Junta de Andalucía

Cofinanciado por la Unión Europea

GOBIERNO DE ESPAÑA

Fondos Europeos

Junta de Andalucía

TeJA – Manual de extensión de desarrollos

Versión: v01r04

Fecha: 13/03/2026

HOJA DE CONTROL

Proyecto	TeJA – Manual de extensión de desarrollos		
Documento	Manual de extensión de desarrollos		
Nombre del fichero	TeJA - Manual_extension_desarrollos_v01r04.docx		
Autor	UTE		
Versión/Revisión	v01r05	**Fecha Versión**	13/03/2026
Aprobado por		**Fecha Aprobación**	

REGISTRO DE CAMBIOS

Versión	Causa del cambio	Responsable del cambio	Área	Fecha del cambio
v01r00	Elaboración inicial del documento	UTE	Sv. Informática	22/05/2025
v01r01	Se añaden los ejemplos de petición para variables, condiciones y acciones externas	UTE	Sv. Informática	16/06/2025
V01r02	Información Interfaz Numerador Actualización información variables y condiciones externas Añadir información integraciones SCSP	UTE	Sv. Informática	22/09/2025
V01r03	Añadir información nueva versión de trewa-api-adaptador	UTE	Sv. Informática	02/12/2025
V01r04	Actualización de la versión de trewa-api	UTE	Sv. Informática	21/01/2026
V01r05	Actualización información acceso escritorio tramitación	UTE	Sv. Informática	09/03/2026

CONTROL DE DISTRIBUCIÓN

Nombre y Apellidos	Cargo	Área
Manuel Escobar Montes	Subdirector	Servicios Digitales y Analítica de Datos
José Ignacio Cortés Santos	Jefe de Servicio	Administración Digital
José Andrés García Romero de la Osa	Técnico	Gabinete Estratégico
Almudena López Maraver	Directora de Proyecto	PMO
Ricardo Pierre-Jean de Puelles	Gestor Servicio Senior	UTE

ÍNDICE

1 Introducción

El Tramitador de Expedientes TeJA está diseñado bajo una arquitectura modular en la que es posible incorporar módulos funcionales cuyo objetivo sea implementar funcionalidades específicas de tramitación relacionadas con el negocio de un procedimiento. Dichos módulos funcionales deberán de desarrollarse de manera externa y desacoplada del propio Tramitador de Expedientes.

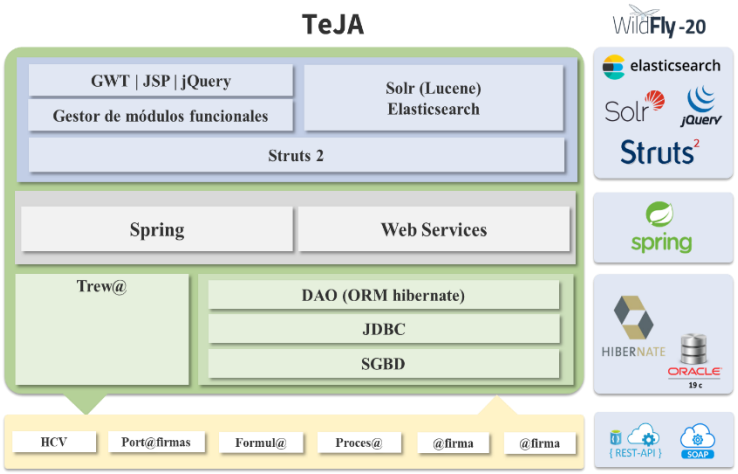
El objetivo del presente documento es detallar los pasos a seguir para el diseño y construcción de módulos funcionales y servicios de manera desacoplada y externa a TeJA.

2 TeJA. Contextualización y capacidad de extensión

2.1 ¿Qué es TeJA? Definición y arquitectura actual

El tramitador de expedientes (TeJA) permite la tramitación electrónica de cualquier familia de procedimientos, sirviendo como punto de partida y como software de base permitiendo una capacidad de extensión para abordar los desarrollos verticales y particulares de cada implantación o negocio, minimizándose las labores de programación necesarias para implantar una solución de tramitación electrónica de procedimientos administrativos.

A modo de contextualización previa, la arquitectura tecnológica de desarrollo utilizada por TeJA es la mostrada a continuación:

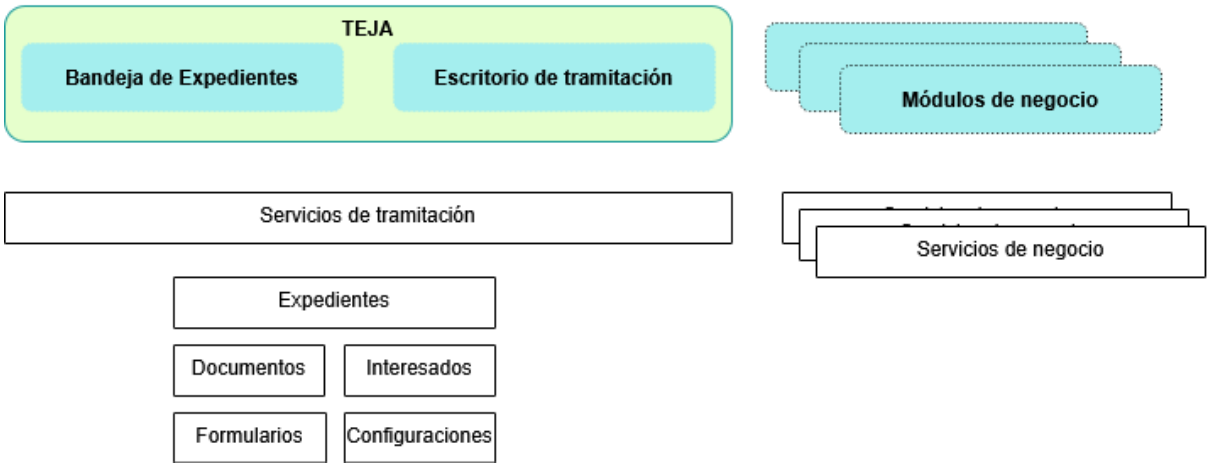


- Máquina virtual **Java versión 14**
- Capa de controlador y servicios de negocio **Spring 5.2.8**
- Capa de acceso a datos **hibernate 5.4.8.**
- Capa de visualización **Struts 2.0.14**
- Integración con el motor de indexación **elasticsearch y Solr**
- Despliegue con el Servidor de Aplicaciones **WildFly 20.0.1Final**
- Acceso a base de datos con esquema compatible **Trew@ 2.6.14.0**

Como evolución de los sistemas que dan cobertura a las necesidades de tramitación, se está llevando a cabo una redefinición de su arquitectura y el diseño global, con el fin de orientarlos hacia una solución de tipo “SaaS” (Software as a Service).

Se expone a continuación un diagrama que refleja la dirección hacia la que se está realizando esta evolución, a fin de contextualizar a los equipos colaboradores y que puedan tomar decisiones de desarrollo en consecuencia.

El tramitador TeJA será una capa frontal (Desarrollada en Angular) que se compondrá de una bandeja de expedientes y el escritorio de tramitación. Por otra parte, existirá una capa de servicios core de tramitación que darán cobertura a las necesidades transversales, como la gestión de expedientes, o la obtención de su configuración. Esta capa estará disponible para su uso en los módulos de negocio desarrollados, que a su vez podrán disponer de sus propios servicios de negocio.



2.2 ¿Qué es un módulo de TeJA? Tipos de módulos permitidos

El sistema TeJA está diseñado de manera modular de forma que se permite la configuración de nuevos módulos funcionales que incrementen la funcionalidad ofrecida de manera general para la tramitación de expedientes.

Tipos de módulos funcionales. Por su naturaleza funcional, existen dos tipos de componentes funcionales o módulos:

- Componentes funcionales de tramitación, globales a cualquier familia de procedimientos.
- Componentes funcionales de negocio, específicos de un procedimiento o familia de procedimientos.

Debido a su forma de visualización y objeto, existen cuatro tipos de módulos en TeJA:

- **Pestaña (desacoplado)**: Módulos cuya funcionalidad es accesible a través de una sección o pestaña del escritorio de tramitación. *Ej: listado de datos de negocio, resumen de datos presentados, …*
- **Utilidades (desacoplado)**: Módulos invocados desde el propio escritorio de tramitación, ubicados como utilidades, y que dan soporte a la tramitación de los expedientes. *Ej: Datos de negocio expediente, utilidad de baremación, etc.*
- **Administración (internos):** Módulos propios del tramitador cuya funcionalidad es accesible desde la herramienta de administración, con objeto de realizar tareas de administración, parametrización o mantenimiento.
- **Web Service (internos):** Módulos propios del tramitador cuya funcionalidad es incluir el servicio web implementado en el archivo de descripción de web service que define el catálogo de servicios de TeJA. Este tipo de módulo se encuentra en desuso y se recomienda implementar cualquier servicio web de manera externa a TeJA.

Para añadir nuevas funcionalidades específicas del negocio relacionadas con la tramitación de expedientes, TeJA se ha diseñado de forma que se permita la **capacidad de extensión mediante:**

- **Configuración de módulos externos.** Dichos módulos se deberán realizar de manera externa y desacoplada integrándose en el propio escritorio de tramitación permitiendo así visualizar de manera unificada la tramitación del expediente junto con sus posibles datos o módulos específicos del negocio de cada procedimiento.
- **Configuración de servicios realizados de manera externa.** Dichos servicios se deberán realizar de manera externa y desacoplada permitiendo cubrir las funcionalidades para la resolución de variables, condiciones, acciones, numeradores y nuevos servicios SCSP.

El objetivo de los siguientes apartados del presente documento será detallar las especificaciones y pautas a seguir para la construcción de dichos módulos funcionales y servicios externos que se quieran integrar sobre el Tramitador de Expedientes TeJA.

2.3 Capacidad de extensión de TeJA mediante módulos externos.

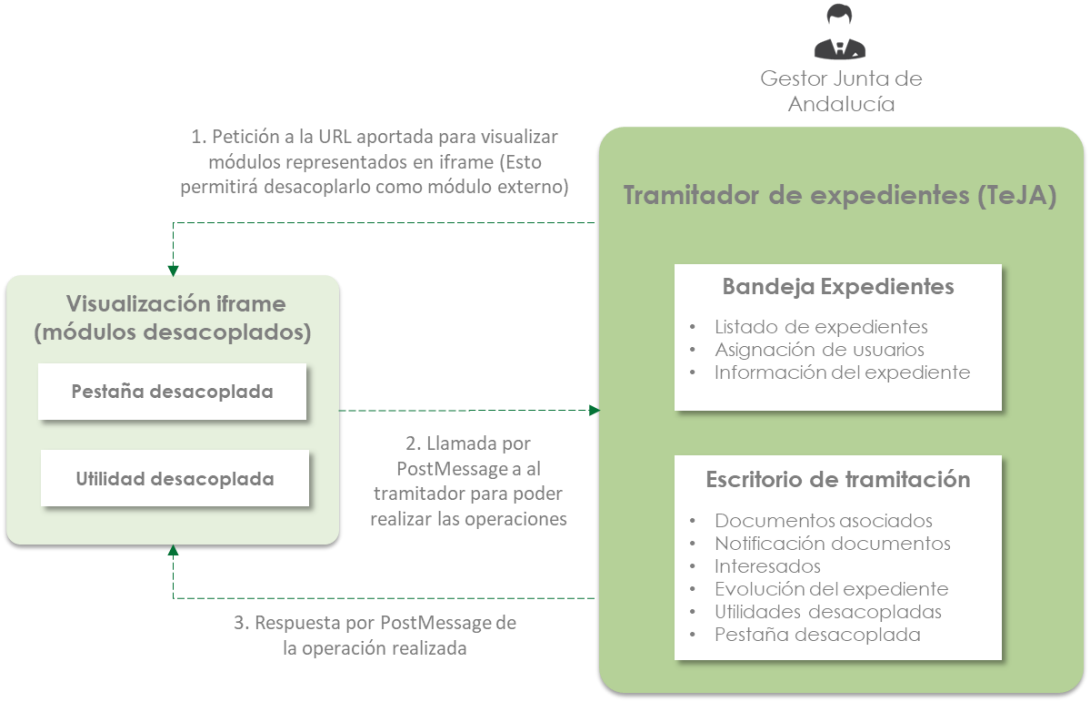
En este apartado se detallarán cuáles son los tipos módulos para los cuales se permitirá realizar un desarrollo funcional propio.

Únicamente se permitirá el **desarrollo de los módulos de tipo pestaña (desacoplado) o utilidad (desacoplado).** Dichos desarrollos deberán de realizarse de manera externa y desacoplada al Tramitador de expedientes siguiendo las pautas que se describirán en el apartado "*3 Guía de desarrollo para módulos funcionales*" del presente documento. Estas pautas serán de obligado cumplimiento para lograr la construcción de un módulo compatible con TeJA y podrán ser completadas con otras recomendaciones relacionadas con normas de codificación y diseño de patrones.

A modo de contextualización inicial, este tipo de módulos se deberán dar configurar en la administración del Tramitador como módulo de tipo externo o desacoplado, y será necesario indicar la ruta URL de despliegue de dicho módulo (Ej: https://sevidor/contexto_despliegue_modulo) para que TeJA pueda realizar la invocación a dicha URL.

Dichos módulos se mostrarán en el Tramitador de Expedientes en un marco (iframe) en el que se visualizará la información de estos. Una vez abierto dicho iframe toda la funcionalidad quedará delegada en el propio módulo externo por lo que se permitirá implementar cualquier tipo de funcionalidad o desarrollo.

Adicionalmente, se permitirá que dichos módulos externos se comuniquen con el Tramitador de Expedientes para poder realizar un conjunto de operaciones definidas. Por ejemplo: obtener datos del contexto de TeJA, recargar algunas partes del tramitador (interesados, transiciones, etc.), abrir y cerrar ventanas modales, etc. Dicha comunicación se deberá realizar mediante intercambio de mensajes (mediante PostMessage), siguiendo el siguiente diagrama general de comunicaciones:



En el apartado "*3 Guía de desarrollo para módulos funcionales*" se ampliará con más detalle todo lo necesario para el desarrollo y comunicación de este tipo de módulos externos.

2.4 Capacidad de extensión de TeJA mediante servicios externos

En este apartado se detallarán cuáles son las funcionalidades que permite extender TeJA mediante servicio externos.

- **Servicios SCSP** (externo): servicios de consulta de datos para servicios SCSP.
- **Variables** (externas): servicios para la resolución de variables utilizadas en las plantillas de los documentos generados.
- **Condiciones y acciones** (externas): servicios para la evaluación de condiciones y ejecución de acciones configurables desde la administración de Trewa..

- **Numeradores** (externas): servicios para el calculo personalizado del número de un expediente al realizar el alta del mismo.

A modo de contextualización inicial, este tipo de servicios externos se deberán dar configurar en la administración del Trewa de forma similar a la actual configuración, pero indicando la URL del servicio REST externo (en lugar de la clase y método Java). En el apartado "*0*

Guía de desarrollo para servicios externos" se ampliará con más detalle todo lo necesario para el desarrollo y comunicación de este tipo de servicios externos

2.5 Requisitos previos

Para el desarrollo de módulos funcionales (de manera externa y desacoplada) se debe disponer de acceso una instancia de TeJA donde poder configurarlos y replicar la llamada que realiza el Tramitador de Expedientes a dicho módulo externo.

Para solicitar el acceso a una instancia de TeJA en un entorno de integración deberá realizarlo abriendo una petición en NAOS: https://naossuite.juntadeandalucia.es/autogestion

3 Guía de desarrollo para módulos funcionales

3.1 Arquitectura base de los módulos de administración digital

Como se ha descrito en el apartado anterior, el tramitador de expedientes TeJA está diseñado de manera que permite extender la funcionalidad de tramitación mediante el desarrollo de módulos funcionales (pestañas y utilidades) implantados de manera externa al tramitador. Dichos módulos deberán **desarrollarse y desplegarse en servidores o contenedores independientes a TeJA.**

No existen restricciones a nivel de arquitectura y tecnologías con la que implementar dichos módulos externos, sin embargo, se recomienda que tanto el diseño como la implementación se realicen cumpliendo las directrices marcadas por la Oficina de Arquitectura de la ADA. Para más información puede acceder al portal del desarrollador en el siguiente enlace: https://desarrollo.juntadeandalucia.es/recursos/buscador?f%5B0%5D=tipo_recurso%3A685

En relación con la arquitectura, para la implantación de la capa frontend se recomienda la utilización de “Angular” con “TypeScript” como framework principal, aprovechando su diseño modular, robustez y amplia comunidad de soporte. La librería Material Angular complementará el desarrollo, ofreciendo componentes visuales modernos basados en “Material Design” que garantizan interfaces consistentes, accesibles y responsivas. La versión más reciente (Angular 17) introduce mejoras significativas en rendimiento, experiencia de desarrollo y compatibilidad con tecnologías actuales, consolidándose como una solución ideal para aplicaciones dinámicas y escalables. Bajo estas tecnologías se han desarrollado un conjunto de componentes web que pueden reutilizarse a la hora de abordar los desarrollos consiguiendo de esta manera un doble objetivo:

- ✓ **Facilitar la implementación** de módulo externos, al disponer de un conjunto de componentes web ya desarrollados.
- ✓ **Unificar los estilos y aspectos visuales** de todos los módulos externos desarrollados, al implementarse en base a los mismos componentes y ficheros de estilos (*.scss y *.css).

