BestPractices

Minsait

El presente documento contiene una guía de buenas prácticas a la hora de trabajar con la tecnología de UiPath

Contenido

[1](#_Toc132627375)

[Alcance del documento 6](#_Toc132627376)

[Estructura de los procesos 6](#_Toc132627377)

[Nomenclatura 6](#_Toc132627378)

[1. Nomenclatura del Proceso 6](#_Toc132627379)

[2. Nomenclatura de las Librerías 7](#_Toc132627380)

[3. Nomenclatura de las Tablas y procedures de BBDD 7](#_Toc132627381)

[4. Nomenclatura de las Colas 7](#_Toc132627382)

[5. Nomenclatura de los Documentos 8](#_Toc132627383)

[6. Nomenclatura de Triggers 8](#_Toc132627384)

[7. Nomenclatura Defectos y Cambios 8](#_Toc132627385)

[8. Nomenclatura de Errores 9](#_Toc132627386)

[9. Nomenclatura de las actividades 9](#_Toc132627387)

[10. Nomenclatura de variables 9](#_Toc132627388)

[11. Nomenclatura de argumentos 10](#_Toc132627389)

[12. Nomenclatura y localización de Assets: 10](#_Toc132627390)

[Desarrollo 10](#_Toc132627391)

[1. REFramework 10](#_Toc132627392)

[D.1.1 10](#_Toc132627393)

[D.1.2 11](#_Toc132627394)

[D.1.3 11](#_Toc132627395)

[D.1.4 11](#_Toc132627396)

[D.1.5 11](#_Toc132627397)

[2. Idioma del desarrollo 11](#_Toc132627398)

[D.2.1 11](#_Toc132627399)

[3. Lenguaje de programación 11](#_Toc132627400)

[D.3.1 11](#_Toc132627401)

[4. Archivos de entrada 11](#_Toc132627402)

[Ficheros de Entrada 11](#_Toc132627403)

[D.4.1 11](#_Toc132627404)

[Emails de entrada 12](#_Toc132627405)

[D.4.2 12](#_Toc132627406)

[5. Relanzamiento 12](#_Toc132627407)

[D.5.1 12](#_Toc132627408)

[D.5.2 12](#_Toc132627409)

[6. Variables 13](#_Toc132627410)

[D.6.1 13](#_Toc132627411)

[D.6.2 13](#_Toc132627412)

[D.6.3 13](#_Toc132627413)

[D.6.4 13](#_Toc132627414)

[7. Argumentos 13](#_Toc132627415)

[D.7.1 Todos los procesos deben de incluir los siguientes argumentos (como mínimo) en Main.xaml: 13](#_Toc132627416)

[D.7.2 13](#_Toc132627417)

[D.7.3 13](#_Toc132627418)

[D.7.4 14](#_Toc132627419)

[D.7.5 14](#_Toc132627420)

[8. Librerías y reutilización de código 14](#_Toc132627421)

[D.8.1 14](#_Toc132627422)

[D.8.2 14](#_Toc132627423)

[D.8.3 14](#_Toc132627424)

[D.8.4 14](#_Toc132627425)

[9. Workflow Layout 14](#_Toc132627426)

[D.9.1 14](#_Toc132627427)

[D9.2 14](#_Toc132627428)

[D9.3 14](#_Toc132627429)

[D.9.4 14](#_Toc132627430)

[10. Selectores 14](#_Toc132627431)

[D.10.1 15](#_Toc132627432)

[D.10.2 15](#_Toc132627433)

[D.10.3 15](#_Toc132627434)

[D.10.4 15](#_Toc132627435)

[11. Actividades 15](#_Toc132627436)

[D.11.1: 15](#_Toc132627437)

[D.11.2 15](#_Toc132627438)

[D.11.3 15](#_Toc132627439)

[D.11.4 15](#_Toc132627440)

[D.11.5 16](#_Toc132627441)

[D.11.6 16](#_Toc132627442)

