**SDD**

Documento de Diseño de la Solución

|  |  |  |
| --- | --- | --- |
| **Nombre del Proceso** | Impacto Autónomo |  |
| **ID del Proceso** | TR890A |  |
| **Área** | Traballo |  |
| **Subárea** | Conselleria de Empego e Igualdade |  |

|  |  |
| --- | --- |
| **Versión** |  |
| **Fecha** | 05/06/2023 |

**TABLA DE CONTENIDOS**

[1. Introducción 4](#_Toc132188620)

[1.1 Propósito del documento 4](#_Toc132188621)

[1.2 Referencia a documentos del proyecto 4](#_Toc132188622)

[2. Resumen del proceso 5](#_Toc132188623)

[2.1 Diseño de Alto Nivel (HLD) 5](#_Toc132188624)

[2.2 Descripción del proceso 7](#_Toc132188625)

[3. Modelo de la información 24](#_Toc132188626)

[4. Procedimiento 25](#_Toc132188627)

[4.1 Excepciones de Negocio 25](#_Toc132188628)

[4.2 Excepciones de Sistema 26](#_Toc132188629)

[4.3 Control de Estados 26](#_Toc132188630)

[4.4 Planificación del Proceso 27](#_Toc132188631)

[4.5 Escalabilidad 27](#_Toc132188632)

[4.6 Sistema de Alerta 27](#_Toc132188633)

[5. Tratamiento de los datos 29](#_Toc132188634)

[5.1 Origen de la información 29](#_Toc132188635)

[5.2 Almacenamiento de los Datos 29](#_Toc132188636)

[5.3 Seguridad 29](#_Toc132188637)

[5.4 Mantenimiento 30](#_Toc132188638)

[6. Consideraciones 31](#_Toc132188639)

[7. Anexo I: Glosario 32](#_Toc132188640)

**CONTROL DEL DOCUMENTO**

Plantilla

El historial de versiones realizado en este documento es el siguiente.

|  |  |  |  |
| --- | --- | --- | --- |
| Versión | Fecha | Autor | Cambios en la versión |
| 1.0 | 28/02/2023 | Carlos Rueda  Camila Rodríguez | Versión Inicial |

Historial de versiones

Los cambios realizados en este documento son los siguientes.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Versión | Fecha | Autor | Aprobado por | Motivo del cambio |
| 1.0 | 05/06/2023 | Alicia Fernández |  | Versión inicial |

|  |
| --- |
| INSTRUCCIONES - Eliminar esta tabla cuando la sección esté terminada |
| La columna “Aprobado por” se trata del responsable de calidad responsable de validar el contenido del documento técnico de la solución SDD. |

1. Introducción

Esta sección describe el diseño a alto nivel del proceso automatizado.

Propósito del documento

El propósito del documento Solution Design Document (SDD) es describir la solución técnica desarrollada para el cumplimiento de los requisitos descritos en el Process Design Document (PDD).

Se incluye cualquier prerrequisito técnico o consideración necesaria para la implementación, operación y mantenimiento del proceso. Es un documento dinámico que se genera a medida que se construye la solución y se termina antes del despliegue del proceso en el entorno de producción.

En este documento se referirá al paquete de automatización (“La Solución” o “Solución”), que representa el Orchestrator de UiPath, las funciones y los componentes, así como a los componentes técnicos secundarios (p.ej. colas, credenciales, plantillas, bases de datos, etc.) usados para completar el proceso a automatizar.

Referencia a documentos del proyecto

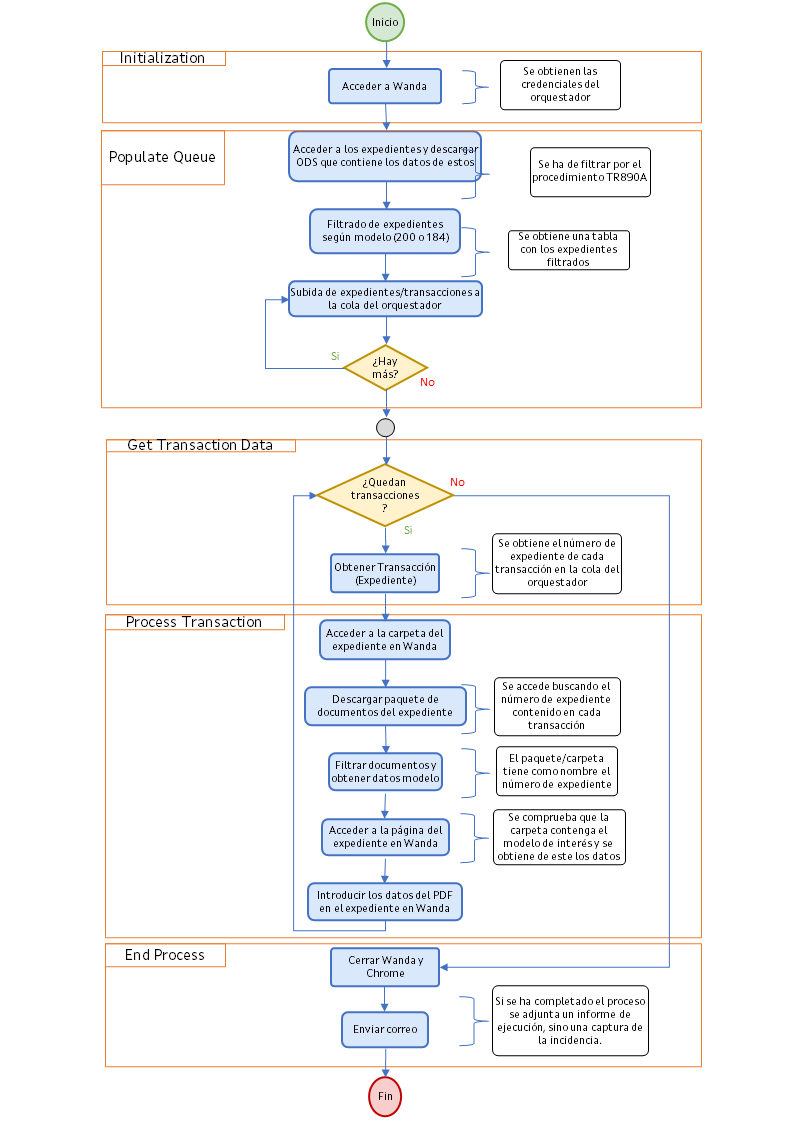
Los siguientes archivos deberán leerse conjuntamente con este documento.

|  |  |  |
| --- | --- | --- |
| Título | Versión | Archivo |
| Documentación plantilla Framework | Sin especificar | “REFramework Documentation-EN.pdf” |
| Documento de Definición del proceso (PDD) del Impacto de Autónomo | Sin especificar | “AMTEGA\_PDD\_ImpactoAutonomo\_\*.docx” |
| Documento Modelo 200 del expediente | Sin especificar | “MOD 200.pdf” o similar |
| Documento Modelo 184 del expediente | Sin especificar | “MOD 184.pdf” o similar |

1. Resumen del proceso

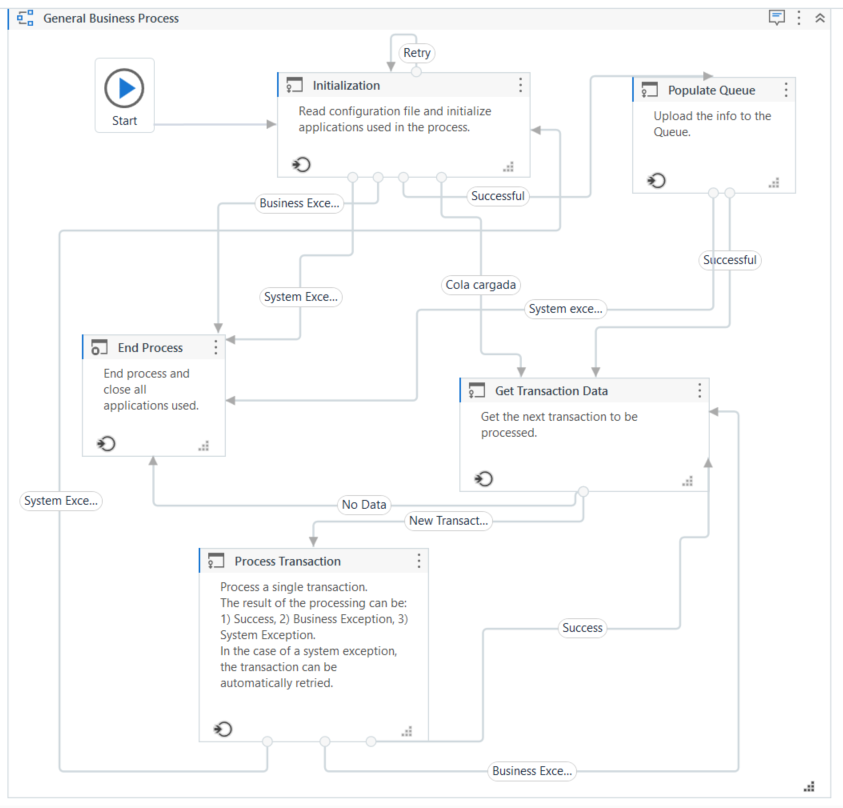
Esta sección describe el diseño a alto nivel del proceso a automatizar.

Diseño de Alto Nivel (HLD)



Descripción del proceso

La solución se ha desarrollado unificando las plantillas del Dispatcher y el Performer.