Este enfoque asegura aplicaciones coherentes y adaptables a las necesidades específicas del ecosistema de módulos de administración digital (no siendo estas las mismas necesidades que las de los portales web gobernados bajo el modelo de servicios digitales de la Junta de Andalucía), optimizando tanto el desarrollo como la experiencia del usuario final.

En caso de no utilizar dicha tecnología para la implementación de los módulos, no podrán reaprovecharse estos componentes ya desarrollados. No obstante, **las pautas y normativas de estilos si deberán ser de obligado cumplimiento** para conseguir unificar la experiencia de usuario al navegar por las diferentes funcionalidades de TeJA y los módulos desarrollados de manera externa, de manera que el usuario tramitador perciba una uniformidad en el diseño y apariencia durante el uso del tramitador.

Por otro lado, se propone adoptar una arquitectura basada en microfrontends, que permite dividir las aplicaciones en módulos frontend independientes. Este enfoque facilita el desarrollo y mantenimiento, ya que cada módulo puede diseñarse, probarse y desplegarse de manera autónoma, incluso con tecnologías distintas si es necesario. Los microfrontends se integran a través de APIs bien definidas, permitiendo una comunicación fluida y una implementación cohesionada. Esto resulta especialmente útil en sistemas complejos, como los de la Administración Pública, donde se requiere flexibilidad para incorporar nuevas funcionalidades sin comprometer la estabilidad del sistema central.

A continuación, se detallarán las pautas de estilos que se deberán cumplir y el conjunto de componentes web existentes. También se detallará la forma en la que se comunicarán los módulos externos y TeJA.

3.2 Definición de políticas de estilos

Actualmente, se está trabajando en la creación de una metodología de trabajo que simplifique tanto el desarrollo de nuevos proyectos como la adaptación de los proyectos existentes al nuevo modelo de soluciones de administración digital de la Junta de Andalucía. Esta **metodología incluye el diseño de una política de estilos común** para todas las aplicaciones, que permita estandarizar la apariencia y el comportamiento de las interfaces de usuario.

La **política de estilos** ofrecerá una colección predefinida de variables que podrán ser fácilmente utilizadas en cualquier proyecto. Estas **variables incluyen los colores oficiales y otros elementos de diseño** definidos en el modelo de servicios digitales y especializado para el desarrollo de soluciones de administración digital. Su uso permite establecer una estructura uniforme que garantiza la coherencia visual y funcional entre diferentes aplicaciones. Al implementar o actualizar un proyecto, simplemente será necesario importar esta política de estilos para disponer de un conjunto completo de variables base adaptadas al estándar, facilitando su integración y aplicación en los diferentes componentes del sistema.

Además, dado que muchos elementos clave en las interfaces de usuario estarán representados por **componentes web reutilizables**, esta política también incluirá una definición detallada de los estilos asociados a cada uno de estos componentes web. Los estilos estarán parametrizados, permitiendo a los desarrolladores personalizar aspectos concretos según las necesidades específicas del proyecto, pero siempre respetando los colores y directrices marcadas en la política de estilos.

Para los **elementos únicos o específicos** que no estén cubiertos por los componentes web, se proporcionará una guía de estilos con opciones de desarrollo recomendadas que cumplan con los principios establecidos en el modelo. Esto garantizará que todos los elementos y componentes, tanto reutilizables como propios, se mantengan homogéneos y alineados con el resto del ecosistema digital de la Junta de Andalucía.

3.2.1 Uso de la política de estilos en los proyectos web

Como se mencionó previamente, los componentes web dispondrán de una serie de estilos predefinidos que no podrán ser modificados. Adicionalmente, también se proporcionará una guía de estilos que permitirá personalizar el diseño de los proyectos de manera coherente. Esta guía incluirá una serie de variables que deben utilizarse en los desarrollos específicos de cada proyecto para asegurar la homogeneización visual en toda la plataforma.

Dicha guía de estilos se publicará a través de una librería en **npm**, lo que permitirá integrarla de forma sencilla en los proyectos Angular. Para importar estos estilos predefinidos, será necesario realizar los siguientes pasos:

1. **Actualizar el package.json**: se debe añadir la dependencia correspondiente a la librería de estilos al archivo “package.json” del proyecto. Por ejemplo:

   ```
   "dependencies": {
     "@jdastyle/styles": "1.0.0"
   }
   ```

2. **Instalar la dependencia**: ejecutar el siguiente comando para instalar la librería:

   npm install

3. **Añadir la referencia a los estilos en angular.json**: Una vez instalada la librería, se deben importar los estilos predefinidos en el proyecto editando el archivo angular.json e incluyendo la ruta al archivo principal de estilos en la sección styles. Por ejemplo:

```
"styles": [
  "node_modules/@jdastyle/styles/main.scss",
  "src/styles.scss"
```

Con estos pasos, los estilos predefinidos y la guía de estilos quedarán disponibles en el proyecto. A partir de este punto, se podrá empezar a utilizar las variables incluidas en la guía, las cuales abarcan diferentes aspectos del diseño, como colores, tipografía, espaciado, z-index, bordes y sombras. Al utilizar estas variables, se garantiza que los nuevos desarrollos sigan la misma línea estética definida en la guía.

3.2.2 Variables definidas en la política de estilos

- **Colores**
  - $color-verde-principal: #087021;
  - $color-verde-principal-hover: #0B4C1A;
  - $color-verde-apoyo: #0B4C1A;
  - $color-verde-apoyo-especifico: #F7FBF8;
  - $color-fondo-blanco: #FFFFFF;
  - $color-gris-principal: #111111;
  - $color-gris-apoyo: #BEBEBE;
  - $color-gris-contenedor: #EEEEEE;
  - $color-gris-apoyo-especifico: #F5F5F5;
  - $color-estado-exito: #2ABC67;
  - $color-estado-informacion: #5BC0DE;
  - $color-estado-error: #D4403A;
  - $color-estado-advertencia: #F0AD4E;

- **Tipografía**
  - $font-primary: 'Source Sans Pro', sans-serif;
  - $font-secondary: 'Montserrat';
  - $font-size-base: 16px; (Tamaño base para calcular rem)
  - $line-height-base: 1.5; (Altura de línea estándar)

- **Tamaños de fuentes para encabezados**
  - $h1-font-size: 2.5rem; (Tamaño para h1)
  - $h2-font-size: 2rem; (Tamaño para h2)
  - $h3-font-size: 1.75rem; (Tamaño para h3)
  - $h4-font-size: 1.5rem; (Tamaño para h4)
  - $h5-font-size: 1.25rem; (Tamaño para h5)
  - $h6-font-size: 1rem; (Tamaño para h6)

- **Pesos de fuentes (font-weight)**
  - $h1-font-weight: 700; (Peso de fuente para h1)
  - $h2-font-weight: 600; (Peso de fuente para h2)
  - $h3-font-weight: 500; (Peso de fuente para h3)
  - $h4-font-weight: 400; (Peso de fuente para h4)
  - $h5-font-weight: 400; (Peso de fuente para h5)
  - $h6-font-weight: 400; (Peso de fuente para h6)

- **Escala de tamaños de fuente**
  - $font-size-sm: 0.875rem; (Tamaño pequeño)
  - $font-size-md: 1rem; (Tamaño mediano, base)
  - $font-size-lg: 1.25rem; (Tamaño grande)
  - $font-size-xl: 1.5rem; (Tamaño extragrande)

- **Espaciado**
  - $spacing-xs: 4px;
  - $spacing-sm: 8px;
  - $spacing-md: 16px;
  - $spacing-lg: 24px;
  - $spacing-xl: 32px;

- **Z-Index**
  - $z-index-modal: 1050;
  - $z-index-dropdown: 1000;
  - $z-index-header: 900;
  - $z-index-footer: 800;

- **Bordes y sombras**
  - $border-radius-sm: 4px;
  - $border-radius-sm-md: 6px;
  - $border-radius-md: 8px;
  - $border-radius-md-lg: 12px;
  - $border-radius-lg: 16px;
  - $box-shadow-sm: 0 1px 3px rgba(0, 0, 0, 0.12);
  - $box-shadow-md: 0 4px 6px rgba(0, 0, 0, 0.16);
  - $box-shadow-lg: 0 10px 20px rgba(0, 0, 0, 0.2);

3.3 Detalle de componentes web

Como se mencionó anteriormente, se está trabajando en la creación de una amplia colección de componentes web que cubran la mayoría de los elementos habituales en las aplicaciones y páginas web. El objetivo es facilitar el desarrollo, garantizar la homogeneidad visual y funcional de las aplicaciones, y alinearlas con los estándares definidos en el Modelo de Servicios Digitales (MSD) orientado a los portales web de la Junta de Andalucía.

Además, estos componentes y directrices se están definiendo específicamente para su uso en módulos de administración digital, asegurando así que las aplicaciones destinadas a la gestión administrativa cumplan con los requisitos de usabilidad, accesibilidad y eficiencia establecidos en el modelo.

3.3.1 Listado de componentes

A continuación, se describen brevemente los componentes ya disponibles y su funcionalidad:

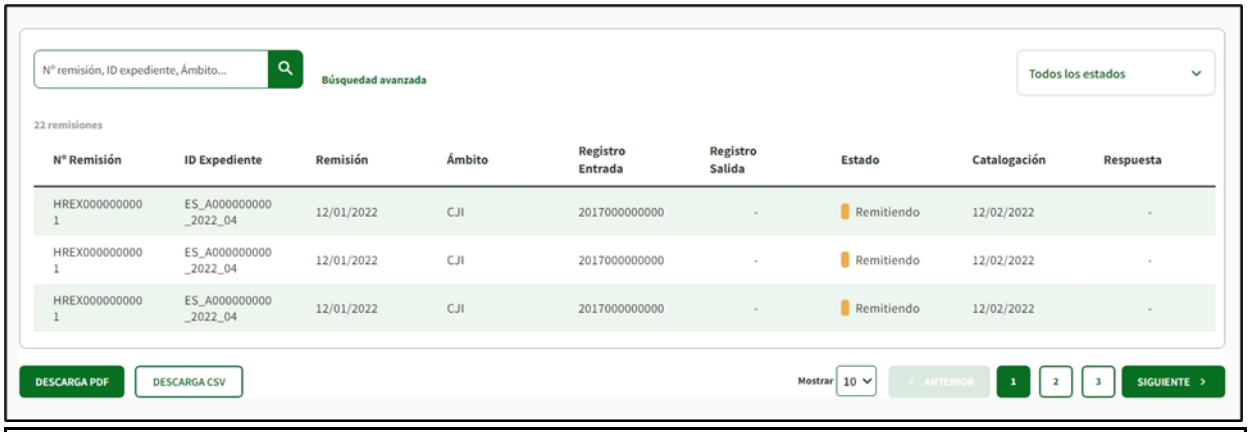
- **jda-button**: Componente de botón completamente configurable que permite personalizar su apariencia, comportamiento y estilos según las necesidades específicas de la aplicación.


```
<jda-button

        [texto]="'Enviar'"
        [tipo]="'principal'"
        [identificador]="'btn-enviar'"
        [iconoIzq]="'icono-izquierdo'"
        [iconoDer]="'icono-derecho'"
        [loading]="false"
        [disabled]="false"
        (evento)="onButtonClick($event)"

></jda-button>
```

- **jda-tabla**: Componente de tabla configurable que facilita la personalización de su apariencia y funcionalidades. Permite seleccionar entre diferentes tipos de entradas y opciones, ya adaptadas al MSD, para ajustarse a diversos requisitos.




```
<jda-tabla

  [items]="[
    {
      'numRemision': 'HREX0000000001',
      'idExpediente': 'ES_A000000000_2022_04',
      'remision': '12/01/2022',
      'ambito': 'CJI',
      'regEntrada': '2017000000000',
      'regSalida': '-',
      'estado': 'Remitiendo',
      'catalogacion': '12/02/2022',
```

```
    'respuesta': '-'
   },
   {
    'numRemision': 'HREX0000000001',
    'idExpediente': 'ES_A000000000_2022_04',
    'remision': '12/01/2022',
    'ambito': 'CJI',
    'regEntrada': '2017000000000',
    'regSalida': '-',
    'estado': 'Remitiendo',
    'catalogacion': '12/02/2022',
    'respuesta': '-'
   }
  ]"
 [parametrosListado]="{
  'columnas': {
    'columns': [
      'numRemision',
      'idExpediente',
      'remision',
      'ambito',
      'regEntrada',
      'regSalida',
      'estado',
      'catalogacion',
      'respuesta'
    ],
    'columnsName': [
      'Nº Remisión',
      'ID Expediente',
      'Remisión',
      'Ámbito',
      'Registro Entrada',
      'Registro Salida',
      'Estado',
      'Catalogación',
      'Respuesta'
    ]
  },
  'acciones': []
 }"
 [filtroPrincipal]="{
  'nombreFiltro': 'Filtro libre',
  'identificadorFiltro': 'LIBRE',
  'tipo': 'text',
  'placeholder': 'Nº remisión, ID expediente, Ámbito...',
  'opciones': null
 }"
 [filtroSecundario]="{
  'nombreFiltro': 'Todos los estados',
  'identificadorFiltro': 'TODOS-ESTADOS',
  'tipo': 'select',
  'placeholder': null,
```

```
    'opciones': [{
        'descripcion': 'Remitiendo',
        'id': 'REMITIENDO',
        'color': '#F0AD4E'
      },{
        'descripcion': 'Fallida',
        'id': 'FALLIDA',
        'color': '#FF5F5F'
      }
    ]
  }"
  [filtrosAvanzado]="[
    {
      'nombreFiltro': 'Id. remisión',
      'identificadorFiltro': 'ID-REMISION',
      'tipo': 'text',
      'placeholder': null,
      'opciones': null
    },{
      'nombreFiltro': 'Estado',
      'identificadorFiltro': 'ESTADO',
      'tipo': 'select',
      'placeholder': null,
      'opciones': [{
        'descripcion': 'Remitiendo',
        'id': 'REMITIENDO',
        'color': '#F0AD4E'
       },{
        'descripcion': 'Fallida',
        'id': 'FALLIDA',
        'color': '#FF5F5F'
       },{
        'descripcion': 'Nueva',
        'id': 'NUEVA',
        'color': '#5BC0DE'
       },{
        'descripcion': 'Finalizada',
        'id': 'FINALIZADA',
        'color': '#2ABC67'
      }]
    }
  ]";
  [numMostrar]="[10, 25, 50]"
  [totalItems]="22"
  [textoDescripcion]="22 remisiones"
  [isLoading]="false"
  [isErrorItems]=""
  (datosEnviados)="manejadorEventos($event)">
</jda-tabla>
```

- **jda-input**: Componente de campo de entrada configurable, diseñado para admitir diferentes tipos de inputs según la necesidad del proyecto. Su diseño y funcionalidad están completamente alineados con el MSD y cuentan con opciones flexibles para adaptarse a distintas exigencias.


```
<jda-input

 (inputEvent)="manejadorEventos($event)"
 [inputInput]="
  {
   'nombreFiltro': 'Total de remisiones',
   'identificadorFiltro': 'REG-ENTRADA',
   'tipo': 'text',
   'valor': 'Prueba de texto introducido',
   'delay': 2000,
   'placeholder': 'Escriba un registro de entrada válido',
   'requerido': false,
   'disabled': false,
   'validacion': '[A-Za-z]+',
   'info': {
     'titulo': 'Municipio',
     'contenido': 'Campo obligatorio: seleccione el municipio destino.'
   }
  }"
 style="width: 300px;"
></jda-input>
```

- **jda-breadcrumb**: Este componente, también conocido como "migas de pan", permite al usuario visualizar el historial de navegación de las pantallas por las que ha pasado, facilitando la orientación dentro de la aplicación.