[D.11.7 16](#_Toc132627443)

[D.11.8 16](#_Toc132627444)

[D.11.9 16](#_Toc132627445)

[12. Integración 16](#_Toc132627446)

[D.12.1 16](#_Toc132627447)

[13. Uso de BBDD 16](#_Toc132627448)

[D.13.1 16](#_Toc132627449)

[D.13.2 16](#_Toc132627450)

[D.13.3 16](#_Toc132627451)

[D.13.4 16](#_Toc132627452)

[14. Excepciones y Error Handling 16](#_Toc132627453)

[D.14.1 17](#_Toc132627454)

[D.14.2 17](#_Toc132627455)

[D.14.3 17](#_Toc132627456)

[D.14.4 17](#_Toc132627457)

[D.14.5 17](#_Toc132627458)

[Excepciones inicialización 18](#_Toc132627459)

[Excepciones por ítem 18](#_Toc132627460)

[15. Cuentas de correo 18](#_Toc132627461)

[D.15.1 18](#_Toc132627462)

[D.15.2 18](#_Toc132627463)

[16. Uso de APIs y HTTP Request 18](#_Toc132627464)

[D.16.1 18](#_Toc132627465)

[D.16.2 18](#_Toc132627466)

[D.16.3 18](#_Toc132627467)

[D.16.4 18](#_Toc132627468)

[17. Macros Excel 18](#_Toc132627469)

[D.17.1 19](#_Toc132627470)

[D.17.2 19](#_Toc132627471)

[18. Excel y Datatables 19](#_Toc132627472)

[D.18.1 19](#_Toc132627473)

[D.18.2 19](#_Toc132627474)

[D.18.3 19](#_Toc132627475)

[D.18.4 19](#_Toc132627476)

[D.18.5 19](#_Toc132627477)

[D.18.6 19](#_Toc132627478)

[D.18.7 19](#_Toc132627479)

[D.18.8 19](#_Toc132627480)

[19. Multimáquina 19](#_Toc132627481)

[D.19.1 19](#_Toc132627482)

[D.19.2 20](#_Toc132627483)

[20. Logs 20](#_Toc132627484)

[D.20.1 20](#_Toc132627485)

[D.20.2 20](#_Toc132627486)

[D.20.3 20](#_Toc132627487)

[D.20.4 20](#_Toc132627488)

[21. Comentarios y anotaciones 20](#_Toc132627489)

[D.21.1 20](#_Toc132627490)

[D.21.2 20](#_Toc132627491)

[D.21.3 21](#_Toc132627492)

[22. Configuraciones 21](#_Toc132627493)

[D.22.1 21](#_Toc132627494)

[D.22.2 21](#_Toc132627495)

[D.22.3 21](#_Toc132627496)

[D.22.4: 21](#_Toc132627497)

[D.22.5 22](#_Toc132627498)

[D.22.6 22](#_Toc132627499)

[D.22.7 22](#_Toc132627500)

[23. Control de versiones 22](#_Toc132627501)

[D.23.1: 22](#_Toc132627502)

[D.23.2: 22](#_Toc132627503)

[D.23.3 22](#_Toc132627504)

[D.23.4 22](#_Toc132627505)

[D.23.5 22](#_Toc132627506)

[24. Rutas y URLs 23](#_Toc132627507)

[D.24.1 23](#_Toc132627508)

[25. Workflow Analyzer 23](#_Toc132627509)

[D.25.1 23](#_Toc132627510)

[26. QueueItems 23](#_Toc132627511)

[D.26.1 23](#_Toc132627512)

[27. Triggers 23](#_Toc132627513)

[D.27.1 23](#_Toc132627514)

[28. Assets 24](#_Toc132627515)

[D.28.1 24](#_Toc132627516)

[D.28.2 24](#_Toc132627517)

[D.28.3 24](#_Toc132627518)

[29.1. Publicación de procesos - SemVer 24](#_Toc132627519)