Las plantillas de UiPath que se han usado para desarrollar la solución del proceso automatizado son:

* ***\*\OneDrive - EY\Desarrollo\03. Plantillas UiPath\Plantilla\_Dispacher***
* **\*\OneDrive - EY\Desarrollo\03. Plantillas UiPath\Plantilla\_Performer**

Este proceso está estructurado en cinco bloques o máquinas de estado, el funcionamiento de forma resumida de cada uno es el siguiente:

* *Initialization*: En este bloque, de forma resumida, se cierran las aplicaciones que estuviesen abierta de antes de iniciar la ejecución, se comprueba si es la primera ejecución y en ese caso establece los ajustes e invoca las funciones/componentes necesarios. Fuera de esta comprobación se abre el navegador y se inicia sesión en Wanda.
* *Populate Queue: S*e buscan los expedientes del procedimiento indicado (TR890A) y se descarga un ODS con el cual se pueden filtrar los expedientes según el modelo (200 o 184) de interés. Se añaden las transacciones (compuestas por los expedientes filtrados y datos asociados a estos) a la cola del orquestador.
* *Get Transaction Data*: Se obtienen los datos, subidos anteriormente en la cola del orquestador.
* *Process Transaction*: De forma resumida, en este bloque se buscan aquellos expedientes ya filtrados y se descarga la carpeta asociada a estos. Una vez descargadas, se recorren estas carpetas en busca del documento que corresponda al modelo 200 o 184 (según se indique) y se extraen los datos de interés de estos. Posteriormente, se editan los expedientes introduciendo estos datos en los campos específicos y se guarda la información. Además, se va pintando determinada información de la transacción en un archivo .xlsx (Informe\_ejecucion.xlsx), incluyendo el estado de esta (como, por ejemplo, “Success” o “Failed”). En el caso de que ocurra una incidencia de sistema se reintentará la transacción el número de veces establecido en el “Config” o en el orquestador. Si falla el número máximo de intentos, solo pintará en el Informe de ejecución el estado del último intento.
* *End Process*: Se cierra la sesión en Wanda y el navegador usado. Se convierte el informe de ejecución de .xlsx a .ods y se manda el correo al destinatario especificado. El correo mandado dependerá de cómo haya ido el proceso. Si se ha completado el proceso el correo adjunta el informe de ejecución y sino una captura de pantalla de la incidencia que surgió.

NOTA: Todos los ficheros que se guarden de forma local (han sido descargados) se borrarán tras haber sido utilizados por la actividad que los necesitaba. De esta forma se busca evitar el almacenamiento de ficheros ya usados/procesados.

* + 1. Preparación del Entorno

En primer lugar, hace falta crear un fichero que contenga los datos que va a usar el robot, como por ejemplo el nombre de la carpeta del orquestador que contiene las credenciales de inicio de sesión de Wanda o la ruta local donde se va a guardar el informe de ejecución. Este fichero Excel se llama *Config* y se encuentra dentro de la carpeta *Data*­. Por cada dato, representado en una fila, se establecerá un nombre, un valor y una descripción.

Como se va a trabajar con el orquestador hace falta realizar unos pasos previos:

* Crear una carpeta de trabajo
  + Añadir un “Asset” de tipo credencial, donde se guarda el nombre del usuario y la contraseña para el inicio de sesión en Wanda.
  + Añadir una cola o “Queue”, darle un nombre y establecer el número de reintentos (por defecto es uno).

Nota: Los nombres/rutas de estos elementos que se han creado se indican en el Config.

* + - 1. Dispatcher

Una vez realizado esto, se pasa a UiPath donde se desarrolla el Dispatcher. Partiendo de la plantilla homónima se va añadiendo las funciones o componentes específicos de este proceso.

Todos los componentes/funciones implementadas específicamente para este proceso presentan un *Try Catch* que en el caso de fallar lanzarán un *throw* con una incidencia de sistema.

Como se comentó en el apartado anterior, este se compone de tres bloques:

* + - * 1. Initialization

El bloque de la inicialización se encarga de leer el archivo que contiene la configuración (*Config.xlxs*) e iniciar las aplicaciones usadas en el proceso.

* **Componente:** **comp\_CloseAllApplications**

Se cierran las aplicaciones previas al proceso, siendo en este caso el navegador “Chrome”. Primero se intentará cerrar mediante la actividad “Close application”, en caso de que no se cierre tras esperar un determinado “Delay”, se procederá con un “Kill Process”.

Se comprueba si es la primera ejecución del proceso comprobando que la variable *Config* de tipo *Dictionary<String, Object>* esté vacía. Si es así:

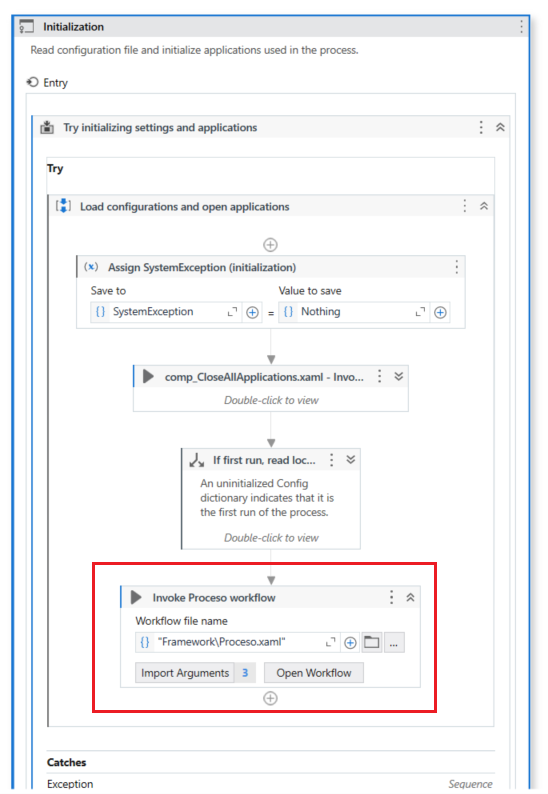
* **Framework:** ***InitAllSettings***

Lee el Config.xlsx y rellenará la variable de este nombre. Además, obtendrá del orquestador los valores del “Asset”.

* **Componente: *comp\_CleanQueue***

Se comprueba que la cola del orquestador (nombre pasado por el *Config*) que se va a utilizar en el proceso esté vacía y sino se borran los elementos de esta.

En último lugar, se añaden las funciones o componentes específicos (esquematizados en el HDL) del proceso dentro del flujo de trabajo “Proceso”.



Dentro de “Proceso”, por lo tanto, estarán por orden:

* **Componente:** **comp\_WANDA\_castellano\_Login**

1. Obtener las credenciales del orquestador, para lo cual hace falta el nombre de la carpeta del orquestador y el nombre del *asset* de las credenciales, los cuales se pasan como argumentos (*in\_strWandaOrchCredFolder* y *in\_strWandaOrchCredName*, respectivamente) con los valores definidos en el *Config*.
2. Abrir el navegador (*Chrome*) y acceder a la página de *login* de Wanda, donde tras maximizar la ventana comprueba que exista el mensaje de bienvenida a Wanda, en caso de lo encuentre se cierra sesión y se vuelve a iniciar escribiendo las credenciales correspondientes. Esto dará paso a la página de inicio de Wanda.
3. En caso de las credenciales sean inválidas se lanzará un *throw* con una incidencia de sistema indicando esto. Si las credenciales son válidas pero el usuario no tiene permisos en la aplicación se lanzará una incidencia de negocio.