- **jda-stepper**: Componente que ayuda a estructurar y visualizar flujos de trabajo divididos en pasos. Ideal para procesos que requieren una secuencia ordenada y coordinada de acciones.

- **jda-leyenda**: Diseñado para mostrar resúmenes o descripciones breves de diferentes datos, proporcionando una vista compacta y clara de información relevante según las necesidades del usuario.

- **jda-modal**: Componente para mostrar ventanas modales con contenido personalizable, útiles para destacar información importante o para interacciones específicas en las pantallas.

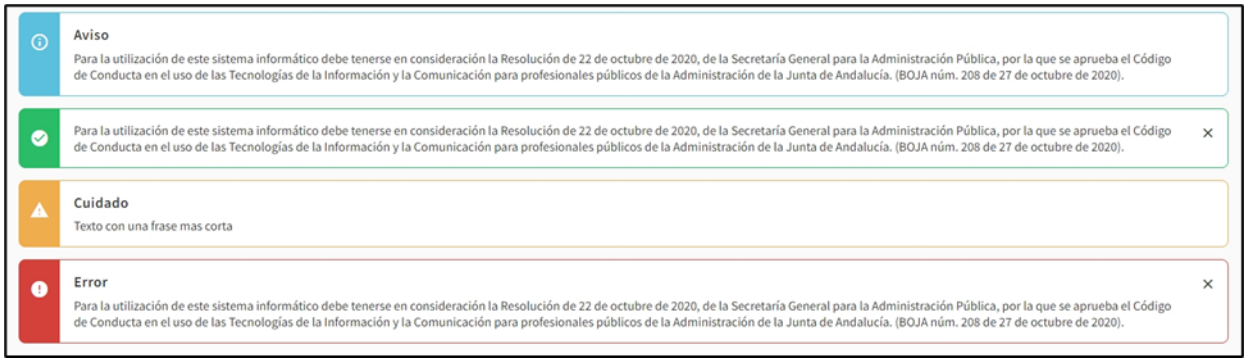
- **jda-footer**: Componente de pie de página diseñado según los estándares del Modelo de Servicios Digitales de la Junta de Andalucía, que garantiza una integración visual y funcional homogénea.

- **jda-header**: Componente de cabecera configurable que permite personalizar su apariencia y opciones, adaptándose a las necesidades del proyecto mientras mantiene el estilo definido por el MSD


```
<jda-header

 (evento)="manejadorEventos($event)"
 [headerInput]='{
  "nombre": "HERRAMIENTA DE REMISIÓN DE EXPEDIENTES",
  "logo": "assets/imgs/logo_junta.png",
  "descripcion": "Junta de Andalucia",
  "secciones": [
   {
    "nombre:" "Inicio",
    "identificador": "INICIO",
    "icono": "home",
    "imagen:" "null",
    "url": "inicio"
   },{
    "nombre": "Ayuda",
    "identificador": "AYUDA",
    "icono": "info",
    "imagen:" "null",
    "url": "ayuda"
   }, {
    "nombre": "Componentes",
    "identificador": "COMPONENTES",
    "icono": "build",
    "imagen": "null",
    "url": "componentes"
   }, {
    "nombre": "Ajustes",
    "identificador": "AJUSTES",
    "icono": "settings",
    "imagen": "null",
    "url": "ajustes"
   }, {
    "nombre": "Esteban",
    "identificador": "PERFIL",
    "icono": "person",
    "imagen": "png/fjnklsdfjios...",
    "url": "perfil"
   }
  ]
 }'
></jda-header>
```

- **jda-aviso**: Herramienta esencial para comunicar mensajes destacados al usuario, como avisos informativos, notificaciones de error o mensajes de éxito, con un diseño atractivo y alineado con el MSD.




```
<jda-aviso
  [titulo]="'Aviso'"
  [descripcion]="'Para la utilización de este sistema informático debe tenerse en consideración…'"
  [identificador]="'AVISO-MIPERFIL1'"
  [icono]="'info_outline'"
  [tipo]="'info'"
  [esFijo]="true"
  (evento)="manejadorEventos($event)"
></jda-aviso>
```

Estos componentes han sido diseñados para ser reutilizables y configurables, lo que no solo facilita el desarrollo de nuevas aplicaciones, sino que también simplifica la actualización de proyectos existentes al proporcionar una base estándar que cumple con los requisitos del MSD. Además, todos los componentes están diseñados para ofrecer opciones de personalización parametrizada, de manera que cada proyecto pueda ajustar los detalles necesarios sin perder la coherencia con el ecosistema digital general.

3.3.2 Importación de componentes

A continuación, se detallan los pasos para importar y usar los distintos componentes en un proyecto Angular:

1. **Instalar la dependencia del componente.** Antes de poder usar el componente, debes instalarlo en tu proyecto Angular a través del gestor de paquetes npm. Se debe ejecutar el siguiente comando en su terminal:

   npm install {{nombre del componente}}

2. **Añadir el componente al módulo correspondiente.** Una vez instalada la dependencia, se debe importar el componente en el módulo Angular donde se utilizará. Esto se hace editando el archivo del módulo (generalmente app.module.ts o un módulo específico). Se debe añadir el import correspondiente:

   import { JdaButtonComponent } from 'jda-button';

   Adicionalmente, se debe asegurar de incluirlo en la sección “declarations” del módulo si es un componente, o en “imports” si se trata de un módulo Angular completo.

3. **Incorporar el componente en una plantilla HTML.** Una vez que el componente está disponible en el módulo, se pude usar en las plantillas de tus componentes Angular (archivos *.html), Llamado al selector del componente y configurándolo con los parámetros que requiera.

3.4 Definición de la comunicación actual de los módulos externos con TeJA

3.4.1 Formato y tecnología para el intercambio de información y representación visual

Representación visual. Tal y como se ha mencionado en apartados anteriores, para incluir en el escritorio de tramitación funcionalidades adicionales implementadas en módulos externos (utilidad o pestaña desacoplada) TeJA utilizará elementos **iframe** de HTML. Dichos módulos deberán implementarse y desplegarse de manera independiente y externa a TeJA.

Tecnología intercambio de información. La comunicación entre los módulos externos y TeJA se realizará mediante intercambio de mensajes mediante el uso de "**PostMessage**" de JavaScript. Cuando desde TeJA se carga un módulo externo, TeJA establece en el atributo src del iframe la URL de acceso al módulo, a partir de este momento, el módulo puede necesitar información de TeJA para poder iniciarse e interactuar correctamente, por ejemplo: datos del expediente, datos del usuario tramitador, etc.

Por otra parte, También TeJA puede necesitar actualizar otros elementos del tramitador en función del negocio que se ejecuta en el módulo, por lo que se debe establecer un mecanismo seguro de comunicación entre TeJA y el módulo externo, este mecanismo se basa en el método "PostMessage" de JavaScript que permite publicar eventos de tipo "messaging" y en el método "addEventListener" de JavaScript que permite capturar un mensaje y llamar a una función cuando se recibe un evento.

Formato peticiones y respuestas. Para homogeneizar los mensajes de petición y respuesta intercambiados entre TeJA y los módulos, se define un conjunto de estructuras de datos en formato JSON con los siguientes campos:

- **Comunicación desde módulos hacia TeJA:**
  - **Request**: Todas las peticiones originadas por los módulos hacia TeJA deberán cumplir con la siguiente estructura:

```
var msg = {
        request: {
                id: id,
                operation: X,
        },
        payload: {
                params: { // La lista de params variará en función de la operación solicitada
                   ...
                }
        }
}
```

- **Response**: Todas las respuestas proporcionadas por TeJA a los módulos deberán cumplir con la siguiente estructura:

```
var msg = {
        response: {
                id: idReq,
                operation: X,
                version: 0
        },
        payload: {
                flag: 'ACK', // o 'NACK' si error
                data: '' // Solo se incluye en caso de ACK
                error: { // Solo se incluye en caso de NACK
                        code: 0, // Codigo de error
                        description: '' // Descripcion con el motivo del error
                }
        }
}
```

➢ **Comunicación desde TeJA hacia los módulos**:

Según sea la operación solicitada por el módulo, TeJA puede enviar al módulo una petición de bienvenida mediante la cual poder confirmar la carga completa del iframe.

- **Request**: Todas las peticiones de bienvenida originadas por TeJA hacia los módulos cumplen con la siguiente estructura:

```
var msg = {
        response: {
                ini: true
        },
        payload: {
                etiqueta: 'iframe' // Selector JQuery del iframe donde se carga el módulo
        }
}
```

- **Response**: Todas las respuestas proporcionadas por los módulos a TeJA deberán cumplir con la siguiente estructura:

```
var msg = {
        response: {
                iniResponse: true
        },
        payload: {
                flag: 'ACK',
                data: ''
        }
}
```

3.4.1.1 Primeros pasos. Suscripción y publicación de eventos

Suscripción a eventos. Lo primero que debemos hacer en un módulo externo es suscribirnos a los eventos “message” de tipo messaging. Con esto se consigue que el módulo externo este preparado para capturar los posibles mensajes que se envíen desde TeJA. A continuación, se muestra ejemplo de código para realizar dicha suscripción:

```
var win = null; // Variable que se inicializa dentro de la función winSource del modulo
                // al procesar el postMessage emitido por TeJA con ini:true
var etiq = null; // Variable que se inicializa dentro de la función winSource del modulo
                 // con la etiqueta recibida en el postMessage  emitido por TeJA con ini:true
var id = null; // Variable que se inicializa en las funciones del módulo al crear el mensaje postMessage que emitirá para TeJA

// Suscribir al evento messaging, winSource es la función JavaScript del módulo encargada de procesar el mensaje
window.addEventListener('message', winSource, false);

const winSource = (e) => {
        if (e.data) {
                var msg = JSON.parse(e.data);
                if (msg.response != undefined){
                        if (msg.response.ini == true) {
                                win = e.source;
                                etiq = msg.payload.etiqueta;
                                var msgResponse = {
                                        response: {
                                                iniResponse: true,
                                        },
                                        payload: {
                                                flag: 'ACK',
                                                data: ''
                                        }
                                }
                                // Publicacion de evento messaging para TeJA
                                win.postMessage(JSON.stringify(msgResponse),'*');
                        } else if (id == msg.response.id){
                                if (msg.payload.flag == 'ACK'){
                                        console.log('ACK desde iframe')
                                } else if (msg.payload.flag == 'NACK'){
                                        console.log('NACK desde iframe',msg.payload.error.description)
                                }
                        }
                } else if (msg.request != undefined){
                        if (msg.request.operation != undefined){
                                console.log(msg.request.operation);
                        }
                        if (msg.payload != undefined &&
                                msg.payload.params != undefined &&
                                msg.payload.params.data != undefined){
                                console.log(msg.payload.params.data);
                        }
                }
        }
}
```

Publicación de eventos. Con esto se permitirá el envío de PostMessage desde el módulo externo hacia TeJA para poder realizar operaciones. A continuación se muestra un ejemplo de envío para la operación 0 (abrir modal).

```
function sendModal(titulo, url, tamanio, recarga) {
    if (win != null) {
        try {
            id = Date.now();
            var msg = {
                'request': {
                    'id': id,
                    'operation': 0,
                },
                'payload': {
                    'params': {
                        'etiqueta': etiq,
                        'titulo': titulo,
                        'url': url,
                        'tamanio': tamanio,
                        'recarga': recarga
                    }
                }
            }
            console.log('msg send', msg)
            win.postMessage(JSON.stringify(msg), "*");
        }
        catch (ex) {
            console.log("[" + id + "] Error al enviar request: " + ex);
        }
    }
}
```

A continuación se detalla las diferentes operaciones que los módulos pueden solicitar a TeJA y los parámetros que se deben enviar en cada petición.

3.4.2 Operaciones y llamadas permitidas

Operación	Id	Parámetros	Descripción
OP_ABRIR_MODAL	0	titulo	El título que se muestra en el modal que se abre
		url	La URL donde se aloja el módulo desacoplado que se carga
		tamanio	Para dimensionar el modal, se establecen 4 tamaños posibles: S, M, L y XL
		recarga	Indica si al cerrar el modal se requiere refrescar la información que muestra el Tramitador de expedientes con un booleano true/false
OP_DIMENSIONAR_IFRAME	1	etiqueta	Selector jQuery que identifica el elemento HTML sobre el que se realiza la acción
		height	Número entero para dimensionar en px la altura del modal
OP_RELOAD_MODULO	4	time	Número entero de milisegundos que se espera para que se ejecute la operación
OP_CERRAR_MODAL	5		
OP_RECARGA_TRANSICIONES	8		
OP_RECARGA_HISTORICO_DEL_EXPEDIENTE	9		

OP_ABRIR_MENSAJE_TOASTS	10	tipo	El estilo del mensaje toasts que se muestra según el valor aportado: success, warning, error
		mensaje	El mensaje que se muestra en la notificación toasts
OP_RECARGA_USUARIOS_ASIGNADOS	11		
OP_MANDAR_DATA_DE_IFRAME_A_IFRAME	12	etiqTo	Selector jQuery que identifica el elemento HTML que envía información
		etiqFrom	Selector jQuery que identifica el elemento HTML que recibe la información
		data	Mapa de parámetros intercambiados entre dos módulos
OP_ABRIR_VENTANA	13	url	La URL donde se aloja el módulo desacoplado que se carga
OP_OBTENER_CONTEXTO	14	fields	Lista de campos para devolver información adicional al expediente {“sistema”, “instalación”, “usuario”, “interesados”}
OP_PONER_SPINNER	15		
OP_QUITAR_SPINNER	16		

3.4.2.1 OP_ABRIR_MODAL

Operación que permite indicar a TeJA que abra un módulo de tipo utilidad desacoplada en una modal. Cuando TeJA recibe una petición de abrir modal, en primer lugar prepara la ventana modal que se va a abrir en función del “params.tamanio” (realiza operaciones de eliminación de título, eliminación de iframe, etc.).

A continuación, añade el params.titulo al #tituloModalUtilidad[params.tamanio], crea un iframe con la URL params.url y lo añade al #contenidoModalUtilidad[params.tamanio].

- Si “params.recargar” es true se recargará el parent cuando se cierre la modal al pulsar el icono superior derecho de la modal.
- Si “params.recargar” es false pero params.url incluye 'Asigna', se recargará los datos del expediente cuando se cierre la modal al pulsar el icono superior derecho de la modal.
- Si “params.recargar” es false y params.url no incluye 'Asigna', solo se cierra la modal y no se realiza ningún refresco al pulsar el icono superior derecho de la modal.

Una vez inicializada la modal, TeJA envía cada segundo un postMessage al módulo con la petición de bienvenida para que el módulo responda cuando haya cargado completamente el iframe.

Se recomienda acompañar la operación OP_ABRIR_MODAL con las siguientes operaciones:

- OP_PONER_SPINNER para mostrar el icono de espera mientras se carga el contenido del módulo.
- OP_OBTENER_CONTEXTO para obtener los datos del expediente.
- OP_DIMENSIONAR_IFRAME para ajustar el tamaño final del iframe con el contenido del módulo.
- OP_QUITAR_SPINNER para ocultar el icono de espera una vez cargado el contenido del módulo.
- OP_CERRAR_MODAL para cerrar la modal y finalizar la comunicación con el Tramitador.

3.4.2.2 OP_DIMENSIONAR_IFRAME

Operación que permite indicar a TeJA cual es el tamaño (alto de la pantalla) que tiene que asignar a un iframe.

Cuando TeJA envía al módulo el postMessage con la petición de bienvenida, envía la etiqueta con el selector JQuery del iframe donde se cargará el módulo, este selector es el que deberá enviar el módulo como params.etiqueta para dimensionar correctamente el iframe.

Cuando TeJA recibe una petición de dimensionar iframe, asigna al elemento “params.etiqueta” la propiedad css min-height con el valor “params.height” en pixeles.

3.4.2.3 OP_RELOAD_MODULO

Operación que permite indicar a TeJA que recargue un módulo en el escritorio de tramitación. Cuando TeJA recibe una petición de recarga de módulo en el escritorio de tramitación, se realiza un click en el elemento ".div-portletsEscritorio.active" correspondiente a la pestaña activa del grupo de portlets del escritorio.

3.4.2.4 OP_CERRAR_MODAL

Operación que permite indicar a TeJA que cierre una modal. Cuando TeJA recibe una petición de cierre de modal, vacía el contenido de la ventana modal (realiza operaciones de eliminación de título, eliminación de iframe, etc.) y la cierra.