[D.29.1 24](#_Toc132627520)

[D.29.2 24](#_Toc132627521)

[Anexo: Recomendaciones generales 25](#_Toc132627522)

[A1 25](#_Toc132627523)

[A2 25](#_Toc132627524)

[A3 25](#_Toc132627525)

[A4 25](#_Toc132627526)

[A5 25](#_Toc132627527)

[A6 25](#_Toc132627528)

[A7 25](#_Toc132627529)

[A8 25](#_Toc132627530)

[A9 25](#_Toc132627531)

[A10 25](#_Toc132627532)

[A11 26](#_Toc132627533)

[A12 26](#_Toc132627534)

[A13 26](#_Toc132627535)

[A14 26](#_Toc132627536)

[A15 26](#_Toc132627537)

[A16 26](#_Toc132627538)

[A17 26](#_Toc132627539)

# Alcance del documento

Este manual pretender establecer y unificar los criterios de diseño de procesos RPA de **Minsait** en Proyecto para adecuarlos a un enfoque que permita mejorar el desarrollo, pruebas, operación y mantenimiento de los mismos una vez en entornos productivos.

Las recomendaciones aquí expuestas deben ser tenidas en cuenta a la hora del diseño de la solución, es decir, antes del comienzo de la implementación del proceso.

**Los textos indicados en azul serán recomendaciones provisionales, a decidir durante la implementación del proyecto y supeditados a la existencia de la infraestructura y políticas de seguridad adecuadas.**

**Los textos en naranja indican notas y recomendaciones en caso de que el texto en azul no pueda cumplirse.**

**Los textos en verde serán criterios a convenir con cliente**

# Estructura de los procesos

Los procesos se desarrollarán haciendo uso de la plantilla **ExtendedREFramework** para todos los procesos. El uso de la plantilla es necesario para disponer de las versiones correctas de dependencias, mensajes, alertas, y comunicación con BBDD. En caso de no disponer de BBDD adicional, se usará el mismo REFramework modificado para no hacer uso de la BBDD.

Todos los procesos (transaccionales o no), deberán aplicar el uso de colas. En los procesos transaccionales, el uso de colas trazará los datos globales de cada transacción y adicionalmente los datos particulares de transacción irán recogidos en una tabla de Base de Datos, que enlazará de forma unívoca los ítems de la cola. Para los procesos no transaccionales, será la traza de ejecución la transacción de cola y si fuera requerido tener datos particulares de ejecución, se enlazará de forma unívoca con ítems de una tabla en BBDD.

Por defecto todos los procesos se diseñarán con tres etapas principales:

* **Colector**: Recoge los datos de una aplicación, transforma y adapta los datos y los introduce en cola
* **Consumidor**: Recoge el ítem de cola y hace las acciones necesarias (lógica de negocio) en las aplicaciones.
  + En ocasiones en las que la lógica de negocio sea muy complicada y/o implique interacción en varias aplicaciones distintas, se dividirá en varios subprocesos de tipo Consumidor
* **Reporte**: Recoge los datos de las últimas ejecuciones (según reglas de negocio) y envía el reporte operativo

La plantilla base (ExtendedREFramework) se puede utilizar para cualquier tipo de proceso, tanto lineal como transaccional, haciendo las adaptaciones necesarias. Por defecto, la plantilla está pensada para procesos de tipo consumidor (transaccional) por colas. Para algunos casos de adaptaciones muy comunes, existirá una sub-plantilla en la que ya se han hecho los cambios necesarios para un tipo de proceso, por ejemplo:

* ExtendedREFramework\_ColectorExcel: Para procesos tipo colector en los que se lee un Excel (DataTable) y se deben de cargar datos en cola
* ExtendedREFramework\_Reporter: Para los procesos genéricos de reporte
* (…)

Todos los procesos (transaccionales o no), deberán aplicar el uso de colas. En los procesos transaccionales, el uso de colas trazará los datos globales de cada transacción y adicionalmente los datos particulares de transacción irán recogidos en una tabla de Base de Datos, que enlazará de forma unívoca los ítems de la cola. Para los procesos no transaccionales, será la traza de ejecución la transacción de cola y si fuera requerido tener datos particulares de ejecución, se enlazará de forma unívoca con ítems de una tabla en BBDD.