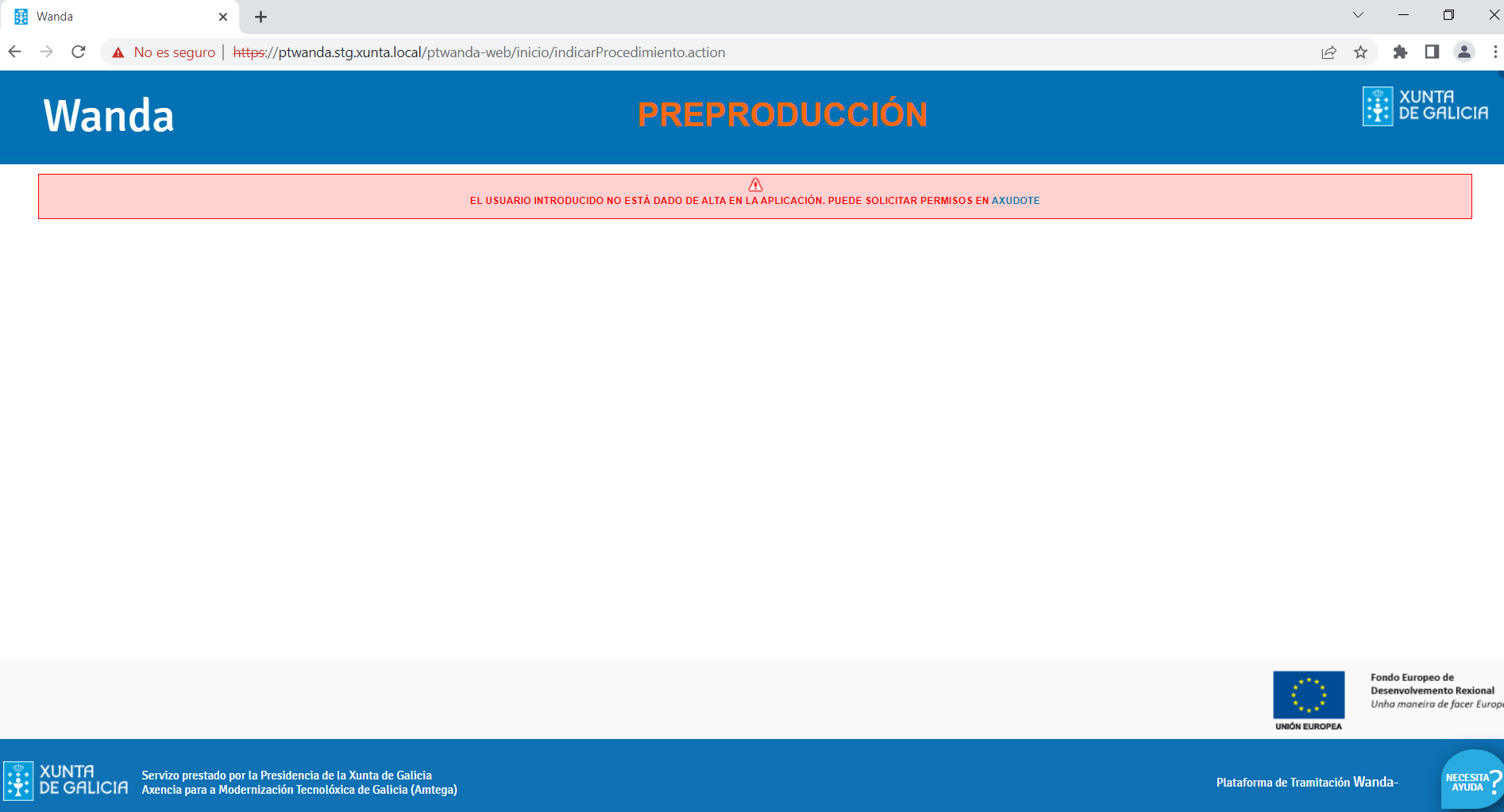
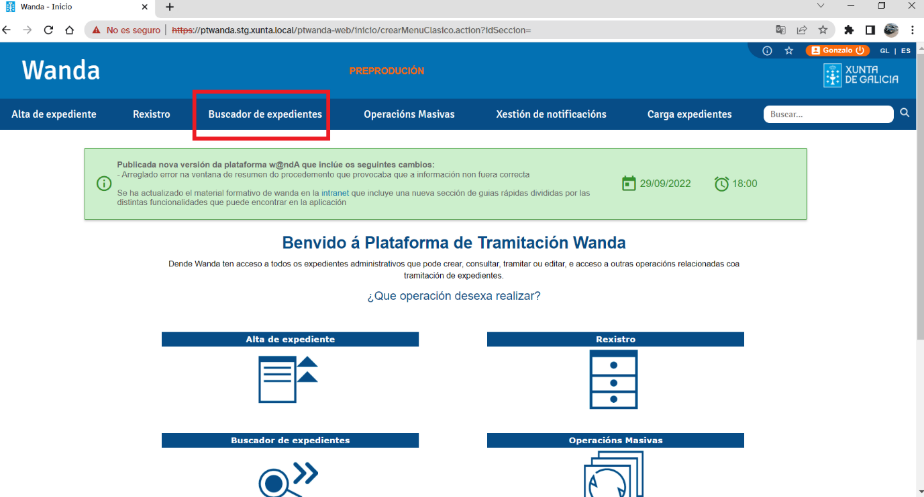


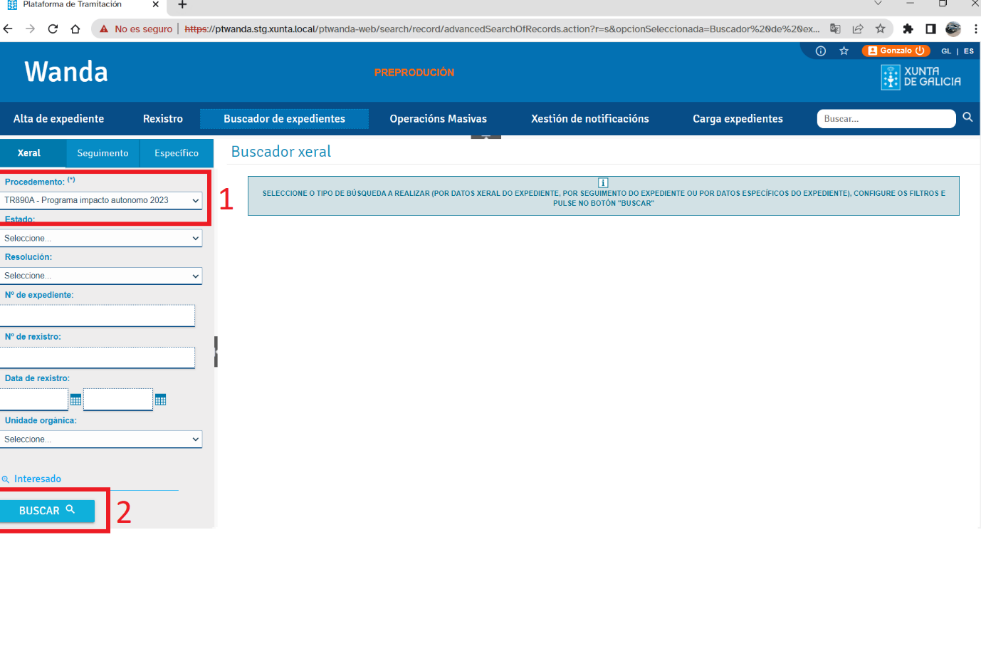
Imagen 1: Ejemplo credenciales válidas, pero sin acceso a la aplicación

* **Función: func\_WANDA\_DescargaODSExpedientes**

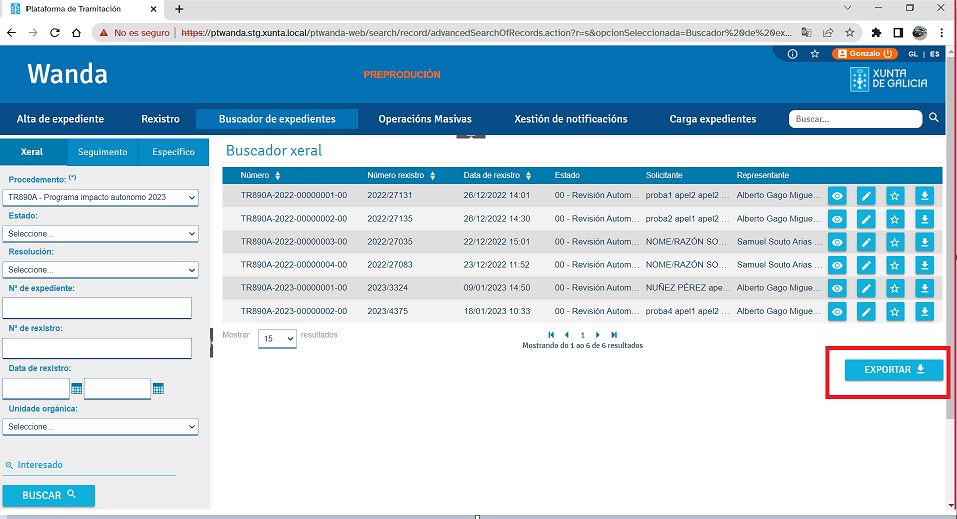
1. Desde la página de Wanda se clica en el botón *Buscador de expedientes*.



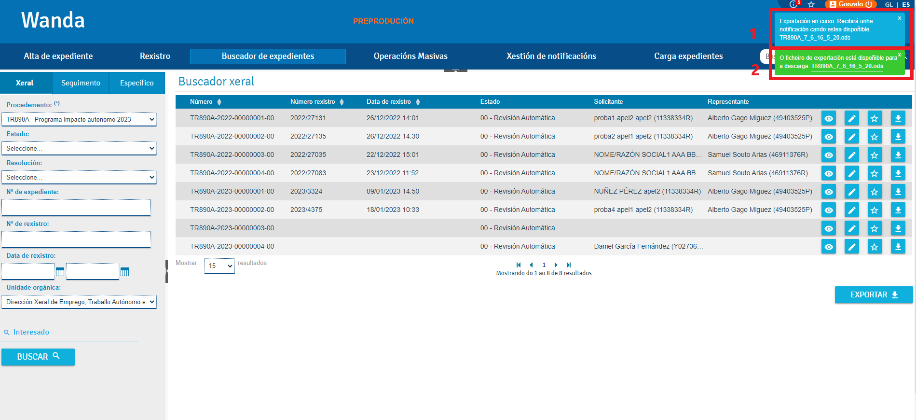
1. Se selecciona en “Procedimiento” el procedimiento de interés y se clica en el botón de buscar. Este se define mediante el argumento *in\_strProcedimientoWanda,* cuyo valor se establece en el *Config.* Se incluye una actividad *Element Exists* para comprobar que aparezca algún expediente como resultado de la búsqueda, en caso de que no se lanzará un *throw* con una incidencia de sistema.



1. Si se encuentran resultados, se clicará en *exportar* y se seleccionará la opción de *ODS*.



1. Posteriormente debe aparecer un elemento en la pantalla (se comprueba mediante la actividad *On Element Appear*) del cual se extraerá mediante *Get Text* el nombre del fichero que se va a descargar, denominado como *vStrNombreODS*. Debajo de este primer elemento aparece uno inmediatamente abajo en el cual aparece el mismo nombre del fichero, donde haciendo clic se inicia la descarga.