3.4.2.5 OP_RECARGA_TRANSICIONES

Operación que permite indicar a TeJA que recargue las transiciones de la fase seleccionada. Cuando TeJA recibe una petición de recarga de transiciones, realiza una llamada ajax a “listarTransicionesExpediente.action” que provoca la recarga del elemento #bloqueTransiciones de la fase actual seleccionada.

Se recomienda llamar a esta operación, únicamente si desde el módulo externo se han realizado cambios en la tramitación de fases del expediente o en algunas de las condiciones asociadas a la tramitación.

3.4.2.6 OP_RECARGA_HISTORICO_DEL_EXPEDIENTE

Operación que permite indicar a TeJA que recargue el histórico del expediente. Cuando TeJA recibe una petición de recarga de transiciones, realiza una llamada ajax a “listarEvolucion.action” que provoca la recarga del elemento #historico.

Se recomienda llamar a esta operación, únicamente si desde el módulo externo se han realizado cambios en la tramitación de fases del expediente.

3.4.2.7 OP_ABRIR_MENSAJE_TOASTS

Operación que permite indicar a TeJA que muestre un mensaje toasts. Cuando TeJA recibe una petición de mostrar un mensaje toasts, prepara los parámetros necesarios para mostrar el “params.mensaje” con la librería toastr en funcion del “params.tipo” ('success', 'error' o 'warning').

Se recomienda llamar a esta operación para mostrar mensajes informativos al usuario sobre el resultado de las ejecuciones realizadas en el módulo.

3.4.2.8 OP_RECARGA_USUARIOS_ASIGNADOS

Operación que permite indicar a TeJA que recargue los usuarios asignados a un expediente. Cuando TeJA recibe una petición de recarga de usuarios asignados, realiza una llamada ajax a “getUsuariosAsignadosEvo.action” que provoca la recarga del elemento #historicoAsignado.

Se recomienda llamar a esta operación, únicamente si desde el módulo externo se han realizado cambios en los usuarios asignados al expediente.

3.4.2.9 OP_MANDAR_DATA_DE_IFRAME_A_IFRAME

Operación que permite a TeJA intercambiar información entre distintos iframes.

Cuando TeJA recibe una petición de intercambio de datos entre iframes, obtiene el iframe “params.etiqTo” y lanza un evento postMessage para el iframe con “params.etiqFrom” y “params.data”, el iframe de destino captura el evento postMessage y realiza sus propias operaciones en funcion de los datos recibidos.

3.4.2.10 OP_ABRIR_VENTANA

Operación que permite a TeJA abrir una URL en una nueva ventana. Cuando TeJA recibe una petición de apertura de ventana, ejecuta la funcion JavaScript window.open con la URL recibida en params.url.

3.4.2.11 OP_OBTENER_CONTEXTO

Operación que permite solicitar a TeJA información relacionada con un expediente.

Cuando TeJA recibe una petición de obtención de contexto, genera por defecto un JSON con los datos del expediente el cual devolverá en la respuesta. Dicha respuesta se puede ampliar para que devuelva mayor nivel información añadiendo como parámetro de entrada en la llamada a “OP_OBTENER_CONTEXTO” una lista con los siguientes datos: “sistema”, “instalación”, “usuario”, “interesados”.

Además la operación devolverá un token que tendrá que ser validado por el módulo externo. En el siguiente apartado se detallará el formato de dicho JSON y como se debe realizar la comunicación de manera segura para asegurar la veracidad de la información.

3.4.2.12 OP_PONER_SPINNER

Operación que permite indicar a TeJA que muestre un spinner de espera. Cuando TeJA recibe una petición de mostrar spinner, crea un elemento div con el spinner y lo añade al document.

3.4.2.13 OP_QUITAR_SPINNER

Operación que permite indicar a TeJA que quite el spinner de espera. Cuando TeJA recibe una petición de quitar spinner, elimina del document el elemento div con el spinner.

3.4.3 Políticas de seguridad e intercambio de información

Debido a que TeJA proporciona operaciones que suministran a los módulos información detallada de los expedientes, se establecerá una política de seguridad basada en tokens, que permitirá a los módulos validar la integridad de los datos de expediente facilitados por TeJA.

Cuando un módulo necesite obtener información de un expediente, publicará un evento postMessage con la operación OP_OBTENER_CONTEXTO, TeJA capturará el evento y procesará la petición, generando un JSON con los datos del expediente solicitados por el módulo para el expediente cargado en ese momento en el escritorio de tramitación. Una vez generado el JSON con los datos del expediente, TeJA solicitará al servicio de tokens habilitado para ello, la generación de un token con los datos del expediente.

A continuación, se muestra un ejemplo de formato de petición realizada por un módulo externo a TeJA para la operación "OP_OBTENER_CONTEXTO". Por defecto TeJA devuelve únicamente los datos asociados al expediente, para obtener más información se tiene que indicar el atributo fields con la información adicional que se desea obtener:

```
var msg = {
            "request": {
              "id": 1733238228785,
              "operation": 14,
              "fields": [
                  "sistema",
                  "instalacion",
                  "usuario"
              ]
            }
          }
```

Se muestra el formato de respuesta proporcionada por TeJA a la petición de la operación OP_OBTENER_CONTEXTO:

```
var msg = {
            "response": {
              "id": 1733238228785,
              "operation": 14,
              "version": 0
            },
            "payload": {
              "flag": "ACK",
              "data": {
                "token": "eyJhbGciOiJIUzI1NiJ9.eyJqdGkiOiJQVF9ERVMiLCJpYXQiOjE3MzM...",
              }
            }
          }
```

Donde el decode en base64 del token tiene el siguiente formato:

```
"expediente": {
        "id": "1844",
        "csv": "HeKN6S648XKRQ32EYNSNCXDHVGR5U6",
        "num_expediente": "HeKN6020000000210",
        "titulo": "Registro de licitadores de la Comunidad Autónoma de Andalucía",
        "unidad_organizativa": "UO1UO",
        "fechaAlta": "30/04/2024 16:54:23",
        "faseActual": [
          {
            "descripcion": "SOLICITUD REALIZADA DE FORMA MANUAL",
            "id": "53",
          }
        ],
        "procedimiento": {
```

```
      "descripcion": "Registro de Licitadores de Andalucía",
      "abreviatura": "RLCAA",
      "id": "5"
    }
  },
  "sistema": {
    "descripcion": "Consejería de Hacienda, Industria y Energía",
    "jndiTrewa": "TrewaCHIE",
    "idTrewa": "3",
    "codigo": "CHIE"
  },
  "instalacion": {
    "id": "203",
    "nombre": "CHIE"
  },
  "usuario": {
    "nombre": "EIDAS",
    "apellido1": "CERTIFICADO",
    "apellido2": "PRUEBAS",
    "id": "99999999R",
    "unidadOrganizativa": "Consejería de Hacienda, Industria y Energía",
    "puestoTrabajo": {
      "codigo": "TECNICO",
      "nombre": "TECNICO"
    },
    "perfiles": [
      {
        "sistema_idTrewa": "2",
        "sistema_codigo": "TREWA",
        "perfilTrewa": "2",
        "nombre": "TR_R_ADMINISTRADOR"
      },
      {
        "sistema_idTrewa": "2",
        "sistema_codigo": "TREWA",
        "perfilTrewa": "3",
        "nombre": "TR_R_USUARIO"
      },
      {
        "sistema_idTrewa": "3",
        "sistema_codigo": "CHIE",
        "perfilTrewa": "4",
        "nombre": "PF_NOTIF_TECNICO"
      }
    ]
  }
  "interesados": [
      {
        "nombre": "interesado1",
        "apellido1": "ape1interesado1",
        "apellido2": " ape2interesado1",
        "identificador": "99999999R",
        "tipo_identificador": "NIF",
        "razon_interes": "SOLICITANTE"
      },
      {
        "nombre": "interesado2",
        "apellido1": "ape1interesado2",
        "apellido2": "ape2interesado2",
        "identificador": "99999999R",
```

```
        "tipo_identificador": "NIF",
        "razon_interes": "REPRESENTANTE"
      }
    ]
}
```


Para la generación y validación de tokens se hará uso del servicio genérico “vea-tokens-svc”, este servicio proporciona dos endpoints:

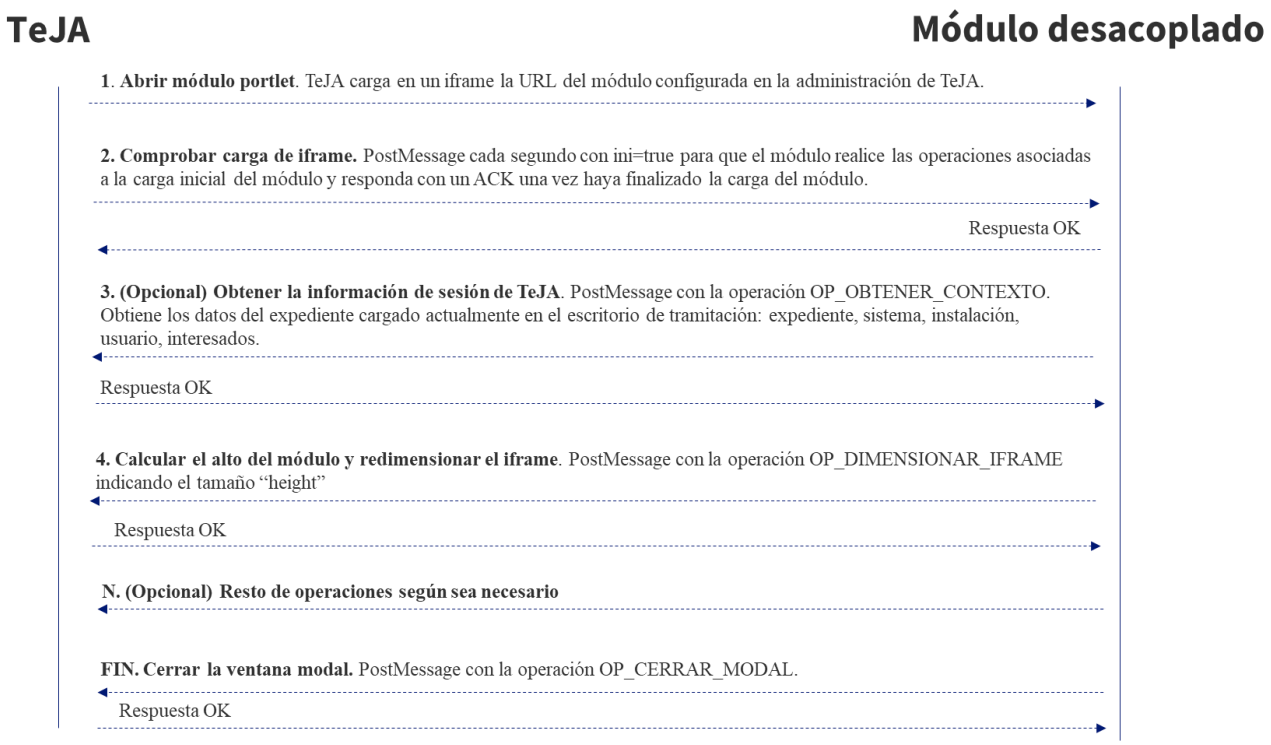
- generateToken: Endpoint que permite generar un token mediante autenticación básica basada en usuario y password. Recibe como parámetro un Map<String, Object> con toda la información que se debe incluir en los datos del token.
- validateToken: Endpoint que permite validar un token.

¡Importante! Se recomienda que los módulos comprueben la integridad de los datos devueltos por la operación “OP_OBTENER_CONTEXTO” de TeJA antes de procesarlos, para ello deberán realizar una petición de validación de token al servicio de tokens habilitado para ello.

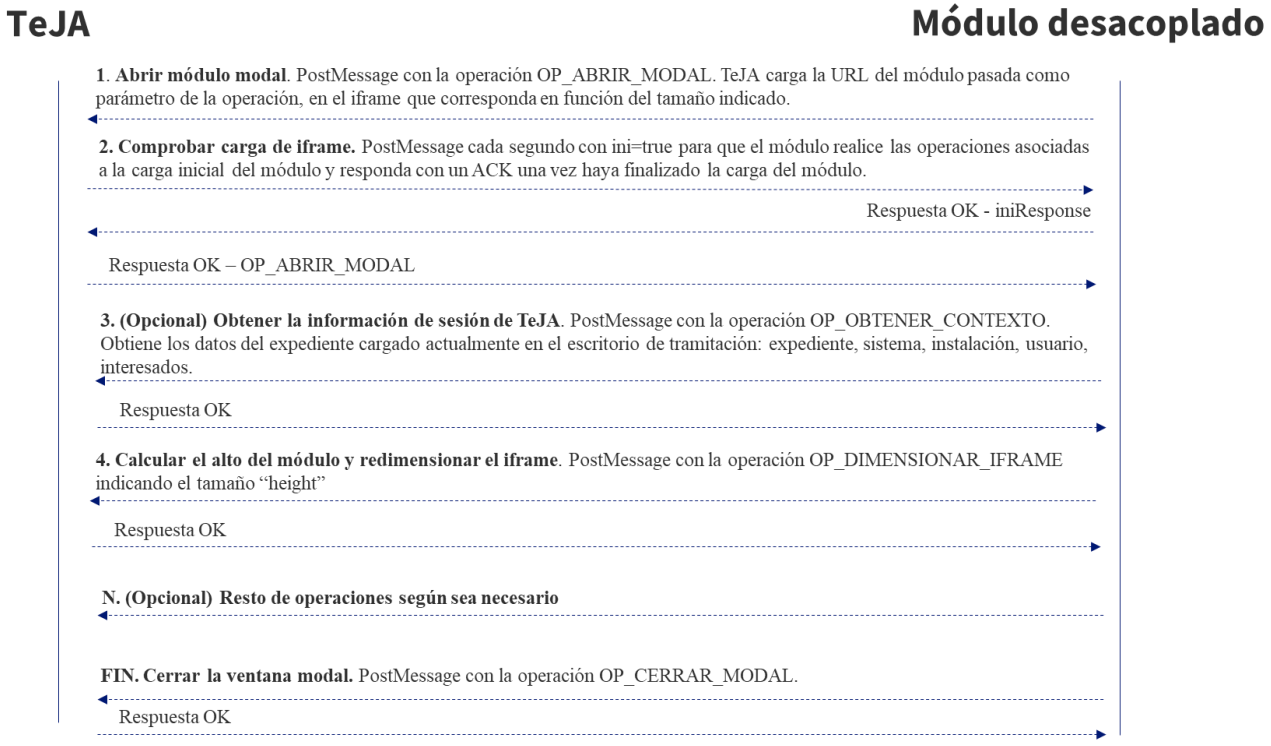
3.4.4 Definición del flujo completo de comunicaciones

A continuación, se presenta un diagrama de secuencia con un ejemplo de las peticiones que se realizarían para abrir un módulo desarrollado de manera externa y desacoplada.