# Nomenclatura

## Logotipo, nombre de la empresa Descripción generada automáticamenteNomenclatura del Proceso

Los procesos deben llevar nombres que los identifique fácilmente, la estructura de un proceso será:

<Área>&<SECUENCIAL> \_ <3Siglas> \_ <Descripción>

Un ejemplo podría ser un proceso para el área de RRHH, donde la nomenclatura sería:

**RH001\_DDN\_DescargaDeNotificaciones**

Para los subprocesos Colector, Consumidor y Reporte, la raíz es la misma y se añade al final “\_Colector” / “\_Consumidor” / “\_Reporte”.

Nombre corto de proceso:

<Área>&<SECUENCIAL> \_ <Departamento> \_ <3Siglas>

RH001\_DDN

Código Proceso (para utilizar en assets etc.):

<Área>&<3Siglas>

RHDDN

## Nomenclatura de las Librerías

Las Librerías irán separadas por aplicación de uso (Una librería por cada aplicación). El nombre de la librería será RPA.[CODIGO\_APLICACION][Aplicacion]. Por ejemplo: RPA.005.Salesforce

Las acciones (workflows) de estas librerías seguirán la nomenclatura indicada en el apartado Workflows, sin añadir el nombre de la aplicación

## Nomenclatura de las Tablas y procedures de BBDD

Las tablas de procesos deben comenzar con el código de identificación del proceso. Para todo proceso, debe existir al menos una tabla Principal, donde se guarde la información que se relaciona con la cola de UiPath 1:1 y se llamará Main.

CLDDN\_Main

El resto de tablas de procesos se nombrará con una descripción de la tabla. Por ejemplo: CLDDN\_ConfiguracionClientes.

Así mismo, los procedures necesarios para un proceso se nombrarán con el código identificativo del proceso seguido de un descriptivo para el procedure, Por ejemplo:

COADE\_SelectGrupos

Para las tablas de traza histórica, el nombre será el mismo que para la tabla principal, añadiendo “Hist” al final. Por ejemplo: CLDDN\_MainHist

La tabla de Errores se llamará: CodErrores.

## 4. Nomenclatura de las Colas

- Si existe solamente una cola para el proceso, se nombrará como sigue:

[Código]\_[DescripciónTransacción]

Ej.: CLDDN\_EmailsClientes

## Nomenclatura de los Documentos

Se nombran como:

RPA\_Proceso[Código]\_[NombreDocumento]\_vX.X

Texto

Descripción generada automáticamente

## Nomenclatura de Triggers

Los triggers llevarán exactamente el mismo nombre del proceso que lanzan, más una descripción del trigger. Si se trata del dispatcher, será:

<AREA>\_<Departamento>&<SECUENCIAL>\_<3Siglas>\_<Descripción>\_[TriggerDescripción]

Ejemplo:

PE001\_OPE\_DDN\_DescargaNotificacionesEmail\_PrimeraEjecucion

Si se trata del consumidor, será:

<AREA>\_<Departamento>&<SECUENCIAL>\_<3Siglas>\_<Descripción>\_Consumidor\_[TriggerDescripción]

## Nomenclatura Defectos y Cambios

Los Defectos de código tendrán la siguiente nomenclatura:

<Código>\_DFXXXX

Donde XXXX es el contador de defecto.

Los Cambios tanto inesperados como esperados, si son pequeños, seguirán la siguiente nomenclatura:

<Código>\_CMBXXXX

Donde XXXX es el contador de cambio pequeño.