1. Como el archivo tarda en descargarse, se ha introducido un *Do While* el cual irá aumentando el *delay* un segundo hasta que encuentre que exista el archivo .ods.
2. Por último, se mueve el .ods de la carpeta de descarga (se pasa como argumento *in\_strCarpetaDescargas*, definido el valor en el *Config*) a la carpeta donde se vaya a trabajar (Data\Temp). El nombre del fichero se cambia al nombre del procedimiento (*in\_strProcedimientoWanda)*.

* **Función: func\_FiltradoExpedienteXModelo**

1. Primero se convierte el fichero .ods a .xlsx. Esto se hace invocando el componente: *comp\_castellano\_LibreOfficeConverter*. El cual tiene como argumento de salida la ruta donde se guardará el fichero (*out\_strOutputFilePath*).
2. Posteriormente se lee el fichero mediante la actividad *Read Range* y se filtra la tabla según el tipo de modelo de interés, establecido en el *Config* y pasado mediante un argumento de entrada (*in\_strTipoModelo*). Esto da como resultado una tabla almacenada en el argumento de salida *out\_dtListadoExpedientes*.

El proceso tendrá como argumento de salida *out\_DtQueueElements*, que tiene como valor la tabla filtrada obtenida en *func\_FiltradoExpedienteXModelo.*

Las transiciones que presenta la inicialización del Dispatcher son las siguientes:

* *Retry*: Se reintenta la inicialización en caso de que se haya producido una incidencia de sistema y de que el número de reintentos sea máximo tres.
* *Business Exception*: Si falla por una incidencia de negocio el destino será el bloque de *End Process*.
* *System Exception*: Si hay una incidencia de sistema y se ha reintentado ya tres veces la inicialización se pasa al *End Process.*
* *Successful*: Si no ha habido ninguna incidencia el siguiente bloque será el de *Add Queue Item*.
  + - * 1. Add Queue Item

Este bloque se encarga de introducir o actualizar información en la cola del orquestador.

Contiene la función: ***func\_AddQueue\_Input***, la cual mediante la actividad *Bulk Add Queue Items* introduce los datos en la cola *in\_StrQueueName* que se encuentra en la carpeta *in\_StrQueueFolder*, valores definidos en el Config. Los datos provienen de la tabla filtrada que se obtuvo en *Initialization*, pasada como argumento (*in\_DtQueueElements).*

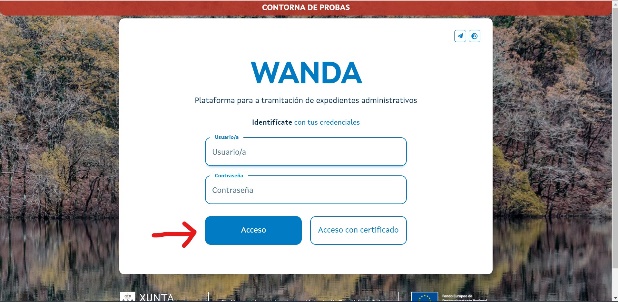
Las transiciones que presenta añadir ítems a la cola son las siguientes:

* *System Exception*: Si hay una incidencia de sistema se pasa al *End Process.*
* *Successful*: Si no ha habido ninguna incidencia el siguiente bloque será el de *End Process.*
  + - * 1. End Process

El funcionamiento de este bloque es terminar el proceso, cerrar las aplicaciones usadas y mandar un correo en caso de incidencia.

* **Componente: comp\_WANDA\_castellano\_LogOut**

1. Primero clica en el botón de *cerrar sesión* y comprueba que tras esto exista el botón *Acceso*, el cual aparece en la página donde se introducen las credenciales.

1. En caso de que no haya podido cerrar sesión correctamente se usa un  *Close Window*.

* **Componente: *comp\_CloseAllApplications***

(Definido en **comp\_CloseAllApplications**)

**comp\_CloseAllApplicationscomp\_CloseAllApplicationscomp\_CloseAllApplicationscomp\_CloseAllApplications**

Tras cerrar las aplicaciones comprueba si se ha producido alguna incidencia, en caso de que si realizará una captura de pantalla a través del flujo de trabajo **Framework: *TakeScreenshot*,** al cual se le pasa como argumento la ruta donde se guardará la captura. Esta ruta se establece en el *Config.*

Si se ha producido una incidencia, dependiendo de si es una de sistema o de negocio se generará un cuerpo de correo distinto y posteriormente se enviará al destinatario indicado en el *Config.*

Para llevar al cabo la generación del correo y el envío de este se han añadido dos componentes:

* **Componente: *comp\_castellano\_GenerarCuerpoCorreo***

Este componente se basa en generar el cuerpo de una plantilla de correo .html. La ruta de esta se define en el *Config*.

1. En primer lugar, se asignan valores a las variables de la plantilla del correo mediante una actividad *Multiple Assign*. Por ejemplo, el color del fondo, las imágenes que van a aparecer, el título, etc.
2. Se lee la plantilla del correo y se reemplazan las variables de esta por los valores establecidos en el punto 1.
3. Por último, se utilizará un *switch* para elegir el cuerpo que tendrá el correo en función del caso.

Casos contemplados en el *switch:*

* + completado:

Este caso significa que el proceso se ha completado, es decir, se ha ejecutado de principio a fin (Dispatcher y Performer).

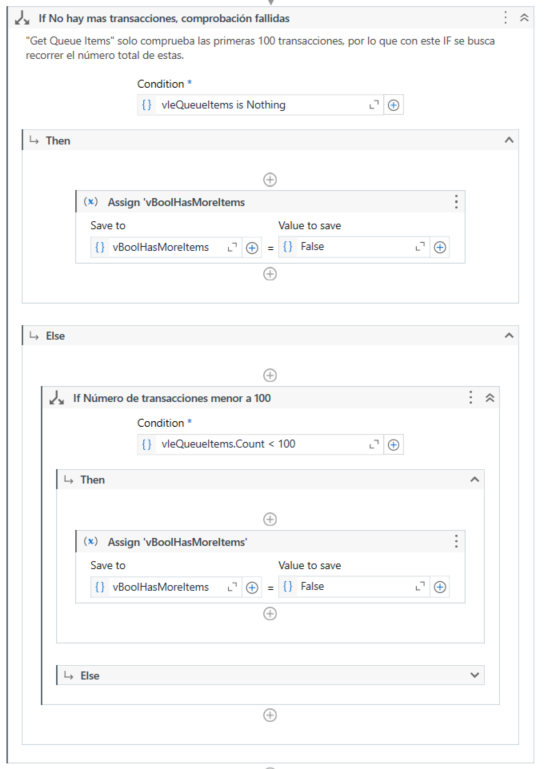
En primer lugar, se invoca el siguiente componente:

* + - **Componente: *comp\_castellano\_GenerarReporteOrquestador***

Este componente tiene como finalidad generar una tabla estadística indicando los diferentes estados que se han dado a lo largo del proceso. Se incluyen el total y el ratio de las transacciones que han sido completadas, en las que ha ocurrido alguna incidencia y las que han quedado sin tratar.

Con la primera secuencia que se encuentra en el componente se obtiene una lista de trabajos (*jobs*) en ejecución desde el orquestador de UiPath, filtrando por el nombre del proceso y el estado de ejecución. Luego, para cada trabajo (*job*) obtenido, se extrae y asigna la hora de inicio a la variable *vDateFilteredStartTime*. El *break* que se encuentra después del *Assign* indica que se debe detener el bucle después de la primera iteración.

En la siguiente secuencia se busca obtener el número de transacciones de la cola diferenciando por el estado de cada una (*Failed, Successful, New…*). En la actividad *Get Queue Items* se indica, en las propiedades, el tipo de transacción según el estado que se quiera obtener. Hay que tener en cuenta que esta actividad solo comprueba las primeras cien transacciones, por lo que se añade una actividad *if* para solventar este problema.



Para el número total de transacciones no se incluyen las que hayan sido borradas (*Deleted*).

Finalmente, de forma acumulativa (a la tabla con las estadísticas de las transacciones exitosas se le añaden las de las fallidas y así con las demás), se construye la tabla final.

Una vez obtenido el *DataTable,* se introducen los valores de esta tabla en la plantilla de la tabla del correo y se reemplazan el texto inicial y final por el mensaje asignado en las variables de la plantilla del correo.

Nota: En el resto de los casos no se incluye la tabla de estadísticas.

* + parada:

Este caso indica que ha habido una incidencia de tipo negocio a la hora de iniciar sesión en Wanda y por lo tanto no se puede continuar con el proceso. Esta información se verá reflejada en el cuerpo del correo.