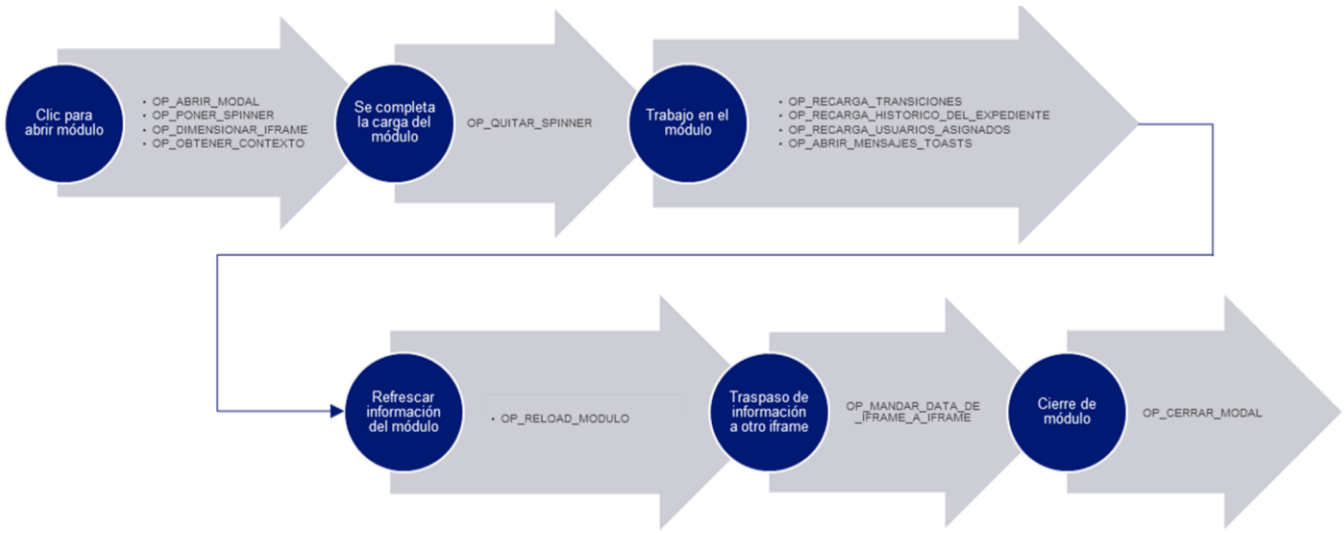
3.4.4.1 Módulo tipo pestaña



3.4.4.2 Módulo tipo utilidad (modal)



Por último, se muestra a modo resumen un diagrama con el conjunto de operaciones y la fase o momento donde se ejecutarían:



Por tanto, cualquier implementación de este tipo de módulos **deberá de cumplir los siguientes requisitos**:

- ✓ Estar diseñados e **implementados de manera externa e independiente** a la arquitectura y despliegue de TeJA
- ✓ Estar desplegados en un **servidor o contenedor independiente** a TeJA
- ✓ La **comunicación** con TeJA deberá realizarse **mediante PostMessage** tal y como se ha indicado en los apartados anteriores
- ✓ El intercambio de **información se deberá realizar de manera segura** siguiendo las indicaciones establecidas en los apartados anteriores.
- ✓ Para el desarrollo de estos módulos externos se recomienda cumplir con las arquitecturas, tecnologías y buenas prácticas definidas por la oficina de arquitectura de la ADA
- ✓ Será de obligado uso la utilización de los ficheros *.scss y *.css de estilos para que la experiencia de usuario sea uniforme al navegar por TeJA y posteriormente abrir los módulos externos desarrollados
- ✓ Para establecer la configuración y comunicación desde TeJA hacia el módulo implementado este se deberá configurar desde la administración de TeJA

3.4.5 Ejemplos de integración

3.4.5.1 Carga de módulo tipo pestaña

Se describe la integración utilizando como ejemplo el módulo de interesados incluido en el escritorio de tramitación de TeJA.

Operaciones implicadas:

- OP_DIMENSIONAR_IFRAME

Integración:

1. Al pulsar sobre la pestaña “INTERESADOS” del escritorio de tramitación, TeJA llama a la función JavaScript cargarIframe({target: 'contenidoPestana-53', width: '100%', url: '../modulos/interesados/listarInteresados.action'}); esta función carga la URL en el iframe #contenidoPestana-53.
2. A continuación, TeJA llama a la función JavaScript iniPortlet(); que cada segundo emite un evento postMessage con el siguiente mensaje:

```
var msg = {
  "response": {
    "ini": true
  },
  "payload": {
    "etiqueta": ".tab-content .active iframe"
  }
}
```

3. El evento es capturado por el módulo de interesados que calcula el alto que se tiene que asignar al módulo y emite un evento postMessage con el siguiente mensaje:

```
var msg = {
  "request": {
    "id": 1733906636819,
    "operation": 1
  },
  "payload": {
    "params": {
      "etiqueta": ".tab-content .active iframe",
       "height": 543
    }
  }
}
```

4. El evento es capturado por TeJA que establece el valor 543px a la propiedad CSS min-height del elemento ".tab-content .active iframe" y emite un evento postMessage con el siguiente mensaje:

```
var msg = {
  "response": {
    "id": 1733904963678,
    "operation": 1,
    "version": 0
  },
  "payload": {
    "flag": "ACK",
    "data": ""
  }
}
```

5. El módulo emite un evento postMessage con el siguiente mensaje:

```
var msgResponse = {
  "response": {
    "iniResponse": true
  },
  "payload": {
    "flag": "ACK",
    "data": ""
  }
}
```

6. El evento es capturado por TeJA que considera que la carga del módulo ha finalizado.

3.4.5.2 Carga de módulo tipo utilidad (modal)

Se describe la integración utilizando como ejemplo la modal “AGREGAR INTERESADO” del módulo de interesados incluido en el escritorio de tramitación de TeJA.

Operaciones implicadas:

- OP_ABRIR_MODAL
- OP_DIMENSIONAR_IFRAME

Integración:

1. Al pulsar sobre el botón “AGREGAR INTERESADO” del módulo, el módulo llama a la función JavaScript sendModal('Agregar interesado', '../modulos/utilidadInteresados/nuevoInteresado.action','L',false); esta función emite un evento postMessage con el siguiente mensaje:

```
var msg = {
  "request": {
    "id": 1733907603696,
    "operation": 0
  },
  "payload": {
    "params": {
      "etiqueta": ".tab-content .active iframe",
      "titulo": "Agregar interesado",
      "url": "../modulos/utilidadInteresados/nuevoInteresado.action",
      "tamanio": "L",
      "recarga": false
    }
  }
}
```

2. El evento es capturado por TeJA que carga el título en el elemento #tituloModalUtilidadL, la URL en el iframe #contenidoModalUtilidadL y muestra la modal #utilidadLargeModal.
3. Una vez abierta la modal, cada segundo, TeJA emite un evento postMessage con el siguiente mensaje:

```
var msg = {
  "response": {
    "ini": true
  },
  "payload": {
    "etiqueta": ".modal.show iframe"
  }
}
```

4. El evento es capturado por el módulo de interesados que calcula el alto que se tiene que asignar a la modal y emite un evento postMessage con el siguiente mensaje:

```
var msg = {
  "request": {
    "id": 1733908262974,
    "operation": 1
  },
  "payload": {
    "params": {
      "etiqueta": ".modal.show iframe",
      "height": 502
    }
  }
}
```

5. A continuación, el módulo emite un evento postMessage con el siguiente mensaje:

```
var msgResponse = {
  "response": {
    "iniResponse": true
  },
  "payload": {
    "flag": "ACK",
    "data": ""
  }
}
```

6. El evento del punto 4 con la "operation": 1 es capturado por TeJA que establece el valor 502px a la propiedad CSS min-height del elemento .modal.show iframe y emite un evento postMessage con el siguiente mensaje:

```
var msg = {
  "response": {
    "id": 1733908262974,
    "operation": 1,
    "version": 0
  },
  "payload": {
    "flag": "ACK",
    "data": ""
  }
}
```

7. El evento es capturado por el módulo de interesados y no realiza ninguna operación más.

8. El evento del punto 5 con "iniResponse": true es capturado por TeJA que considera que la carga del módulo ha finalizado.

3.4.5.3 Recarga de módulo

Se describe la integración mínima necesaria para recargar un módulo.

Operaciones implicadas:

- OP_RELOAD_MODULO

Integración:

1. Al pulsar sobre un botón “RECARGAR” del módulo, se llamará a una función JavaScript reloadModulo(); esta función emitirá un evento postMessage con el siguiente mensaje:

```
var msg = {
  "request": {
    "id": 1733912207279,
    "operation": 4
  },
  "payload": {
    "params": {
      "etiqueta": ".tab-content .active iframe",
      "time": 1000
    }
  }
}
```

2. El evento es capturado por TeJA que ejecuta el evento click sobre el elemento ".div-portletsEscritorio.active" del escritorio de tramitación y emite un evento postMessage con el siguiente mensaje:

```
var msg = {
  "response": {
    "id": 1733912207279,
    "operation": 4,
    "version": 0
  },
  "payload": {
    "flag": "ACK",
    "data": ""
  }
}
```

3. El evento es capturado por el módulo que procesa el mensaje y no realiza ninguna operación más.

3.4.5.4 Recarga de transiciones

Se describe la integración mínima necesaria para recargar las transiciones del escritorio de tramitación.

Operaciones implicadas:

- OP_RECARGA_TRANSICIONES

Integración:

1. Al pulsar sobre un botón, por ejemplo “RECARGAR TRANSICIONES” del módulo externo, se llamará a una función JavaScript recargaTransiciones(); esta función emitirá un evento postMessage con el siguiente mensaje:

```
var msg = {
  "request": {
    "id": 1733912797174,
    "operation": 8
  },
  "payload": {
    "params": {
      "etiqueta": ".tab-content .active iframe"
    }
  }
}
```

2. El evento es capturado por TeJA que ejecuta una petición AJAX al action "../modulos/tramitacion/listarTransicionesExpediente.action", actualiza el contenido del elemento "#bloqueTransiciones-refFase" con el resultado del action y emite un evento postMessage con el siguiente mensaje:

```
var msg = {
  "response": {
    "id": 1733912797174,
    "operation": 8,
    "version": 0
  },
  "payload": {
    "flag": "ACK",
    "data": ""
  }
}
```

3. El evento es capturado por el módulo que procesa el mensaje y no realiza ninguna operación más.

3.4.5.5 Recarga de histórico

Se describe la integración mínima necesaria para recargar el histórico del expediente del escritorio de tramitación.

Operaciones implicadas:

- OP_RECARGA_HISTORICO_DEL_EXPEDIENTE

Integración:

1. Al pulsar sobre un botón, por ejemplo “RECARGAR HISTORICO” del módulo externo, se llamará a una función JavaScript recargaHistorico(); esta función emitirá un evento postMessage con el siguiente mensaje:

```
var msg = {
  "request": {
    "id": 1733913177150,
    "operation": 9
  },
  "payload": {
    "params": {
      "etiqueta": ".tab-content .active iframe"
    }
  }
}
```

2. El evento es capturado por TeJA que ejecuta una petición AJAX al action "../modulos/evolucion/listarEvolucion.action", actualiza el contenido del elemento "#historico" con el resultado del action y emite un evento postMessage con el siguiente mensaje:

```
var msg = {
  "response": {
    "id": 1733913177150,
    "operation": 9,
    "version": 0
  },
  "payload": {
    "flag": "ACK",
    "data": ""
  }
}
```

3. El evento es capturado por el módulo que procesa el mensaje y no realiza ninguna operación más.

3.4.5.6 Recarga de usuarios asignados

Se describe la integración mínima necesaria para recargar de los usuarios asignados del expediente del escritorio de tramitación.

Operaciones implicadas:

- OP_RECARGA_USUARIOS_ASIGNADOS

Integración:

1. Al pulsar sobre un botón, por ejemplo “RECARGAR HISTORICO” del módulo externo, se llamará a una función JavaScript recargaHistorico(); esta función emitirá un evento postMessage con el siguiente mensaje:

```
var msg = {
  "request": {
    "id": 1733913398326,
    "operation": 11
  },
  "payload": {
    "params": {
      "etiqueta": ".tab-content .active iframe"
    }
  }
}
```

2. El evento es capturado por TeJA que ejecuta una petición AJAX al action "../modulos/usuariosAsignados/getUsuariosAsignadosEvo.action", actualiza el contenido del elemento "#historicoAsignado" con el resultado del action y emite un evento postMessage con el siguiente mensaje:

```
var msg = {
  "response": {
    "id": 1733913398326,
    "operation": 11,
    "version": 0
  },
  "payload": {
    "flag": "ACK",
    "data": ""
  }
}
```

3. El evento es capturado por el módulo que procesa el mensaje y no realiza ninguna operación más.

3.4.5.7 Abrir ventana

Se describe la integración mínima necesaria para abrir una URL en una nueva ventana.

Operaciones implicadas:

- OP_ABRIR_VENTANA

Integración:

1. Al pulsar sobre un botón, por ejemplo “ABRIR VENTANA” del módulo externo, se llamará a una función JavaScript abrirVentana("https://juntadeandalucia.es/"); esta función emitirá un evento postMessage con el siguiente mensaje:

```
var msg = {
  "request": {
    "id": 1733915064177,
```

```
    "operation": 13
  },
  "payload": {
    "params": {
      "etiqueta": ".tab-content .active iframe",
      "url": "https://juntadeandalucia.es/"
    }
  }
}
```

2. El evento es capturado por TeJA que ejecuta la función JavaScript window.open(url); y emite un evento postMessage con el siguiente mensaje:

```
var msg = {
  "response": {
    "id": 1733915064177,
    "operation": 13,
    "version": 0
  },
  "payload": {
    "flag": "ACK",
    "data": ""
  }
}
```

3. El evento es capturado por el módulo que procesa el mensaje y no realiza ninguna operación más.

3.4.5.8 Mostrar mensaje toast

Se describe la integración mínima necesaria para mostrar un mensaje toast en TeJA.

Operaciones implicadas:

- OP_ABRIR_MENSAJE_TOASTS

Integración:

1. Al pulsar sobre un botón, por ejemplo “MOSTRAR MENSAJE” del módulo externo, se llamará a una función JavaScript mostrarMensajeToast('success', 'Prueba de mensaje toast.'); esta función emitirá un evento postMessage con el siguiente mensaje:

```
var msg = {
  "request": {
    "id": 1733919007325,
    "operation": 10
  },
  "payload": {
    "params": {
      "etiqueta": ".tab-content .active iframe",
      "tipo": "success",
      "mensaje": "Prueba de mensaje toast."
    }
  }
}
```

2. El evento es capturado por TeJA que muestra el mensaje “Prueba de mensaje toast.” con estilo success mediante la librería toastr y emite un evento postMessage con el siguiente mensaje:

```
var msg = {
  "response": {
    "id": 1733919007325,
    "operation": 10,
    "version": 0
  },
  "payload": {
    "flag": "ACK",
    "data": ""
  }
}
```

3. El evento es capturado por el módulo que procesa el mensaje y no realiza ninguna operación más.

3.4.5.9 Mostrar spinner

Se describe la integración mínima necesaria para mostrar un spinner en TeJA.

Operaciones implicadas:

- OP_PONER_SPINNER

Integración:

1. Al pulsar sobre un botón, por ejemplo “MOSTRAR SPINNER” del módulo externo, se llamará a una función JavaScript mostrarSpinner(); esta función emitirá un evento postMessage con el siguiente mensaje:

```
var msg = {
  "request": {
    "id": 1733919395870,
    "operation": 15
  }
}
```

2. El evento es capturado por TeJA que muestra el spinner y emite un evento postMessage con el siguiente mensaje:

```
var msg = {
  "response": {
    "id": 1733919395870,
    "operation": 15,
    "version": 0
  },
  "payload": {
    "flag": "ACK",
    "data": ""
  }
}
```

3. El evento es capturado por el módulo que procesa el mensaje y no realiza ninguna operación más.

3.4.5.10 Ocultar spinner

Se describe la integración mínima necesaria para ocultar un spinner en TeJA.

Operaciones implicadas:

- OP_QUITAR_SPINNER

Integración:

1. Al pulsar sobre un botón, por ejemplo “OCULTAR SPINNER” del módulo externo, se llamará a una función JavaScript ocultarSpinner(); esta función emitirá un evento postMessage con el siguiente mensaje:

```
var msg = {
  "request": {
    "id": 1733919620957,
    "operation": 16
  }
}
```

2. El evento es capturado por TeJA que oculta el spinner y emite un evento postMessage con el siguiente mensaje:

```
var msg = {
  "response": {
    "id": 1733919620957,
    "operation": 16,
    "version": 0
  },
  "payload": {
    "flag": "ACK",
    "data": ""
  }
}
```

3. El evento es capturado por el módulo que procesa el mensaje y no realiza ninguna operación más.

3.4.5.11 Obtener contexto

Se describe la integración mínima necesaria para obtener los datos del expediente cargado actualmente en el escritorio de tramitación de TeJA.