Los cambios grandes o evolutivos:

<Código>\_EVOXXXX

Donde XXXX es el contador de evolutivo.

## Nomenclatura de Errores

Todas las excepciones conocidas deben llevar un código único y todos los códigos deben asignarse tanto en el SDD como en el Manual de Operación.

Las excepciones seguirán la siguiente nomenclatura:

[TipoExcepcion]\_[Aplicación]\_[Proceso]\_[Contador]

[TipoExcepcion]:

0 – Excepción de Negocio

1 – Excepción de Sistema

[Aplicación]:

0000 – Global o no dependiente de aplicación

0001 – 9999 Contador de Aplicación

[Proceso]

0000 – Global o no dependiente de proceso

0001 – 9999 Contador de Proceso

[Contador]

0001 – 9999

Por ejemplo, para una excepción de sistema recogido en una librería de la aplicación de salesforce, el código sería:

1-0001-0000-0015 – Excepción al comprobar si se ha cargado la ventana XXX esperada.

(La descripción es con carácter ejemplificativo).

Para una excepción de negocio relacionada con el proceso DNE cuyo contador de proceso es 001, sería:

0-0000-0001-0002 – Email con formato erróneo.

(La descripción es con carácter ejemplificativo).

Dichas excepciones Conocidas, sus códigos y descripciones, han de estar definidas en la tabla CodErrores de las BBDD de RPA, no hardcodeadas en el código fuente.

## Nomenclatura de las actividades

Es obligatorio cambiar el nombre por defecto de la actividad para hacerlo descriptivo

Así, por ejemplo:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamentetendría que ser Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

## Nomenclatura de variables

Se utilizará PascalCase para los nombres de variables

Los nombres de las variables deben de ser descriptivos. No llevarán prefijos salvo en aquellos casos en los que se necesite transformar el mismo valor de un tipo de variable a otro, por ejemplo, strImporteEnEuros->intImporteEnEuros

No se pueden tener múltiples variables que sólo se diferencien por un número o un código identificativo, por ejemplo:

ImporteA, ImporteB -> ImporteMensual, ImporteTrimestral

Las variables de tipo booleano deben de tener nombres para el caso “positivo”: VentanaAparece en lugar de VentanaNoAparece

## Nomenclatura de argumentos

Aplican las mismas reglas que para la nomenclatura de variables, añadiendo además un prefijo según el tipo de argumento:

in\_[nombre] para argumentos de entrada

out\_[nombre] para argumentos de salida

inout\_[nombre] para argumentos de entrada/salida

## Nomenclatura y localización de Assets:

Los assets que sean “Globales”, es decir, de uso para cualquier máquina y cualquier proceso (por ejemplo, credenciales de aplicaciones comunes, URLs etc) se dejan en la carpeta base ***RPA CoE***

Las Credenciales se crearán con la nomenclatura:

CREDENCIAL\_[Aplicación]\_[Entorno]

Por ejemplo:

**CREDENCIAL\_APIGES\_PRO**

Las URLs se crearán con la estructura:

URL\_[Web/API/Path]\_[Aplicación]\_[Entorno] Para el caso de la URL base

URL\_[Web/API/Path]\_[Aplicación]\_[Endpoint] Para las URLs relativas/endpoints.

Por ejemplo:

https://docs.uipath.com/ : **URL\_Web\_UIPDocs\_Pro**

/activities/other/latest : **URL\_Web\_UIPDocs\_LatestActivities**

Los assets de uso específico para un proceso, irán en la carpeta del proceso, y como prefijo tendrá el código identificativo proceso, por ejemplo **COADE\_PlantillaCorreoEnviar**

# Desarrollo

## 1. REFramework

D.1.1: Es obligatorio el uso de la plantilla ExtendedREFramework para todos los procesos

Esta plantilla se puede encontrar con facilidad en UiPath Studio en la vista *Start*, bajo la sección *New from Template*:

(AÑADIR FOTO)

La plantilla está basada en el REFramework original de UIPath, al cual se le añaden los parámetros estándar de configuración para procesos de Proyecto, así como las actividades de gestión de excepciones por base de datos

D.1.2: Almacenar la configuración (Config.xlsx) en dicho fichero de configuración y consumirlo desde él es obligatorio. No se podrán dejar constantes por el código, como por ejemplo cadenas de texto literales.