* + incidencia:

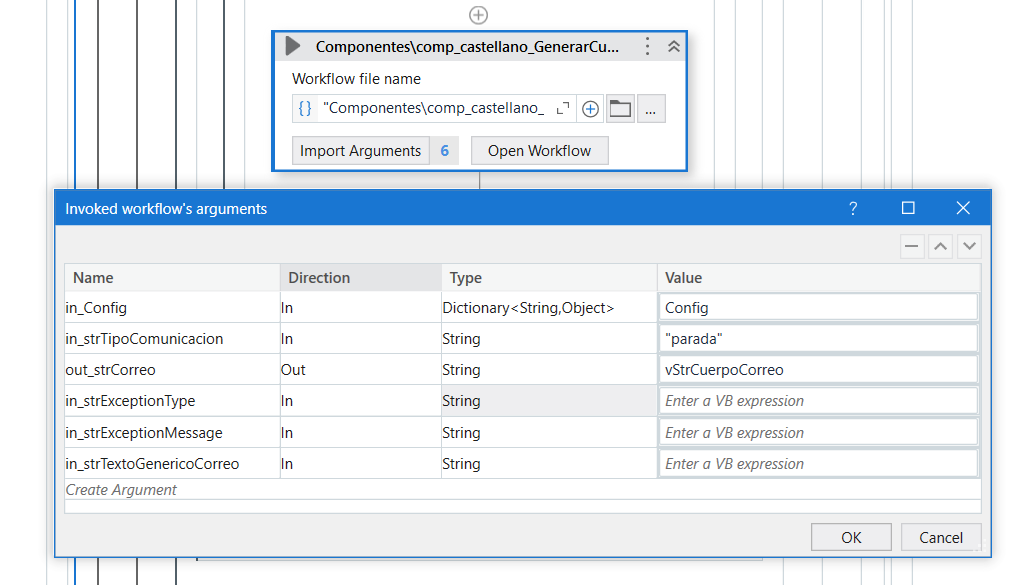
En este caso se indica en el cuerpo del correo que tipo de incidencia ha ocurrido y el mensaje correspondiente a esta.

* + genérico:

Este tipo de correo no se usa en el proceso, pero se ha implementado para que quede recogido la opción de un correo con un cuerpo genérico a disposición de si se quiere utilizar en otro proceso.

Para saber qué tipo de correo hay que mandar, este componente tiene como argumento de entrada *in\_strTipoComunicacion* en el cual, cuando se invoca, se establece el valor con el nombre de uno de los posibles casos mencionados anteriormente.

En la imagen a continuación se puede ver el caso de que se quiera mandar un correo cuando se produzca una parada.



Nota: En el Dispatcher de este proceso las dos opciones factibles de tipo de correo que se puede mandar son el de tipo parada o incidencia. El *completado* solo se podría mandar al llegar al *End Process* del Performer.

* **Componente: *comp\_castellano\_EnviarCorreo***

En este componente se busca mandar un correo a un destinatario definido en el argumento *in\_strCorreoDestinatario*. Se incluye un *Do While* para que en el caso de que falle el envío se intente reenviar el correo con o sin archivos hasta un máximo de cinco veces. En caso de un correo de tipo *completado* se adjunta el informe de ejecución en el correo y en caso de incidencia, una captura de pantalla que muestre el error. Si se supera los cinco intentos de reenvío del correo se lanzará un *throw* con una incidencia de sistema.

* + 1. Procesamiento del caso
       1. Performer

En este apartado se va a desarrollar el funcionamiento del Performer, el cual está estructurado en cuatro bloques.

* + - * 1. Initialization

Este bloque tiene una funcionalidad parecida al del *Dispatcher*.

En primer lugar, cierra todas las aplicaciones invocando **comp\_CloseAllApplications**. Luego comprueba si es la primera ejecución del proceso viendo que la variable *Config* de tipo *Dictionary<String, Object>* esté vacía. Si es así:

* Invoca ***InitAllSettings***
* Comprueba que el nombre de la carpeta de la cola y el nombre de esta esté especificado en el *Config* y en tal caso sobrescribe los nombres del orquestador por estos.

Lee el Config.xlsx y rellenará la variable de este nombre. Además, obtendrá del orquestador los valores del “Asset”.

Comprueba si se alcanzó el número *MaxConsecutiveSystemExceptions*, lanza una incidencia en la inicialización y va al estado *End Process*, finalizando así la ejecución. Si *MaxConsecutiveSystemExceptions* es 0, se permite cualquier número de excepciones consecutivas del sistema.

Posteriormente se invoca el **Componente: *comp\_castellano\_GetQueueMaxRetryNumber,*** con el cual se obtiene, del orquestador, el número máximo de intentos de la cola (*vIntQueueMaxRetryNumber*). Por defecto es uno.

Tras esto, dentro del *Invoke* de ***InitAllAplications*** se hace un *invoke* de las funciones/componentes propios del proceso. En este caso se invoca **comp\_WANDA\_castellano\_Login**.

Por último, se mueve la plantilla del informe de ejecución a la carpeta de interés. Las rutas que se tienen que indicar en esta actividad se establecen en el *Config.*

Las transiciones que presenta la inicialización del *Performer* son las siguientes:

* *Business Exception*: Si falla por una incidencia de negocio el destino será el bloque de *End Process*.
* *System Exception*: Si hay una incidencia de sistema se pasa al *End Process.*
* *Successful*: Si no ha habido ninguna incidencia el siguiente bloque será el de *Get Transaction Item*.
  + - * 1. Get Transaction Data

Recupera un nuevo dato de transacción para procesarlo. La variable *TransactionNumber* contiene el número de transacción actual y, al incrementar esta variable, el *framework* recupera la siguiente transacción. Si el *framework* está reintentando una transacción fallida, esta variable no se incrementa hasta que se alcanza el número máximo de intentos de reintento.

Dentro del bloque se encuentra:

* **Framework: *GetTransactionData***

Se obtiene una transacción de un origen especificado (por ejemplo, colas de Orchestrator, hojas de cálculo, bases de datos, buzones o API web).

Como hay varias transacciones, hay que utilizar el argumento *in\_TransactionNumber* como índice para recuperar la transacción correcta que se va a procesar. Cuando no queden más transacciones, se establece *out\_TransactionItem* como *Nothing*, lo que conduce al estado *End Process*.

Existe un paso opcional (un *if)* que se puede utilizar para incluir más información sobre una transacción y se utiliza principalmente para fines de registro y visualización. Para ello se crean argumentos de salida para almacenar esta información (por ejemplo: *out\_TransactionID*).

Las transiciones que presenta este bloque del *Performer* son las siguientes:

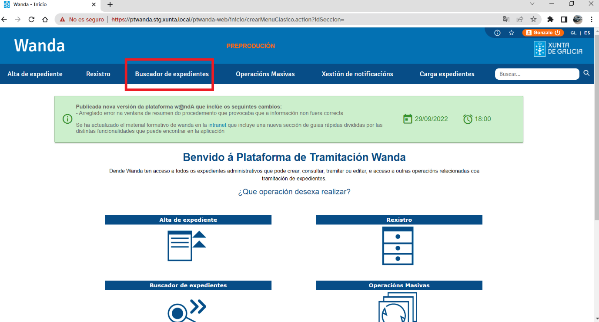
* *New Transaction*: Si hay una nueva transacción se pasa a *Process Transaction*.
* *No Data*: Se pasaría al *End Process.*
  + - * 1. Process Transaction

Se procesa cada transacción una a una.

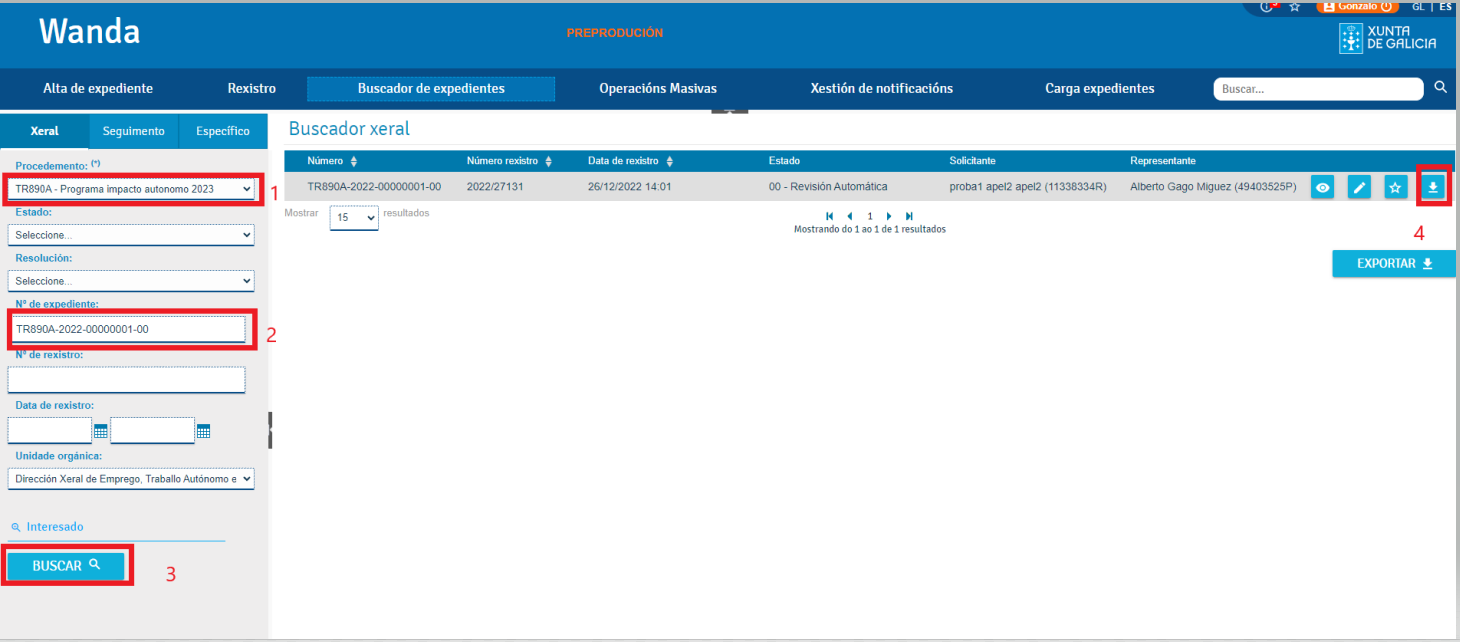
Primeramente, se añaden las funciones o componentes específicos (esquematizados en el HDL) del proceso dentro del flujo de trabajo “Proceso”:

* **Función:** ***func\_WANDA\_DescargarCarpetaExpediente***

En la página de inicio de WANDA se selecciona la pestaña “Buscador de expedientes”.