Operaciones implicadas:

- OP_OBTENER_CONTEXTO

Integración:

1. Al pulsar sobre un botón, por ejemplo “OBTENER CONTEXTO” del módulo externo, se llamará a una función JavaScript obtenerContexto(); esta función emitirá un evento postMessage con el siguiente mensaje:

```
var msg = {
  "request": {
    "id": 1733920459167,
    "operation": 14,
    "fields": [
        "sistema",
        "instalacion",
        "usuario"
    ]
  }
}
```

2. El evento es capturado por TeJA que genera un JSON con los datos del expediente cargado actualmente en el escritorio de tramitación, solicita un token al servicio de tokens habilitado para ello y emite un evento postMessage con el siguiente mensaje:


```
var msg = {
  "response": {
    "id": 1733238228785,
    "operation": 14,
    "version": 0
  },
  "payload": {
    "flag": "ACK",
    "data": {
      "token": "eyJhbGciOiJIUzI1NiJ9.eyJqdGkiOiJQVF9ERVMiLCJpYXQiOjE3MzM...",
      }
  }
}
```


Donde el decode en base 64 del token proporciona la siguiente información:


```
"expediente": {
  "id": "1844",
  "csv": "HeKN6S648XKRQ32EYNSNCXDHVGR5U6",
  "num_expediente": "HeKN6020000000210",
  "titulo": "Registro de licitadores de la Comunidad Autónoma de Andalucía",
  "unidad_organizativa": "UO1UO",
  "fechaAlta": "30/04/2024 16:54:23",
  "faseActual": [
    {
      "descripcion": "SOLICITUD REALIZADA DE FORMA MANUAL",
      "id": "53",
    }
  ],
  "procedimiento": {
    "descripcion": "Registro de Licitadores de Andalucía",
    "abreviatura": "RLCAA",
    "id": "5"
  }
},
"sistema": {
  "descripcion": "Consejería de Hacienda, Industria y Energía",
  "jndiTrewa": "TrewaCHIE",
  "idTrewa": "3",
  "codigo": "CHIE"
},
"instalacion": {
  "id": "203",
  "nombre": "CHIE"
},
"usuario": {
  "nombre": "EIDAS",
  "apellido1": "CERTIFICADO",
  "apellido2": "PRUEBAS",
  "id": "99999999R",
  "unidadOrganizativa": "Consejería de Hacienda, Industria y Energía",
  "puestoTrabajo": {
    "codigo": "TECNICO",
    "nombre": "TECNICO"
  },
  "perfiles": [
```

```
        {
          "sistema_idTrewa": "2",
          "sistema_codigo": "TREWA",
          "perfilTrewa": "2",
          "nombre": "TR_R_ADMINISTRADOR"
        },
        {
          "sistema_idTrewa": "2",
          "sistema_codigo": "TREWA",
          "perfilTrewa": "3",
          "nombre": "TR_R_USUARIO"
        },
        {
          "sistema_idTrewa": "3",
          "sistema_codigo": "CHIE",
          "perfilTrewa": "4",
          "nombre": "PF_NOTIF_TECNICO"
        }
      ]
    }
    "interesados": [
        {
          "nombre": "interesado1",
          "apellido1": "ape1interesado1",
          "apellido2": " ape2interesado1",
          "identificador": "99999999R",
          "tipo_identificador": "NIF",
          "razon_interes": "SOLICITANTE"
        },
        {
          "nombre": "interesado2",
          "apellido1": "ape1interesado2",
          "apellido2": "ape2interesado2",
          "identificador": "99999999R",
          "tipo_identificador": "NIF",
          "razon_interes": "REPRESENTANTE"
        }
      ]
```


3. El evento es capturado por el módulo que procesa el mensaje y no realiza ninguna operación más.

3.4.5.12 Enviar datos de un iframe a otro iframe

Se describe la integración mínima necesaria para enviar datos de un iframe a otro iframe pasando por TeJA.

Operaciones implicadas:

- OP_MANDAR_DATA_DE_IFRAME_A_IFRAME

Integración:

1. Al pulsar sobre un botón, por ejemplo “ENVIAR DATOS A IFRAME” del módulo externo, se llamará a una función JavaScript sendDataIframeToIframe(); esta función emitirá un evento postMessage con el siguiente mensaje:

```
var msg = {
  "request": {
    "id": 1733920723894,
    "operation": 12
  },
```

```
  "payload": {
    "params": {
      "etiqFrom": ".tab-content .active iframe",
      "etiqTo": "#contenidoPestana-53 iframe",
      "data": "PRUEBA"
    }
  }
}
```

2. El evento es capturado por TeJA que obtiene el iframe "#contenidoPestana-53 iframe" y emite un evento postMessage con el siguiente mensaje:

```
var msg = {
  "request": {
    "id": 1733921143890,
    "operation": 12
  },
  "payload": {
    "params": {
      "etiqFrom": ".tab-content .active iframe",
      "data": "PRUEBA"
    }
  }
}
```

3. El evento es capturado por el iframe "#contenidoPestana-53 iframe" que procesa el mensaje y no realiza ninguna operación más.

4 Guía de desarrollo para servicios externos

4.1 SCSP externo

El objetivo de esta sección es detallar las especificaciones y pautas a seguir para el desarrollo de un servicio SCSP externo mediante un servicio REST a diferencia que el interno es un módulo propio de TeJA. Esta funcionalidad va a coexistir con la forma tradicional de llamar a los servicios SCSP por reflexión java. Para configurar un servicio SCSP externo se deberá realizar de forma análoga a la actual en Trewa indicando que se trata de un servicio externo e indicando su URL.

La arquitectura que se ha definido consta de un procedimiento en dos pasos para completar una comunicación completa con los servicios de interoperabilidad de SCSP. Para ello el servicio REST externo deberá publicar dos endpoints que deberán ir con seguridad mediante usuario y contraseña (Basic Auth)

- **/otros-datos**: endpoint encargado de devolver un Map<String, String> con las claves de los datos específicos necesarios para la consulta. Dichos datos serán utilizados posteriormente por TeJA para que pueda recuperarlos de los formularios cumplimentados en el expediente sobre el que se realiza la consulta SCSP.
- **/consulta**: endpoint encargado de realizar la consulta al servicio SCSP mediante las interfaces "SCSPExternoRequest" y "SCSPExternoResponse" que se detallarán en el próximo apartado.

A continuación se detalla la secuencia de pasos:

1. **Consultar datos específicos de formulario**. Existen servicios SCSP que necesitan de unos datos específicos para poder realizar la consulta. Para ello TeJA realizará una primera consulta a la ruta que se haya configurado de servicio externo a un recurso del endpoint publicado con el nombre concreto de **"/otros-datos"**. Este método GET, que deberá implementar el servicio externo, le devolverá a TeJA Map<String, String> con el listado de datos específicos que necesita para completar la consulta SCSP. Este mapa deberá rellenarse con las claves provenientes de las casillas de los formularios y dejando sus valores vacíos.
2. **Construcción de la petición al servicio externo**. A continuación TeJA se encargará de rellenar dicho mapa con los datos aportados en los formularios del expediente, y lo enviará como parte del cuerpo (otrosDatos del objeto "SCSPExternoRequest") en la siguiente comunicación con el endpoint **"/consulta"**. Este endpoint deberá de implementar toda la lógica necesaria para completar la comunicación con los servicios de SCSP, así como de generar un informe en pdf del mismo controlando todo los posibles errores.
3. **Respuesta del servicio externo**. El servicio SCSP externo tras completar toda la lógica interna necesaria para la comunicación con los servicios de interoperabilidad devolverá a TeJA un cuerpo "SCSPExternoResponse" con los datos:
    - **pdf**, siendo el informe pdf generado en base64
    - **xml**, siendo este el xml tal y como lo devuelva la comunicación con SCSP para guardar auditoria de la consulta.
    - **message** que se podrá rellenar con los posibles mensajes de error tanto propios del servicio como del sistema de interoperabilidad que surjan para TeJA pueda indicar al usuario el origen del fallo.

4.1.1 Interfaces necesarias: SCSPExternoRequest – SCSPExternoResponse

A continuación, se detalla la estructura de las interfaces a implementar necesarias para poder realizar la consulta al servicio SCSP externo.

La interfaz SCSPExternoRequest será la que contenga los parámetros (todos de tipo String) necesarios para realizar la petición al Servicio SCSP:

- SCSPExternoRequest
    - EntidadSolicitante
        - Finalidad
        - Consentimiento

        - Funcionario
            - NombreCompletoFuncionario
            - NifFuncionario
        - Titular
            - Nombre
            - Apellido 1
            - Apellido 2
            - TipoDocumentacion
            - Documentacion
            - FechaNacimiento
            - Residencia
                - CodMunicipio
                - CodProvincia
                - País
    - OtrosDatos

La interfaz SCSPExternoRequest será la que contenga los parámetros necesarios para responder a la petición al Servicio SCSP:

- SCSPExternoRequest
    - Pdf (byte[])
    - Xml (Srting)
    - Message (String)

4.1.1.1 Ejemplo de petición JSON de SCSPExternoRequest

```
--header 'Content-Type: application/json' \
--header 'Accept: application/json' \
--header 'Authorization: Basic VEVKQV9ERVM6VEVKQV9ERVM=' \
--data '{
    "entidadSolicitante": {
        "finalidad": "S4111001F_TEST_00001#::#00000000000003928443#::#null",
        "consentimiento": "S",
        "funcionario": {
            "nombreCompletoFuncionario": "Fernandez Eguren, Abraham",
            "nifFuncionario": "72158753W"
        },
        "titular": {
            "nombre": "PRUEBAS",
            "apellido1": "EIDAS",
            "apellido2": "CERTIFICADO",
            "tipoDocumentacion": "NIF",
            "documentacion": "99999999R",
            "fechaNacimiento": "2025-05-02T13:05:26.0000",
            "residencia": {
                "codMunicipio": "",
                "codProvincia": "",
                "pais": "108"
            }
        }
```

```
    },
    "otrosDatos": {
        "CSV_ESCRITURA": "2021031009c6b9bd90ba8143",
        "NRO_PROTOCOLO": "764",
        "PREF_BIS_SI": "false",
        "NOTARIO": "9101049",
        "FECHA_AUTORIZACION": "24/02/2014",
        "LUGAR_NOTARIA": "070407056"
    }
}
```


4.1.1.2 Ejemplo de JSON de SCSPExternoResponse


```
{
    "pdf": [omitido por reducción de tamaño. Seria el pdf en base64],
    "xml": "<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <SOAP-ENV:Body>
                <ns:peticionCliente2BusResponse xmlns:ns="http://www.openuri.org/">
                        <pet:peticion xmlns:pet="http://wawa.es/bus/PeticionCertificado">
                                <pet1:busObject xmlns:pet1="http://wawa.es/bus/PeticionType">
                                        <bus:exception xmlns:bus="http://wawa.es/bus/busObjectType">
                                                <bus1:sesion
xmlns:bus1="http://wawa.es/bus/busExceptionType"/>
                                                <bus1:exception
xmlns:bus1="http://wawa.es/bus/busExceptionType">

        <bus1:errorCodeFuncional>0101</bus1:errorCodeFuncional>
                                                        <bus1:errorDescFuncional>Error al procesar la
peticion sincrona. Error al contactar con el servicio Web especificado
https://intermediacionpp.redsara.es/servicios/SVD/Notarios.ConsultaSubsistencia
peticionSincrona.</bus1:errorDescFuncional>
                                                        <bus1:errorCodeTecnico>0101</bus1:errorCodeTecnico>
                                                        <bus1:errorDescTecnico>Error al procesar la peticion
sincrona. Error al contactar con el servicio Web especificado
https://intermediacionpp.redsara.es/servicios/SVD/Notarios.ConsultaSubsistencia
peticionSincrona.</bus1:errorDescTecnico>
                                                        <bus1:fecha>Fri May 16 14:33:56 CEST
2025</bus1:fecha>
                                                        <bus1:entorno>PRE-PRODUCCION</bus1:entorno>
                                                        <bus1:mensajeEntrada>Error al procesar la peticion
sincrona. Error al contactar con el servicio Web especificado
https://intermediacionpp.redsara.es/servicios/SVD/Notarios.ConsultaSubsistencia
peticionSincrona.</bus1:mensajeEntrada>
                                                </bus1:exception>
                                        </bus:exception>
                                        <ns2:componenteDestino xmlns:ns2="http://wawa.es/bus/busObjectType">
```

```
                                                <ns3:usuario
xmlns:ns3="http://wawa.es/bus/componenteType">CHAP-003</ns3:usuario>
                                                <ns4:password
xmlns:ns4="http://wawa.es/bus/componenteType">CHAP3</ns4:password>
                                        </ns2:componenteDestino>
                                </pet1:busObject>
                                <pet1:Atributos xmlns:pet1="http://wawa.es/bus/PeticionType">
                                        <pet1:idPeticion>NOTPODER00000000000000283</pet1:idPeticion>
                                        <pet1:codCertificado>SVDNOTSUBWS01</pet1:codCertificado>
                                </pet1:Atributos>
                        </pet:peticion>
                </ns:peticionCliente2BusResponse>
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>"
    "message": "Error al procesar la peticion sincrona. Error al contactar con el servicio Web especificado
https://intermediacionpp.redsara.es/servicios/SVD/Notarios.ConsultaSubsistencia peticionSincrona."
}
```


4.1.2 Tratamiento de la petición del servicio REST externo

El servicio REST externo deberá realizar al menos el siguiente tratamiento de la petición realizada por TeJA:

1. **Verificación de seguridad**. Deberá verificar que los datos pasados como autenticación básica (usuario y contraseña) sean válidos y estén autorizados a realizar la consulta SCSP.

2. **Componer la petición** correspondiente al servicio SCSP que se desee consultar con los datos de EntidadSolicitante (Finalidad, Consentimiento, Titular y funcionario) y añadir los datos específicos necesarios para dicho servicio SCSP a través del campo “otrosDatos” que recibe en la petición.

3. **Realizar la petición** al servicio SCSP correspondiente, gestionando el control de errores.

4.1.3 Tratamiento de la respuesta del servicio REST externo

El servicio REST externo deberá realizar un tratamiento de la respuesta xml del servicio SCSP recopilando toda la información necesaria sobre la consulta y generando un informe o justificante de la consulta en formato pdf, devolviendo:

- **pdf**, fichero pdf en base64 con el informe generado de la consulta SCSP. El formato y estilo de dicho fichero pdf deberá consensuarse con la dirección de proyecto de la ADA con el objetivo que todos los justificantes de consultas SCSP sean homogéneos.
- **xml**, mismo xml sin modificar tal y como lo devuelve la comunicación con el servicio SCSP. Es necesario para que TeJA almacene dicho xml como auditoria de la consulta .
- **Message**, cumplimentado con los posibles mensajes de error tanto propios del servicio como del sistema de interoperabilidad que surjan para TeJA pueda indicar al usuario el origen del fallo.

Como parte del procesamiento posterior, TeJA firmará en servidor el informe pdf obtenido de la consulta SCSP y se incorporará al expediente.

4.1.4 Integraciones Servicios SCSP y Tipología

Nombre del servicio	Tipo de Integración
AEAT - Nivel de renta intermediado	Interna
AEAT Consulta de Contrataciones	Interna
AEAT - Obligaciones tributarias – Ayudas y subvenciones	Interna
CCAA - Corriente de pago para contratación	Interna
CCAA - Corriente de pago para ayudas y subvenciones	Interna
CISJUFI - Consulta de título de familia numerosa	Interna
CISJUFI - Consulta de datos de discapacidad	Interna
CORPME - Consulta del registro público concursal	Interna
DGP - Consulta de datos de identidad	Interna
IGAE - Consulta de inhabilitaciones y concesiones para subvenciones y ayudas de BDNS	Interna
INE - Consulta de los datos de convivencia actual	Interna
INE - Consulta de datos de residencia con fecha de última variación padronal	Interna
INSS - Prestaciones públicas del RPSP e incapacidad temporal, maternidad y paternidad	Interna
INSS - Histórico de prestaciones públicas incapacidad temporal en un periodo	Interna
TGSS - Estar al corriente de pago con la seguridad social	Interna
CGN - Consulta de copia simple de poderes notariales	Externa
CGN - Consulta de subsistencia de poderes notariales	Externa
CGN - Consulta de subsistencia de administradores	Externa

4.2 Variables externas

El objetivo de esta sección es detallar las especificaciones y pautas a seguir para el desarrollo de servicios externos para las obtención de los valores de las variables de un documento de tipo generar.

Puesto que un documento de tipo generar puede tener asociadas N variables, para optimizar el cálculo de estas, se ha desarrollado una interfaz de variables externas que permite la obtención de un conjunto de variables en una única llamada al servicio correspondiente para el cálculo. Dicho servicio será de tipo REST, tiene seguridad (user/pass) al realizar la petición y tendrá la siguiente interfaz (los antiguos parámetros para las variables de Trewa se le pasan como datos en la petición: Nº expediente, CSV, nombre de la fase y nombre tipo documento):

4.2.1 Interfaz

Parámetros de entrada:

- **NUM_EXP**: Número del expediente - String
- **CSV**: Código CSV del expediente - String
- **NOMBRE_FASE**: Nombre de la fase actual - String
- **NOMBRE_TIPO_DOCUMENTO**: Nombre del tipo documento -String

- **LISTA_VARIABLES**: Lista de variables que se quieren obtener - Map<String, String> cuyas claves serán las variables a calcular por el servicio externo permitiendo el cálculo de múltiples variables por llamada. Los valores no será necesario enviarlos y se establecerán a vacío o nulo.

