a lo largo del ejercicio 2020 mientras los sistemas se integraban con nuestra pasarela muchos de los trabajos se realizaron manualmente y por lo tanto se requería personal para gestionarlos. A lo largo de 2021 las integraciones de varias aplicaciones con GirolA han ido finalizando y sólo unos pocos trabajos necesitan de intervención manual (relacionados con sistemas heredados).

Hasta la fecha se han integrado totalmente con este gateway 3 aplicaciones de la organización:

1. Incentiva: Sistema de gestión de subvenciones y ayudas (Groovy/Grails 1.0.5) Se ha integrado con la pasarela empleando la API de la base de datos.
2. SGTL: Sistema de gestión de residencias de tiempo libre (Oracle 10g + Java). Integrado con GirolA mediante la API de la base de datos.
3. ERTE210: Sistema de gestión de las ayuda ERTE (Oracle APEX). Integrado via servicios web. Integrado con GirolA empleando la API SOAP. Este sistema ha necesitado 20 horas de desarrollo para la integración con nuestro *gateway*.

Tabla V
USUARIOS REGISTRADOS / USUARIOS ACTIVOS

Perfil	Usu. registrados	Usu. activos
Administrador	4	4
U. Avanzado	22	3
Usuario	1	1
Consulta	2	2

This work is licensed under a Creative Commons 4.0 International License (CC BY-NC-ND 4.0)

Desde un punto de vista cuantitativo la aplicación de este sistema en nuestra organización ha sido un éxito. En apenas un año se han realizado invocaciones a distintos servicios web para 2.784.732 elementos. Se han integrado 19 servicios web distintos que permiten principalmente realizar consultas sobre determinados asuntos, pero también permiten solicitar a los sistemas remotos que elaboren algún trabajo o realicen algunas acciones. Hasta el momento se han integrado 3 aplicaciones con esta pasarela, pero la idea a futuro es que se convierta en un sistema de uso obligatorio por las aplicaciones de la organización.

Desde un punto de vista cualitativo la integración de las aplicaciones con sistemas externos se ha simplificado en enorme medida. Una aplicación antes tenía que integrarse con N sistemas destino. Ahora una aplicación tiene que integrarse con un único sistema: GirolA. Los mensajes que las aplicaciones intercambian con la pasarela son homogéneos. Todos los servicios web de los sistemas destino son tratados de una misma manera cuando se emplea nuestro *gateway*. Hemos podido comprobar que todo ello redunda en una reducción importante del esfuerzo necesario para desarrollar la integración de una aplicación con sistemas externos. Por último, damos soporte a la Integración de aplicaciones heredadas a través del uso de ficheros generados a través de procesos externos a las mismas.

IX. Conclusiones y trabajos futuros

En este artículo hemos presentado GirolA, una pasarela de servicios web cuyos objetivos principales son desacoplar a las aplicaciones de las bibliotecas de integración, simplificar la integración de las mismas con los proveedores de servicios web y resolver el problema de la integración de las aplicaciones heredadas. La experiencia de uso de la pasarela refleja que se han conseguido los objetivos, simplificando significativamente la integración de servicios en una gran organización.

Nuestros planes de futuro con respecto a GirolA pasan por seguir extendiendo su uso en las distintas aplicaciones de la organización. Hasta el momento se ha realizado un esfuerzo en esta línea pero focalizado sólo en algunos sistemas. También deseamos continuar ampliando el número de servicios web soportados por la pasarela y dotarla de una API REST.

Referencias

[1] Singh, Munindar P., and Michael N. Huhns. *Service-oriented computing*. Chichester: Wiley, 2005.

[2] Rashed A. Bahlool, and Ahmed M. Zeki. "Comparative Study between Web Services Technologies: REST and WSDL." *2019 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, p. 1-4, IEEE, 2019.

[3] Tanabe, Yudai, Tomoyuki Aotani, and Hidehiko Masuhara. "A context-oriented programming approach to dependency hell." *Proceedings of the 10th International Workshop on Context-Oriented Programming: Advanced Modularity for Run-time Composition*, pp. 8-14, 2018.

[4] Huy, Hoang Pham, Takahiro Kawamura, and Tetsuo Hasegawa. "Web service gateway-A step forward to e-business." Proceedings. IEEE International Conference on Web Services, 2004, p. 648-655 IEEE, 2004.

[5] "WebSphere Application Server for z/OS web services gateway", https://www.ibm.com/docs/es/was-zos/9.0.5?topic=gateway-web-services-frequently-asked-questions#cwsg_faq__i2

[6] Membrane Service Proxy, https://www.membrane-soa.org

[7] Jan Königsberger and Bernhard Mitschang. "R2SMA-A Middleware Architecture to Access Legacy Enterprise Web Services using Lightweight REST APIs." ICEIS (2), p. 704-711, 2018.

[8] Bixin, Fan Yu Yang Fan Liu, and Zhou Bin. "Research and Implementation of a SOAP-CORBA Gateway System [J]." Computer Engineering and Applications, vol. 29, 2004.

[9] Gerald Brose. "A gateway to web services security–Securing SOAP with proxies." International Conference on Web Services, p. 101-108, Springer, Berlin, Heidelberg, 2003.

[10] Nils Gruschka, and Norbert Luttenberger. "Protecting web services from dos attacks by soap message validation." IFIP International Information Security Conference, p. 171-182, Springer, Boston, MA, 2006.

[11] Ernesto Damiani, et al. "Fine grained access control for SOAP e-services." Proceedings of the 10th international conference on World Wide Web. p. 504-513, 2001.

[12] Katayama Toshiaki, Mitsuteru Nakao, and Toshihisa Takagi. "TogoWS: integrated SOAP and REST APIs for interoperable bioinformatics Web services." Nucleic acids research vol. 38, no suppl_2, p. W706-W711, 2010

[13] SCSP, Sustitución de Certificados en Soporte Papel, https://administracionelectronica.gob.es/ctt/scsp

[14] https://www.agenciatributaria.es/static_files/Sede/Procedimiento_ayuda/ZA05/indice_serv_web_sum_aapp.pdf

[15] jq a lightweight and flexible command-line JSON processor, https://stedolan.github.io/jq

[16] Antonio Delgado, et al. "Reusing UI elements with model-based user interface development." International Journal of Human-Computer Studies, vol. 86, p. 48-62, 2016

[17] Shashank Sharma and Thomas Keir. "Scheduling Tasks." Beginning Fedora: From Novice to Professional, p. 427-431, 2007

This work is licensed under a Creative Commons 4.0 International License (CC BY-NC-ND 4.0)