Se selecciona en “Procedimiento” el procedimiento pasado como argumento (especificado en el *Config*) y se introduce el número de expediente (obtenido de los datos contenidos en la transacción). Se clica en el botón de buscar y si se encuentra el expediente se pincha en el símbolo de descarga, sino lanzará un incidencia de sistema.



Como puede tardar más tiempo de la cuenta, se añadió una condición de que mientras que no encuentre el fichero descargado aumenta en un segundo un *delay*.

Por último, se descomprime el fichero del expediente en la ruta en la que se va a trabajar y posteriormente se elimina el fichero .ZIP.

Lo siguiente que se hace es comprobar si es el documento del que se quiere extraer los datos es un modelo 200 o 184. Este viene determinado en el *Config* y en función de si es uno u otro se utilizará la función:

* **Función: *func\_ObtenerDatosModelo200 / func\_ObtenerDatosModelo184***

Estas dos funciones tienen la misma estructura y funcionamiento, a diferencia de que analizan documentos distintos.

Los ficheros que se han descargado de Wanda son carpetas que tienen como nombre el número de expediente. Dentro de esta carpeta puede haber más de un documento, por lo que entre los que hay se debe comprobar que exista el documento del modelo de interés. Para esto se utilizará la actividad *For Each File in Folder* y mediante la actividad *Matches* se comprueba que exista los datos que se quieren extraer del modelo.

A la hora de utilizar la actividad *Read PDFText* según el modelo se configura un rango de lectura de las páginas diferente. Esto se hace, por si se sabe en que número de página suele estar los datos, para optimizar la búsqueda y tardar menos tiempo.

Para el modelo 200 se han determinado los siguientes *REGEX*:

* + MODELO\s200 : Encuentra que en el documento ponga que es un modelo 200.
  + Verificación(.\*?):? ([A-Z0-9]{16}): Localiza el código seguro de verificación (CSV).
  + 00255\s+(.+?)\s+ : Localiza el importe de la casilla 00255.
  + 200 ([ABCDEFGHJNPQRSUVW]\d{8}): Localiza el NIF del solicitante.

Para el modelo 184 se han determinado los siguientes *REGEX*:

Si los cuatro REGEX han tenido *Match*, significa que la lectura del PDF ha sido correcta y por lo tanto podemos introducirla en nuestro DataTable *out\_DtResultadosExtraidos*. Una vez que encuentre en un fichero los datos que se quieren extraer no se comprobarán el resto de los documentos de la carpeta.

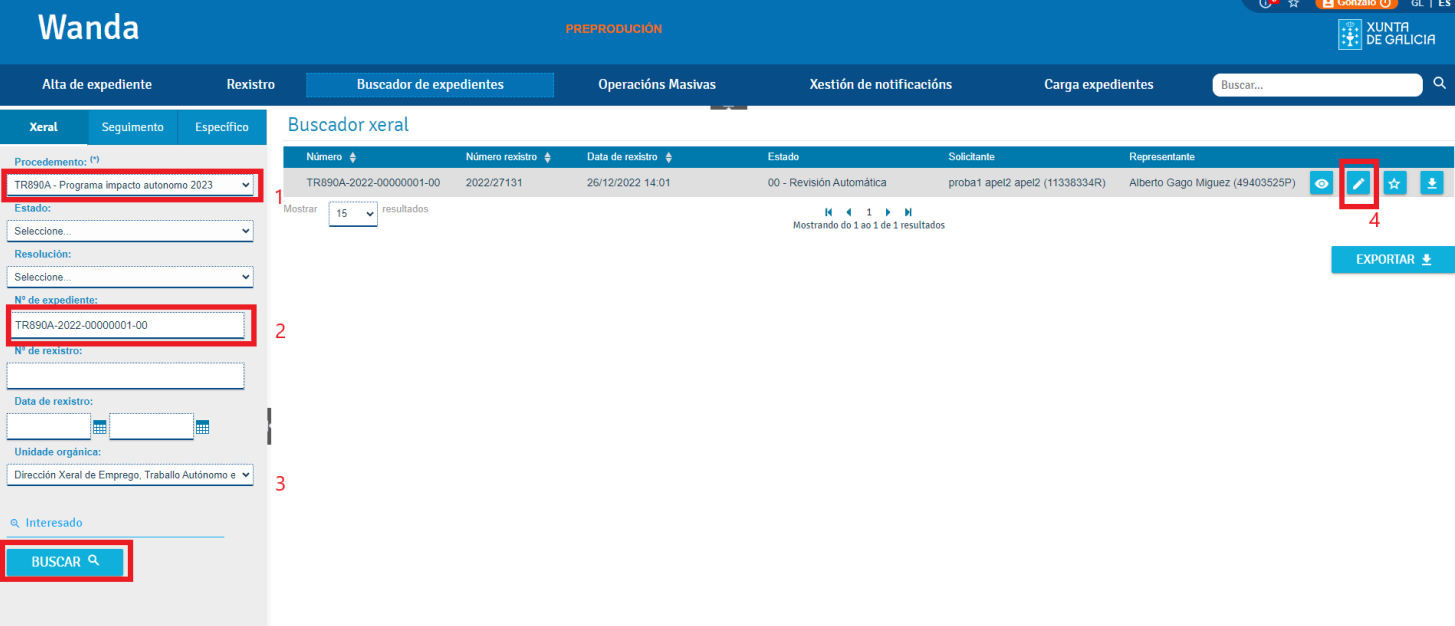
Si no se ha encontrado el documento del modelo de interés se lanza una incidencia de negocio.

En último lugar, se borra la carpeta comprobada.

Las siguientes funciones/componentes del *Process* son:

* **Componente: *Comp\_Wanda\_NavegarAExpediente***

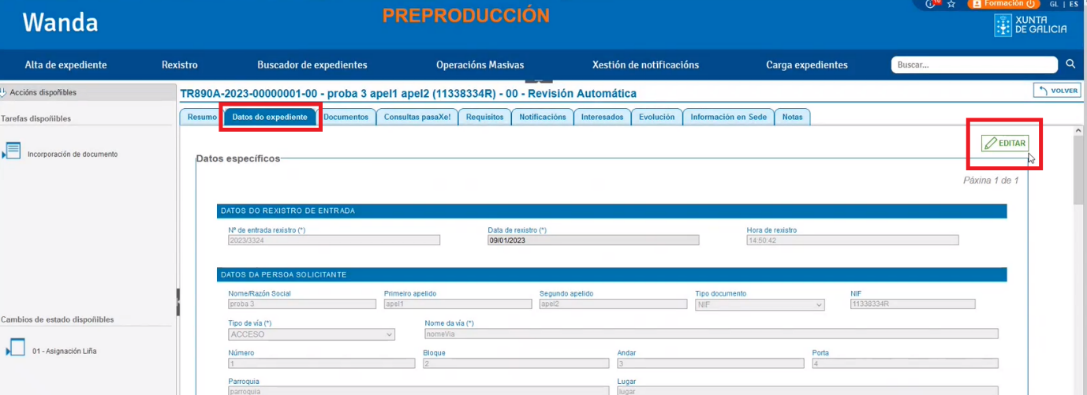
Este componente sigue el mismo camino que ***func\_WANDA\_DescargarCarpetaExpediente*** , es decir, pulsa *Buscador de expediente*, introduce el procedimiento, el número de expediente, pulsa buscar, comprueba que exista y finalmente, como paso distinto al de descargar, clica en el botón de editar.