Por ejemplo:

```
{
  "NUM_EXP":"RLCAA2533085",
  "CSV":"HeKN6MXVXGA5Q92A4KFKD29TUACKMF",
  "NOMBRE_FASE":"TRAMITACIÓN",
  "NOMBRE_TIPO_DOCUMENTO": "GD_INFORME_TECNICO",
  "LISTA_VARIABLES":{
    "PERSONA_DENUNCIADA": null,
    "LUGAR_DENUNCIA": null,
    "IMPORTE_DENUNCIA":null,
    "DATO_NEGOCIO_1": null,
    "DATO_NEGOCIO_2":null,
    "DATO_NEGOCIO_3":null
  }
}
```

Parámetros de salida:

- **LISTA_VARIABLES**: Lista de variables Map<String, String> cuyos valores han sido rellenado por el servicio externo tras el cálculo de cada una de ellas. La lista de variables será la misma que las enviadas en la petición anterior.

Por ejemplo:

```
{

    "PERSONA_DENUNCIADA":  "Persona de prueba",
    "LUGAR_DENUNCIA":  "Sevilla”,
    "IMPORTE_DENUNCIA": "600",
    "DATO_NEGOCIO_1": "valor_negocio_1",
    "DATO_NEGOCIO_2": "valor_negocio_1",
    "DATO_NEGOCIO_3": "valor_negocio_1"

}
```

4.3 Condiciones y acciones externas

El objetivo de esta sección es detallar las especificaciones y pautas a seguir para el desarrollo de servicios externos para condiciones y acciones externas. Estos servicios REST permiten devolver el resultado de la ejecución de una o varias condiciones o acciones y existirán dos interfaces distintas en función de cada caso (acciones/condiciones individuales o múltiples).

En los datos de entrada de las interfaces se han establecido tanto los identificadores internos de base de datos (para aquellos servicios que si dispongan de acceso a la base de datos de Trewa y quieran realizar las búsquedas por ids), como los campos códigos o nombres (para aquellos servicios que no dispongan de acceso a la base de datos de Trewa y en su modelo de negocio tengan referenciados dichos códigos o nombres).

4.3.1 Obtención de una única condición o acción externa

Las implementaciones de un única condición o acción externa serán las encargadas de devolver el resultado de la acción o condición especificada para los datos de entrada de la petición. El servicio será de tipo REST, no tiene seguridad al realizar la petición y tendrá la siguiente interfaz:

4.3.1.1 Interfaz

Parámetros de entrada:

Parámetro en la interfaz	Tipo	Columna en BBDD Trewa	Tabla en BBDD Trewa
ID_CONDICION	String	X_COAC	TR_CONDICIONES_ACCIONES
ID_EXPEDIENTE	String	X_EXPE	TR_EXPEDIENTES
ID_DOCUMENTO_PERMITIDO	String	X_DOPE	TR_DOCUMENTOS_PERMITIDOS
ID_TRANSICION	String	X_TRAN	TR_TRANSICIONES
ID_EXP_FASE	String	X_EXEF	TR_EXPEDIENTES_EN_FASE
ID_TIPO_EVOL	String	TIEV_X_TIEV	TR_EXPEDIENTES_EN_FASE
NOMB_COND	String	C_NOMBRE	TR_CONDICIONES_ACCIONES
CSV_EXP	String	C_CSV	TR_EXPEDIENTES
ETIQ_DOC_PER	String	T_ETIQUETA	TR_DOCUMENTOS_PERMITIDOS
ETIQ_TRANS	String	T_ETIQUETA	TR_TRANSICIONES
NOMB_FASE	String	C_NOMBRE	TR_FASES
ABRV_TIPO_EVOL	String	C_ABREVIATURA	TR_TIPOS_EVOLUCIONES
COD_RPA_TIPO_EVOL	String	C_RPA	TR_TIPOS_EVOLUCIONES
FECHA_ENTRADA_FASE	String	F_SALIDA	TR_EXPEDIENTES_EN_FASE
IDENTIFICADOR_ENI_EXP	String	T_IDENTIFICADOR_ENI	TR_EXPEDIENTES

Por ejemplo:


```
{
  "ID_CONDICION": "128",
  "ID_EXPEDIENTE": "10953",
  "ID_DOCUMENTO_PERMITIDO": null,
  "ID_TRANSICION": "117",
  "ID_EXP_FASE": "25522",
  "ID_TIPO_EVOL": "4",
  "NOMB_COND": "VALIDAR_CALIF_RESOLUCION",
  "CSV_EXP": "TREW1W788T8F8TR8VBDUXU4N3U4RRJ",
  "ETIQ_DOC_PER": "doc_resol",
  "ETIQ_TRANS": "resolver",
  "NOMB_FASE": "RESOLUCION",
  "ABRV_TIPO_EVOL": "RLCAA",
  "COD_RPA_TIPO_EVOL": "10599",
  "FECHA_ENTRADA_FASE": "25/11/20 13:50:00",
  "IDENTIFICADOR_ENI_EXP": "ES_A01002823_2025_EXP_0010599_2025_TREW1300020256447"
}
```


Parámetros de salida:

- **Boolean** true o false en función del resultado de la ejecución

4.3.2 Obtención de múltiples condiciones y acciones externas

Las implementaciones de múltiples condiciones o acciones externas serán las encargadas de devolver el resultado de la acción o condición especificada para los datos de entrada de la petición. El servicio será de tipo REST y tendrá la siguiente interfaz:

4.3.2.1 Interfaz

Parámetros de entrada:

- **MAP_NOMBRE_PARAMETROS**: Mapa con clave tipo TrCondicionAccion y como valores el listado de parámetros método de la condición análogos a los pasados en la interfaz de la ejecución sobre una única condición o acción externa - Map<TrCondicionAccion, List<ParametroMetodo>>

Parámetros de salida:

- **HashMap<String, Boolean>** con el listado de tuplas del resultado de la ejecución de cada una de las acciones o condiciones externas

4.4 Numeradores externos

El objetivo de esta sección es detallar las especificaciones y pautas a seguir para el desarrollo de un servicio de numerador externo mediante un servicio REST. Esta funcionalidad va a coexistir con la forma tradicional de llamar a los servicios de numeradores por reflexión java.

La implementación actual de numeradores externos posee una interfaz establecida. Se ha definido un funcionamiento por el cual TeJA realizará una petición de tipo POST al servicio de numerador configurado en Trewa y este deberá devolver una cadena con el número de expediente .

Parámetros de entrada:

- CodigoTrewa
- Procedimiento
    - Abreviatura
    - CodigoRPS
- UnidadOrganidca
    - CodigoDIR3
    - CodigoProvincia
- TipoExpediente
    - Abreviatura
    - FechaVigor
- Origen
    - Vetanilla
    - Tramitador
- FechaAltaExp
- AsientoAries

Parámetros de salida:

- **Numerador**: String numerador que se quiere usar en el expediente siendo creado. (RPS + año + Abreviatura procedimiento + secuencial)

4.5 Acceso directo a expediente

Servicios externos a la herramienta de tramitación TeJA podrá realizar un acceso directo a un expediente que se encuentre en el mismo. Esta acción se realizará a través de una petición GET al endpoint "/accesoDirectoServlet" con los siguientes parámetros:

Parámetros de entrada:

- **[requerido] callback_logout**: url que sustituirá en la pantalla del escritorio de tramitación la url de logout
- **[requerido] callback_inicio**: url a la que redirigirá el escritorio de tramitación al pulsar el botón de inicio
- **[opcional] puesto_trabajo_seleccionado:** código del puesto de trabajo con el que el usuario accederá al escritorio de tramitación
- **[requerido] csv_expediente:** código CSV del expediente al que se quiere acceder
- **[requerido] jndi:** código jndi de trewa
- **aceptaCondicinesExternas:** true/false para que solo se muestre una vez el código de conducta de acceso a la plataforma y la selección del puesto de trabajo.

5 Anexo

5.1 Activos transversales para el desarrollo de servicios externos

En el desarrollo de servicios externos (ya sean servicios core de tramitación o servicios de negocio) se han identificado un conjunto de actuaciones o acciones que serán utilizadas de manera recurrente en muchos servicios, por ejemplo: uso e instanciación del api-trewa, mapeo de objetos de base de datos a objetos de la capa frontal y manejador de excepciones.

Con el objetivo de simplificar, unificar los desarrollos y facilitar el mantenimiento futuro de dichos servicios, se han elaborado un conjunto de librerías comunes o transversales (utilizables tanto en los servicios de tramitación como los servicios de negocio) que dan cobertura y soporte a dichas acciones.

Dichas librerías utilizan como base el framework FKW-ADA, por lo que será obligatorio utilizarlo en caso de incluir las librerías. A continuación se detallarán cada uno de estos activos.

5.1.1 Adaptador de TrewaApi (Trewa-api-adaptador)

Librería que se encargará de centralizar la instanciación, desconexión, commit y rollback del API de Trewa. El objetivo de dicha librería es unificar y mejorar el control de la forma en la que se utiliza el api de Trewa evitando dejar conexiones abiertas que puedan ocasionar caídas o degradamiento en el rendimiento de los sistemas.

Dicha librería se compone de:

- **Una anotación** (@TrewaAPI), que será utilizada para anotar los métodos en los que se quiera hacer uso del api de Trewa.
- **Un aspecto o interceptor** (TrApiUIAspect) que se encarga de interceptar todas las llamadas a métodos que estén anotados con la anterior anotación y realizar la operaciones necesarias para obtener el api de Trewa.
- **Una clase adaptador** (TrewaApiAdapter) que será utilizada en los desarrollos para la obtención del api de Trewa y establecer los datos necesarios para su creación.

A nivel de utilización por parte de los equipos de desarrollo únicamente se hará uso del adaptador "TrewaApiAdapter" y la anotación "@TrewaAPI", pero conviene detallar como se ha implementado el interceptor para que estos equipos tengan el contexto completo.

En el proceso de ejecución del código, cualquier invocación a un método anotado con "@TrewaAPI" será interceptado por el aspecto "TrApiUIAspect" que se encargará de gestionar la instancia de la API de Trewa **para todo el hilo de ejecución del método**, abstrayendo al programador de la gestión de la API y simplificando el desarrollo únicamente a anotar el método y utilizar las operaciones proporcionadas por el API según convenga.

A continuación se detalla el comportamiento del interceptor:

1. **Obtención de los datos necesarios (JNDI, sistema, usuarioSistema).** Se obtendrán los datos correspondientes al "JNDI", "sistema" y "usuarioSistema" de los parámetros de entrada configurado en el método anotado con @TrewaAPI.

   Si no se configura en la anotación el nombre de los parámetros de entrada por defecto serán "jndi", "sistema" y "usuarioSistema". Si se desea especificar que un parámetro concreto "paramSistema" contiene el dato correspondiente al sistema, se deberá anotar el método *@TrewaApi (sistema="* paramSistema*")*

   Si el método no tiene configurado parámetros de entrada, los valores correspondiente a "jndi", "sistema" y "usuarioSistema" será obligatorio establecer previamente en el thread local del método del aspecto.

2. **Creación de una instancia del api de Trewa**. Se creará una instancia del api de Trewa a partir de los datos anteriores obtenidos de los parámetros de entrada del método o bien estableciéndolos en el thread local del método del aspecto.

   A continuación, se delegará el flujo de ejecución al método interceptado y se almacenará el objeto de respuesta en una variable local tipo Object para que el aspecto pueda devolverlo al método padre que invocó al método anotado.

3. **Finalización y cierre de conexiones.** Una vez finalizada correctamente la ejecución en el método interceptado se realizará el commit sobre la base de datos de Trewa mediante la instancia del api de Trewa inicializada y se devolverá el control y el objeto de respuesta al método padre que invocó al método anotado.

   En caso de excepción se realizará rollback sobre la base de datos de Trewa mediante la instancia del api de Trewa inicializada y se propagará la excepción hacia el método que haya invocado al método interceptado por el aspecto.

   Finalmente, se cerrará la conexión a la base de datos de Trewa mediante la instancia del api de Trewa inicializada y se eliminará la instancia del api de Trewa inicializada del thread local del método interceptado.

Componente	Nombre	Funcionalidad
Anotación	TrewaAPI	Anotación *"@TrewaAPI"* que permite identificar que clases o métodos interactúan con la API de Trewa. Se compone de dos aatributos: • String jndi() default "jndi" → Atributo que permite indicar el nombre del parámetro de entrada que contiene el jndi para el api de Trewa, si no se configura, por defecto, el nombre del parámetro de entrada al método que contiene el jndi para el api de Trewa será "jndi". • String sistema() default "sistema" → Atributo que permite indicar el nombre del parámetro de entrada que contiene el código del sistema para el api de Trewa, si no se configura, por defecto, el nombre del parámetro de entrada al método que contiene el sistema para el api de Trewa será "sistema".

		• String usuarioSistema() default "usuarioSistema" → Atributo que permite indicar el nombre del parámetro de entrada que contiene el código del usuario tramitador para el api de Trewa, si no se configura, por defecto, el nombre del parámetro de entrada al método que contiene usuario para el api de Trewa será "usuarioSistema". • boolean autoCommit(). Atributo que permite indicar el valor que tendrá el autoCommit en el api de Trewa. Por defecto es valor será false.
Interceptor (aspecto)	TrApiUIAspect	Aspecto que intercepta todas las ejecuciones de los métodos públicos de una clase anotada con @TrewaAPI. Se compone de un método "***trApiUi()***" cuya funcionalidad es el control total del apiTrewa (apertura, commit y cierre de conexiones). Para ello obtiene el JNDI de Trewa a partir de los parámetros de entrada al método interceptado o del thread local de este método del aspecto. También obtiene el "sistema" y "usuarioSistema" (código del usuario tramitador) de forma análoga al JNDI. Con dichos datos instanciará el API de Trewa, delegará el flujo de ejecución al método interceptado y al finalizar la ejecución del método realizará commit o rollback del API de Trewa según sea el resultado de la ejecución del método interceptado. Por último, finalizará la conexión con el API de Trewa y devolverá el control al punto de ejecución desde donde se invocó al método anotado con @TrewaAPI.
Adaptador Trewa	TrewaApiAdapter	Clase estática que proporciona la funcionalidad para establecer los datos (en el thread local de un método) necesarios para instanciar el api de Trewa y obtenerla. Se compone de los siguientes métodos: **Api Trewa** • getTrApiUI: Método que permite obtener el objeto TrAPIUI establecido en el thread local de un método • setTrApiUI: Método que permite establecer un objeto TrAPIUI en el thread local de un método • eliminarTrApiUI: Método que permite eliminar el objeto TrAPIUI establecido en el thread local de un método **JNDI** • setJNDI: Método que permite establecer un objeto String con el JNDI de Trewa en el thread local de un método • getJNDI: Método que permite obtener objeto String con el JNDI de Trewa establecido en el thread local de un método • eliminarJNDI: Método que permite eliminar el objeto String con el JNDI de Trewa establecido en el thread local de un método **Sistema** • setSistema: Método que permite establecer un objeto String con el código del sistema de Trewa en el thread local de un método • getSistema: Método que permite obtener objeto String con el código del sistema de Trewa establecido en el thread local de un método • eliminarSistema: Método que permite eliminar el objeto String con el código del sistema de Trewa establecido en el thread local de un método **UsuarioSistema** • setUsuarioSistema: Método que permite establecer un objeto String con el código del usuario Tramitador de Trewa en el thread local de un método

		• getSistema: Método que permite obtener objeto String con el código del usuario Tramitador de Trewa establecido en el thread local de un método • eliminarSistema: Método que permite eliminar el objeto String con el código del usuario Tramitador de Trewa establecido en el thread local de un método **AutoCommit** • setAutoCommit: Método que permite establecer un objeto boolean con el valor de autoCommit de Trewa en el thread local de un método • getAutoCommit: Método que permite obtener un objeto boolean con el valor de autoCommit de Trewa en el thread local de un método

Dependencia maven

```
<dependency>
    <groupId>es.juntadeandalucia.ada</groupId>
    <artifactId>trewa-api-adaptador</artifactId>
    <version>${trewa-api-adaptador-version}</version>
</dependency>
```