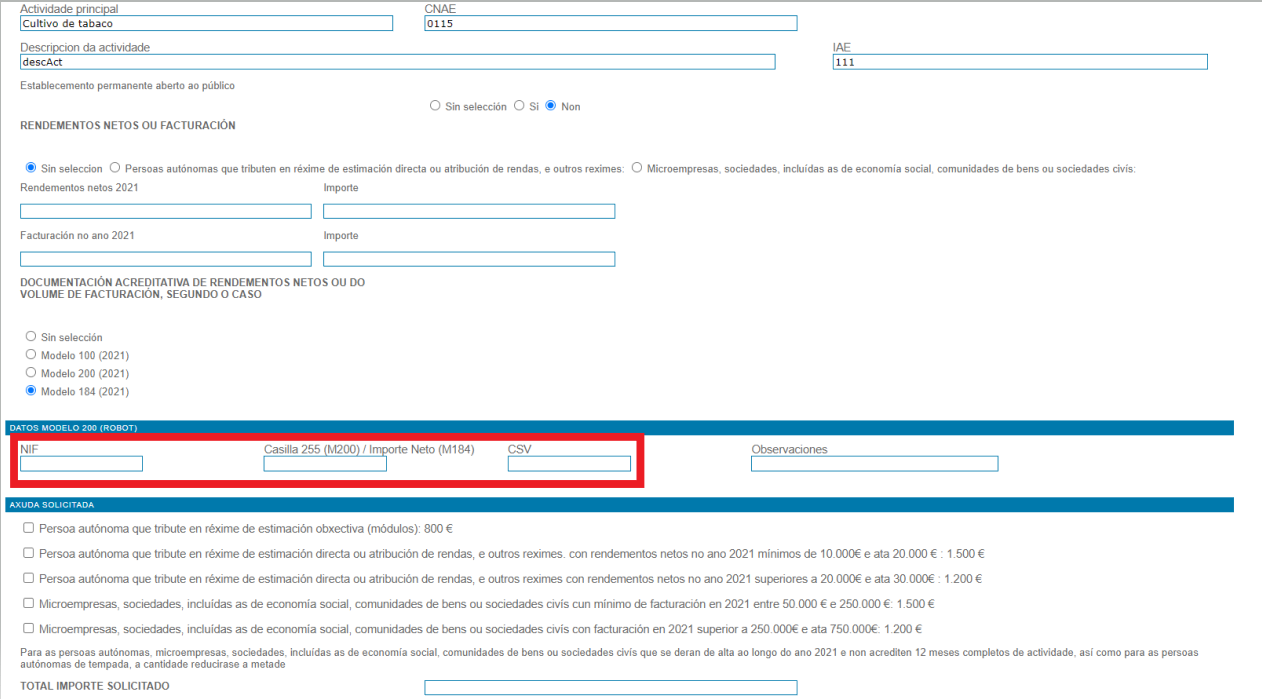
Si no encuentra el expediente lanzará un *throw* con una incidencia de negocio.

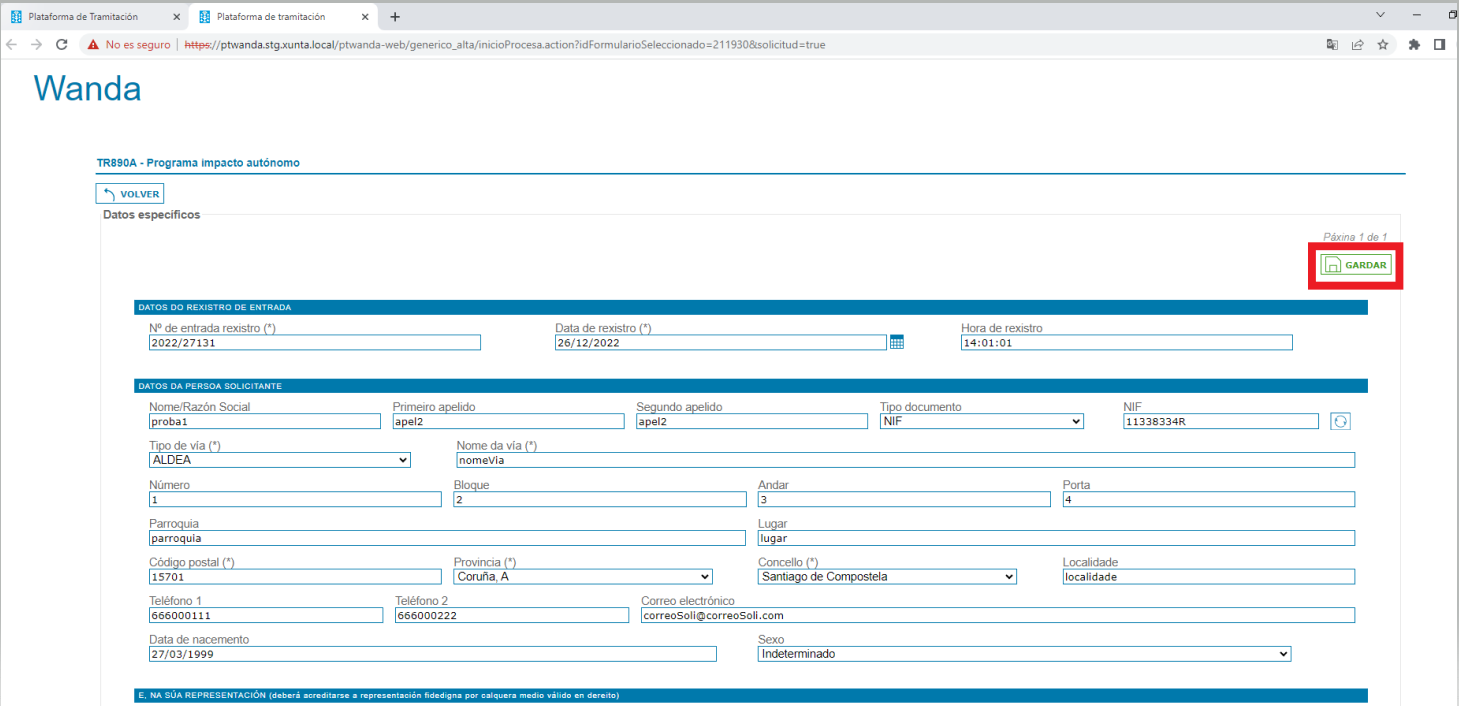
* **Función: *func\_Wanda\_IntroducirInfoEnExpediente***

Se accede a la pestaña *Datos de expediente* y se pincha en el botón de editar.



Se abre una pestaña nueva en la cual saldrá el expediente listo para editar. Mediante *type into* se introducen los datos obtenidos del modelo y se clica el botón de guardar.





Se comprueba mediante un *Element Exists* si se ha guardado bien, en tal caso se pulsa el botón de volver, el cual cerrará la pestaña y actualizará los cambios realizados en el expediente. En caso contrario se lanzará una incidencia de sistema y se cerrará la pestaña.

A continuación del bloque *Process* se invoca:

* **Framework**: ***SetTransactionStatus***

Se establece un flujo de decisión en el cual se comprueba si la transacción ha sido exitosa y si no lo ha sido por qué tipo de incidencia ha ocurrido.

En el caso de que el estado de la transacción sea *Success*  o *Business Ex*ception se incrementa el número de transacciones (*io\_TransactionNumber*) en uno para que la próxima se pueda procesar, se resetea a 0 el contador (*io\_RetryNumber*) para que la siguiente transacción se reintente el número correcto de veces y se resetea el contador de incidencias de sistema consecutivo (*io\_ConsecutiveSystemExceptions*). En caso de una incidencia de sistema seguirá otro camino específicado más adelante.

* + *Success:*

Si no ha habido ninguna incidencia se actualiza el estado de la transacción a exitosa, tras comprobar que el elemento de la cola no es nada y que además es de tipo *Queu Item*.

Después de esto se invoca el siguiente componente:

* + - **Componente: *comp\_castellano\_InformeEjecucion***

Primero se define qué columnas se van a meter en el informe. La plantilla del informe de ejecución alojada en la ruta definida en el *Config* debe contener las columnas correspondientes a los datos que se pinten en la actividad *Multiple Assign*. Se lee el informe de ejecución en cada transacción para obtener las filas que ha ido pintando en el transcurso del proceso. (Una fila por cada transacción realizada). Posteriormente, se comprueba el estado de la transacción y se añade una fila por cada una en la tabla.

Por último, se escribe la tabla en el informe de ejecución.

Tras crear el informe de ejecución se incluye campos de registro personalizados para el mensaje de registro. Se eliminan después del registro para evitar mensajes de estado duplicados sobre una sola transacción.

* + *Business Exception*

Este caso seguirá el mismo procedimiento que el caso de *Success* a excepción de que el estado de la transacción se establece como *Failed*.

* + *System Exception*:

Se realiza una captura de pantalla a través del flujo de trabajo **Framework: *TakeScreenshot*,** al cual se le pasa como argumento la ruta donde se guardará la captura. Esta ruta se establece en el *Config.*

Posteriormente, se establece el estado de la transacción a *Failed*, se incrementa el contador de errores de sistemas consecutivos y se invoca:

* + - **Framework: *RetryCurrentTransaction***

Se encarga del mecanismo de reintentos. Parte del n

En primer lugar, comprueba si tiene que reintentar alguna transacción->

* + - * No: Escribirá en el informe de ejecución y aumentará el número de transacción para continuar con la siguiente.
      * Sí: Se comprueba si se ha alcanzado el número máximo de intentos:

Por último, se eliminan los campos de registro (*log* flield) de la transacción y se invoca un **comp\_CloseAllApplications**.