Ejemplo 1. JNDI, sistema y usuarioSistema como parámetro de entrada. Uso del adaptador en un método que contiene los datos necesarios para crear el api de Trewa como parámetros de entrada.

```
@TrewaAPI
public List< FasesVO > getProcedimientos (String jndi, String sistema, String usuarioSistema, String abreviatura) throws TrException {
    TrAPIUI trApiUI = TrewaApiAdapter.getTrApiUI();
    ClausulaWhere cw = new ClausulaWhere();
    cw.addExpresion(TrDefProcedimiento.CAMPO_ABREVIATURA, OperadorWhere.OP_IGUAL, abreviatura);

    TrDefProcedimiento[] trDefProcedimientos = trApiUI.obtenerDefProcedimientosDefinidos(null, cw, null);
….
}
```

Al llamar a "*TrewaApiAdapter.getTrApiUI();",* crear el api de Trewa con los datos jndi, sistema y usuarioSistema correspondientes a los parámetros del método "getProcedimientos*(…)*"

Ejemplo 2. JNDI, sistema y usuarioSistema como parámetro de entrada con diferente nombre. Uso del adaptador en un método que contiene los datos necesarios para crear el api de Trewa como parámetros de entrada, pero con diferentes nombres a los utilizados por defecto

```
@TrewaAPI(jndi="miJNDI", sistema="miSistema", usuarioSistema="codUsuarioTramitador")
public List<FasesVO> getProcedimientos (String miJNDI, String miSistema, String codUsuarioTramitador, String abreviatura) throws TrException {
    TrAPIUI trApiUI = TrewaApiAdapter.getTrApiUI();
    ClausulaWhere cw = new ClausulaWhere();
    cw.addExpresion(TrDefProcedimiento.CAMPO_ABREVIATURA, OperadorWhere.OP_IGUAL, abreviatura);

    TrDefProcedimiento[] trDefProcedimientos = trApiUI.obtenerDefProcedimientosDefinidos(null, cw, null);
….
}
```

Ejemplo 3. Indicar el dato autoCommit a true. Uso del adaptador en un método que contiene los datos necesarios para crear el api de Trewa como parámetros de entrada indicando el autoCommit que por defecto es "false" a valor "true".

```
@TrewaAPI(autoCommit=true)
public List< FasesVO > getProcedimientos (String jndi, String sistema, String usuarioSistema, String abreviatura) throws TrException {
…
}
```

Ejemplo 4. Sin parámetros de entrada para crear el api. Uso del adaptador en un método que no tiene ningún parámetro de entrada donde se indiquen los valores necesarios para crear el api de Trewa (JNDI, sistema y usuarioSistema). En estos casos es necesario establecer previamente en el Treadlocal del método dichos datos.

```
@TrewaAPI
public List< FasesVO > metodoPrevioAgetProcedimientos () throws TrException {
        TrewaApiAdapter.setJNDI("TrewaCHIE")
        TrewaApiAdapter.setUsuarioSistema("UsuarioPruebas")
        TrewaApiAdapter.setSistema("CHIE")
        getFases();
}

@TrewaAPI
public List< FasesVO > getProcedimientos(String abreviatura) throws TrException {
    TrAPIUI trApiUI = TrewaApiAdapter.getTrApiUI();
    ClausulaWhere cw = new ClausulaWhere();
    cw.addExpresion(TrDefProcedimiento.CAMPO_ABREVIATURA, OperadorWhere.OP_IGUAL, abreviatura);

    TrDefProcedimiento[] trDefProcedimientos = trApiUI.obtenerDefProcedimientosDefinidos(null, cw, null);
…
}
```

Para este tipo de casos, para evitar duplicar el código correspondiente a la obtención y establecimiento de los datos necesarios para crear el api, se recomienda establecerlos mediante alguna forma que permita reutilización de código (métodos de utilidades, métodos privados, interceptores, etc.)

Para la configuración de las diferentes conexiones a bases de datos de trewa de deberá incluir un fichero de configuración adicional con los parámetros de configuración deseados. Se recomienda añadir esta configuración con el siguiente comando de ejecución:
-Dspring.config.additional-location=[ubicacion del fichero]/[nombre del fichero].yml

El contenido del fichero **debe** seguir un formato como el siguiente:

```
gaia:
  data:
    datasources:
      - name: TrewaXXXX
        url: [url]
        username: [username]
        password: [password]
        connectionProperties:
          blockingTimeoutMillis: '5000'
          idleTimeoutMinutes: '30'
        pool:
          minPoolSize: 1
          initialSize: 0
          maxPoolSize: 10
        validation:
          checkValidConnectionSql: select 1 from dual
          validateOnMatch: false
          backgroundValidation: true
          backgroundValidationMillis: 60000
        statement:
          preparedStatementCacheSize: 32
          sharePreparedStatements: false
      - name: TrewaXXXX
        url: [url]
```

```
    username: [username]
    password: [password]
    connectionProperties:
     blockingTimeoutMillis: '5000'
     idleTimeoutMinutes: '30'
    pool:
     minPoolSize: 1
     initialSize: 0
     maxPoolSize: 10
    validation:
     checkValidConnectionSql: select 1 from dual
     validateOnMatch: false
     backgroundValidation: true
     backgroundValidationMillis: 60000
    statement:
     preparedStatementCacheSize: 32
     sharePreparedStatements: false
.
.
.
```

Los parámetros que en el ejemplo anterior tienen un valor especificado son los **recomendados** para cada uno de ellos, pero pueden ser modificados a conveniencia del integrador. A continuación, dejamos la lista completa de parámetros que pueden ser configurados

Etiqueta	**Descripción**
name	Nombre que identifica a la conexión a trewa. No debe repetirse en toda la configuración de datasources.
driver	Tipo de componente software que actúa como un traductor o puente entre una aplicación y un sistema de gestión de bases de datos. Un valor común es “oracle.jdbc.driver.OracleDriver”.
url	Url jdbc de la base de datos. En entornos cloud deberá contener la url completa para funcionar correctamente (por ejemplo, “jdbc:oracle:thin:@prucom.chap.junta-andalucia.es:1531:PRUCOM”).
username	Usuario a utilizar en la conexión a la base de datos.
password	Contraseña del usuario.
connectionProperties.blockingTimeoutMillis	Tiempo máximo en milisegundos que una petición de conexión espera en cola cuando el pool está lleno antes de lanzar un error.
connectionProperties.idleTimeoutMinutes	Minutos que una conexión puede permanecer inactiva en el pool antes de ser cerrada/liberada.
pool.minPoolSize	Número mínimo de conexiones que el pool mantiene abiertas de forma permanente.
pool.initialSize	Número de conexiones que se crean inicialmente al arrancar el pool.
pool.maxPoolSize	Número máximo de conexiones simultáneas permitidas en el pool.
validation.checkValidConnectionSql	Sentencia SQL de validación usada para comprobar si una conexión sigue siendo válida (por ejemplo, SELECT 1).
validation.testOnBorrow	Indica si se valida la conexión (usando el SQL de validación) cada vez que se obtiene del pool. Su valor por defecto en caso de no especificarse es true.
validation.backgroundValidation	Indica si se realiza validación periódica en segundo plano de las conexiones del pool.
validation.backgroundValidationMillis	Intervalo en milisegundos entre validaciones de conexiones cuando está activada la validación en segundo plano.

Trewa-api-adaptador hace uso de DataSource, PoolProperties y ContextResource de Apache, dependencias que necesita para el correcto funcionamiento y configuración de la api de trewa.

5.1.2 Mapeador común (comun-model-mapper)

Librería que se encargará de centralizar los mapeos de objetos entre la capa de servicios y la capa de negocio. El objetivo de dicha librería es unificar y simplificar la forma en la que se mapean dichos objetos entre ambas capas, además de evitar de manera transparente el mapeo de los objetos tipo "colección" cuyo proxy no esté inicializado en el momento de la conversión (Por ejemplo, listas con fetch = FetchType.LAZY), evitando así el problema de rendimiento conocido como N +1 consultas.

Como parámetros de entrada se podrá proporcionar un objeto individual o una lista de objetos, especificando el tipo de objeto al cual se desea convertir el objeto origen. Como resultado de la conversión se obtendrá un objeto del tipo de destino especificado, inicializado con los datos de los campos del objeto origen que coincidan directamente con los campos del tipo de objeto destino y con los datos de los campos del objeto origen trasladados al objeto de destino según el mapeo que se haya implementado específicamente entre los dos tipos de objetos.

Dicha librería se compone de:

- **Una anotación** (@ModelMapperMapping), que será utilizada por aquellas clases donde se desee implementar un tipo especifico de conversión entre dos objetos concretos.
- **Un servicio** (ModelMapperService) que implementa los métodos con los parámetros necesarios para realizar el proceso de conversión de un tipo de objeto origen en un tipo de objeto destino.

  Se utilizará este servicio para convertir objetos VO a Entidades y viceversa. También se utilizará para convertir objetos de Trewa a VO y viceversa.

Componente	Nombre	Funcionalidad
Anotación	ModelMapperMapping	Anotación "@ModelMapperMapping" que permite identificar las clases que implementarán una conversión específica entre dos objetos origen y destino.
Servicio	ModelMapperService	Servicio con la lógica para realizar las conversiones de un objeto origen a otro destino. Se compone de los siguientes métodos. • Inicializar: Este método se ejecutará al terminar de construir el servicio y permite establecer dicho servicio de mapeo los tipos de mapeo específicos entre objetos concretos y las condiciones que el mapeador tiene que tener en cuenta durante el proceso de conversión para ignorar proxies de Hibernate no inicializados. Para ello examinará en el paquete base todas las clases que tenga la anotación anterior, configurando cada uno de los mapeos específicos encontrados. • convertirAClaseDestino: Método que permite convertir un objeto individual del tipo de dato origen al tipo de dato de destino indicado como parámetro de entrada al método. • convertirListaAClaseDestino: Método que permite convertir una lista de objetos del tipo de dato origen al tipo de dato de destino indicado como parámetro de entrada al método

Dependencia maven

```
<dependency>
        <groupId>es.juntadeandalucia.ada</groupId>
```

```
            <artifactId>comun-model-mapper</artifactId>
            <version>${comun-model-mapper-version}</version>
</dependency>
```

Ejemplo de uso

- Establecer el paquete base donde se buscarán los mapeos específicos, por defecto "${app.modelmapper.mapping.basepackage:es.juntadeandalucia.ada}"

- Crear las correspondientes clases específicas de mapeo si fuera necesario. Por ejemplo.

```
package es.juntadeandalucia.ada.teja.cache.svc.service.mapper.mapperdefinition;
…
@ModelMapperMapping
public class TrDefProcedimientoToProcedimientoVO {

  private TrDefProcedimientoToProcedimientoVO () {
            throw new IllegalStateException("Clase de utilidades.");
  }

  public static void configurarMappings(ModelMapper modelMapper) {
            modelMapper.typeMap(TrDefProcedimiento.class, ProcedimientoVO.class).addMapping(
                        TrDefProcedimiento -> TrDefProcedimiento.getREFDEFPROC(),
                        ProcedimientoVO::setId
            );
            modelMapper.typeMap(TrDefProcedimiento.class, ProcedimientoVO.class).addMapping(
                        TrDefProcedimiento -> TrDefProcedimiento.getSTMA().getREFSTMA(),
                        ProcedimientoVO::setIdSistema
            );
…
…
}
```

- Llamar a los métodos para convertir de objeto origen a destino "convertirAClaseDestino" o "convertirListaAClaseDestino"

```
  @TrewaAPI
  @Cacheable(value = "procedimientos")
  public List<ProcedimientoVO> getProcedimientos(List<Long> sistemas) throws TrException {
            TrAPIUI trApiUI = TrewaApiAdapter.getTrApiUI();
            ClausulaWhere cw = new ClausulaWhere();

            if(sistemas != null && !sistemas.isEmpty()) {
                        cw.addExpresionIn(TrDefProcedimiento.CAMPO_REFSTMA, OperadorWhere.OP_IN, sistemas);
            }

            TrDefProcedimiento[] listaProcedimientos = trApiUI.obtenerDefProcedimientosDefinidos(null, cw, null);
            if(ArrayUtils.isNotEmpty(listaProcedimientos)) {
                        return ModelMapperService.convertirListaAClaseDestino(Arrays.asList(listaProcedimientos),
ProcedimientoVO.class);
            }
            return Collections.emptyList();
  }
```

5.1.3 Gestor de excepciones común (comun-excepciones)

Librería que se encargará de centralizar el manejo de excepciones de negocio permitiendo personalizarlas con mensajes para la interfaz de usuario. El objetivo de dicha librería es unificar y mejorar el tratamiento en el manejo de las excepciones que trae por defecto el framework FWK-ADA añadiendo además del código de error un mensaje de texto informativo con la descripción del error (que podrá utilizarse para mostrar mensajes más funcionalidades y adaptados al lenguaje de usuario en la capa frontal). Además cuando ocurre algún problema se encarga de mostrar la traza completa del error en el log del sistema.

Dicha librería se compone de:

- **Un nuevo mensaje de error** (UIMessageExceptionErrorCode) que contiene un atributo de texto con el mensaje de error personalizado.
- **Una nueva excepción** personalizada (UIMessageException) que incluye el mensaje de error anterior.
- **Un manejador de excepciones** (UIMessageExceptionHandler) que se encargará de manejar la excepción con el mensaje personalizado y mostrar la traza completa del error en el log del sistema.

Cuando se lance una excepción de negocio de tipo UIMessageException será interceptada por un handler personalizado que compondrá un mensaje de respuesta para la request realizada al endpoint.

Dependencia maven

```
<dependency>
        <groupId>es.juntadeandalucia.ada</groupId>
        <artifactId>comun-excepciones</artifactId>
        <version>${comun-excepciones-version}</version>
</dependency>
```

Ejemplo de uso

- Creación de una clase enumerado con los mensajes personalizados

```
@AllArgsConstructor
@Getter
public enum UIMessageExceptionErrorCodeImpl implements UIMessageExceptionErrorCode {

        CUSTOM_EXCEPTION("Ejemplo de excepción personalizada"),
        UNAUTHORIZED_EXCEPTION("Usuario no autenticado"),
        PROCEDIMIENTO_NO_ENCONTRADO_POR_ABREVIATURA("No se encontró el procedimiento para la abreviatura especificada"),
        FASES_NO_ENCONTRADAS_POR_PROCEDIMIENTO("No se encontró las fases relacionadas al Procedimiento especificado"),
        PROCEDIMIENTO_NO_ENCONTRADO("La abreviatura informada no corresponde con un Procedimiento");

        private final String reasonPhrase;

}
```

- Uso de dichos mensajes personalizados cuando ocurren errores.

```
@Override
@TrewaAPI
public ProcedimientoVO getProcedimientoByAbreviatura(String abreviatura) throws TrException {
        TrAPIUI trApiUI = TrewaApiAdapter.getTrApiUI();
        ClausulaWhere cw = new ClausulaWhere();
        cw.addExpresion(TrDefProcedimiento.CAMPO_ABREVIATURA, OperadorWhere.OP_IGUAL, abreviatura);

         TrDefProcedimiento[] trDefProcedimientos = trApiUI.obtenerDefProcedimientosDefinidos(null, cw, null);
         if (ArrayUtils.isEmpty(trDefProcedimientos)) {
                throw new UIMessageException (HttpStatus.NOT_FOUND,
                UIMessageExceptionErrorCodeImpl.PROCEDIMIENTO_NO_ENCONTRADO_POR_ABREVIATURA,
                        "No se ha encontrado el procedimiento para la abreviatura : " + abreviatura, null);
          }
        return ModelMapperService.convertirAClaseDestino(trDefProcedimientos[0], ProcedimientoVO.class);
}
```

Finalmente, si desea solicitar acceso a las librerías, por favor, póngase en contacto a través del siguiente buzón de correo electrónico: otgp.svad.ada@juntadeandalucia.es

5.2 Glosario de términos

Acrónimo	Descripción
ADA	Agencia Digital de Andalucía
@firma	Es la plataforma corporativa de autenticación y firma basada en certificados electrónicos para los procedimientos administrativos, trámites y servicios de la Administración de la Junta de Andalucía
MSD	Modelo de Servicios Digitales
Notific@	Sistema para realizar el envío y la gestión de notificaciones electrónicas fehacientes, con generación de evidencias comprobables de la entrega por el emisor y la recepción por el destinatario, conforme a la normativa vigente del procedimiento administrativo común.
Port@firma	Port@firmas es la herramienta destinada a facilitar a los órganos y unidades administrativas el uso de la firma electrónica (basada en certificado electrónico) de documentos procedentes de diferentes sistemas de información.
TeJA	Tramitador de expedientes de la Junta de Andalucía
Trew@	Motor de tramitación de la Junta de Andalucía