El bloque Process Transaction presenta un *Try Catch* dentro del cual se invoca

* + 1. Reinicio

|  |
| --- |
| INSTRUCCIONES - Eliminar esta tabla cuando la sección esté terminada |
| Tras la finalización de un Ítem, si es necesario volver al estado original de ciertas variables, páginas iniciales de las aplicaciones… Será en este apartado donde se exponga todas las actividades que lleve a cabo la automatización.  A través de la plantilla común y estándar, la página “Reset Global Data” deberá comentarse las funcionalidades que recoge. |

* + 1. Recuperación

|  |
| --- |
| INSTRUCCIONES - Eliminar esta tabla cuando la sección esté terminada |
| La gestión de excepciones que debe realizar el robot se contempla en este punto. Es decir, todas las acciones requeridas para poder ejecutar el siguiente Ítem de la cola de trabajo. Por ejemplo, notificar correctamente de la excepción al SME, regresar al inicio de la aplicación. |

* + 1. Finalización

|  |
| --- |
| INSTRUCCIONES - Eliminar esta tabla cuando la sección esté terminada |
| Una vez se hayan completado todos los ítems de la cola, ya sea de forma correcta o con excepción, se deberá realizar una serie de acciones para la finalización del robot, como puede ser enviar un email con los resultados o cerrar todas las aplicaciones. |

1. Modelo de la información

|  |
| --- |
| INSTRUCCIONES - Eliminar esta tabla cuando la sección esté terminada |
| Para tener una imagen global sobre cómo se estructura la información almacenada en las colas, se deberá incluir un modelo que represente de manera formal y abstracta los tipos de datos que existen y su correlación.  Como se puede ver a continuación a modo de ejemplo, habrá que establecer los datos almacenados en la cola y a su vez si existen otras colecciones, Data Table de donde se obtengan. De este modo podremos conocer de una manera rápida cómo se gestiona la información de la cola. |

1. Procedimiento

De cara a un mantenimiento o soporte del proceso, se describirán los pasos a seguir en función de los estados por los que pasa la automatización y las posibles excepciones tanto de Negocio como de Sistema que están recogidas.

|  |
| --- |
| INSTRUCCIONES - Eliminar esta tabla cuando la sección esté terminada |
| La columna ‘Excepción’ es el título de la excepción que se visualizará desde Orchestrator y por la que podremos relacionarla en esta tabla. ‘Escenario’ contiene una breve descripción del motivo de la excepción.  En los pasos a seguir, se deberá detallar cómo actuar en caso de tener que ejecutar el robot manual o al relanzarlo tras alguna excepción mencionada anteriormente. Es importante describir bien cada excepción, para una mayor facilidad a la hora de dar soporte de la automatización. |

Excepciones de Negocio

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Excepción | Escenario | Pasos a seguir | Cola | Estado |
| Usuario sin permiso en Wanda | Credenciales válidas pero el usuario no tiene permiso para acceder a la aplicación. | **1.** Se toma una captura de pantalla.  **2.** Se lanza un “Throw”.  **3.** Se manda un correo al cliente informando de la incidencia. |  |  |
|  |  |  |  |  |

Excepciones de Sistema

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Excepción | Escenario | Pasos a seguir | Cola | Estado |
| Credenciales incorrectas | Las credenciales que se introducen, para el inicio de sesión en Wanda, son incorrectas. | **1.** Se toma una captura de pantalla.  **2.** Se lanza un “Throw”. |  |  |
| No se puede obtener los datos para la tabla de estadísticas | Error al generar la información de la cola para el envío, en una tabla, por email cuando el proceso se ha completado. |  |  |  |

Control de Estados

|  |  |  |
| --- | --- | --- |
| Estado | Nombre | Escenario |
| 1 | Carga de datos | Obtenidos todos los datos y cargados correctamente en la cola de trabajo del robot. |
| 2 | Factura PDF | Se ha creado la factura en formato PDF. |

|  |
| --- |
| INSTRUCCIONES - Eliminar esta tabla cuando la sección esté terminada |
| Un modelo de Estados representa todas las situaciones posibles que se pueden dar a lo largo de un proceso y en los cuales se deberá gestionar de una manera concreta. Esto nos proporcionará el conocimiento necesario para reanudar el proceso en caso de tener cualquier tipo de excepción.  En la columna ‘Nombre’ estableceremos de forma breve el escenario y será el nombre que aparecerá en la Cola visible desde la Control Room / Orchestrator. Por otro lado, la columna ‘Escenario’ será una explicación de las tareas que se han llevado a cabo en dicho estado. |

Planificación del Proceso

|  |
| --- |
| INSTRUCCIONES - Eliminar esta tabla cuando la sección esté terminada |
| Describe el horario en el que se ejecutará el robot y cualquier requerimiento necesario para que el robot pueda comenzar o finalizar el proceso, como puede ser un tipo de configuración en concreto o un input a descargar que sea imprescindible para la ejecución. |

Escalabilidad

|  |
| --- |
| INSTRUCCIONES - Eliminar esta tabla cuando la sección esté terminada |
| Describir las características de funcionamiento del robot desarrollado, como por ejemplo si soporte ejecuciones en paralelo, se puede ejecutar en diferentes máquinas, etc. Se deben de aportar las razones y justificar por las que se ha empleado esta solución técnica.  En caso de contar el proceso con varios robots, se deberá detallar el orden de lanzamiento y posibles requisitos de PRE y PRO ejecución. Ej. Un robot para actualizar inputs en la Cola y otro para procesar las transacciones. |

Sistema de Alerta

**<< OPCIONAL >>** Según la herramienta RPA utilizada y las necesidades de los Robot Controller de las que se dispongan. En caso de no necesitarlo eliminar este apartado.

Cualquier mensaje de alerta implementado en la Solución que no esté registrado en las Colas del programa, pero esté especificado en el PDD, se describe a continuación:

|  |  |  |
| --- | --- | --- |
| Escenario | Método | Destinatario(s) |
|  |  |  |

|  |
| --- |
| INSTRUCCIONES - Eliminar esta tabla cuando la sección esté terminada |
| Describe cualquier notificación que el usuario reciba en caso de una excepción. Algunos pueden haber sido ya listados en el apartado 4.1. como excepción de Negocio o Sistema.  Esto es útil para implementar un sistema donde el Robot Controller reciba la información (email, PopUp…) en su ordenador sobre el estado de las ejecuciones de los robots. |

1. Tratamiento de los datos

Esta sección contiene una explicación detallada de cómo el proceso gestiona todos los datos. Se centrará en la descripción de los detalles más significativos en cuanto a la gestión de la información que realiza la automatización.

Origen de la información

|  |
| --- |
| INSTRUCCIONES - Eliminar esta tabla cuando la sección esté terminada |
| Enumerar las fuentes desde las que el robot accede para, obtener la información necesaria para su correcta ejecución. Por ejemplo, en archivos inputs, extracción web, PDF... |

Almacenamiento de los Datos

|  |
| --- |
| INSTRUCCIONES - Eliminar esta tabla cuando la sección esté terminada |
| Describe dónde se almacenan los datos. Ya sea en la Cola de trabajo, Logs, Ficheros Excel, XML, bases de datos, Control Room/Orchestrator… Permitirá a los auditores realizar el seguimiento y análisis de todas las interacciones del robot en el sistema.  En caso de contar con Cuadros de Mando, se deberán tener en cuenta todos los datos que estas tablas necesiten, por lo que habrá especificar donde se almacenarán. |

Seguridad

|  |
| --- |
| INSTRUCCIONES - Eliminar esta tabla cuando la sección esté terminada |
| Describe cómo se gestiona la privacidad de datos en la Solución. ¿Qué información será visible en las colas? ¿Cómo se evitará la exposición de datos sensibles? ¿Dónde se almacenan las credenciales de acceso a los aplicativos?  Por ejemplo, los datos de empleado que se gestionan en la cola de trabajo son sustituidos por X excepto los dos últimos dígitos. Los documentos de reporte de ejecución se guardan en un servidor al que sólo pueden acceder los trabajadores del departamento de RRHH. |

Mantenimiento

|  |
| --- |
| INSTRUCCIONES - Eliminar esta tabla cuando la sección esté terminada |
| Describe cuanto tiempo estará disponible la información almacenada en la Solución y como será eliminada. También se detalla que mantenimiento realiza la automatización para no incluir información duplicada que pueda provocar una doble ejecución del mismo caso.  Por ejemplo, La información de las colas de trabajo se almacenan en el servidor del programa RPA durante tres meses. Todos los documentos que estén en el sistema del cliente serán los propios usuarios de negocio, los encargados de su mantenimiento. El robot gestiona una estructura de carpetas por año, mes y día, donde en caso de conflicto se sobrescribe el archivo. |

1. Consideraciones

|  |
| --- |
| INSTRUCCIONES - Eliminar esta tabla cuando la sección esté terminada |
| Describir asunciones o dependencias de la solución. Por ejemplo, aquellos requisitos necesarios de orden de lanzamiento en caso de existir varios robots diferenciados para este mismo proceso. También se debe incluir en este apartado, toda aquella información que no se haya incluido previamente y sea vital para la correcta finalización del proceso.  Se deberá incluir en este apartado cualquier objeto que pueda tener cierta sensibilidad a la hora de lanzarse, modificarse o de gran relevancia para el resto de los procesos que estén en producción y pueda generar problemas de compatibilidad. |

1. Anexo I: Glosario

Las siglas y términos utilizados a lo largo de este documento se describen a continuación.

|  |  |  |
| --- | --- | --- |
| Acrónimos / Términos | Sinónimo | Descripción completa |
| PDD | Documento Funcional | Process Design Document |
| SDD | Documento de Diseño | Solution Design Document |
| UAT | Aceptación usuario | User Acceptance Testing |
| SME | Experto de negocio | Subject Matter Expert |
| RC | - | Robot Controller |
| VM | VDI | Virtual Machine |
| VDI | VM | Virtual Desktop Interface |
| SSO | - | Single Sign-On |
| RPA | - | Robotics Process Automation |
| UI | Herramienta RPA | UiPath |

|  |
| --- |
| INSTRUCCIONES - Eliminar esta tabla cuando la sección esté terminada |
| Incluir terminología del proceso / Área / Departamento |