D.1.3: Los ficheros de configuración que finalmente serán consumidos por los robots se almacenarán en la ruta de proceso de la carpeta compartida.El Framework, para cada ejecución, copia de forma temporal dicho fichero para leerlo desde local. La estructura de la ruta por defecto del config será:

[RutaBase]**\03.Config\[Entorno]\[Nombre Proceso]** donde Entorno es :

{01. Preproduccion ó

02. Produccion} En caso de no disponer del recurso compartido, se mantendrán los dos configs en la carpeta del código, y haciendo uso del argumento de entrada del proceso se lanzará con un config u otro.

D.1.4 El flujo de Main.xaml no será modificado sin aprobación. Todo el código relativo a lógica de negocio deberá ir en el resto de workflows del proceso

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

D.1.5 No se pueden tener múltiples versiones del mismo fichero Config (Config v1, Config 22-04-2022 etc) en la carpeta de forma permanente, pero sí se pueden tener versiones para distintos entornos. Se llamarán Config\_[Entorno].xlsx

## 2. Idioma del desarrollo

D.2.1: Como base de buenas prácticas, el idioma del desarrollo (código) será inglés, por lo que todas las variables, argumentos, nombres descriptivos, comentarios y documentación del proyecto que hagamos se harán siempre en este idioma.

**D.2.2:** El idioma de las aplicaciones y navegadores será el Español. Todos los selectores y parámetros relativos a configuración de aplicaciones será el Español

## 3. Lenguaje de programación

D.3.1: El lenguaje de programación a usar para el proyecto será VB.NET

## Logotipo, nombre de la empresa Descripción generada automáticamente4. Archivos de entrada

### Ficheros de Entrada

D.4.1 En el caso de que la entrada del proceso sean ficheros que el robot recogerá al inicio, lo recomendable es crear la siguiente estructura de carpetas de Sharepoint.

1. Input

• ENTRADA: Directorio donde el usuario depositará los ficheros de entrada. Que no haya ningún fichero en el momento de la ejecución no tiene por qué ser un error.

• EN PROCESO: Se colocan en esta carpeta los archivos que están siendo procesados por el robot. De esta forma se minimiza el posible impacto de usuarios/robots accediendo al mismo archivo.

• PROCESADOS: Cuando el proceso termina de tratar todos los registros del fichero, no antes, lo moveremos a la carpeta de procesados creando un directorio con la fecha del día de procesado.

• ERRONEOS: En caso de error en la ejecución de alguno de los ficheros, se traslada a la carpeta erróneos para su tratamiento y aislamiento del resto de casos.

1. Output

La ruta de Sharepoint raíz será:

[(…)](https://clarkemodetcloud.sharepoint.com/:f:/r/sites/CTO/Documentos%20compartidos/CTO%20Corporate/RPA/03.%20Exchange)

Se debe seleccionar si es producción o preproducción

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Dentro se encuentra una estructura por País, Departamento y Proceso. En su interior, se creará la estructura de carpeta mencionada anteriormente.

En caso de no usar sharepoint, la estructura de carpetas es igualmente aplicable para el recurso compartido o bucket que se utilice para los ficheros

### Emails de entrada

D.4.2 En el caso de que la entrada del proceso sean emails que el robot recogerá al inicio, lo recomendable es crear la siguiente estructura de carpetas en la cuenta de email, dentro de una carpeta con las siglas del proceso:

• ENTRADA: Directorio donde se reenviarán los emails de entrada. Pueden ser movidos manualmente por el usuario, o bien automatizados con una regla. Que no haya ningún email en el momento de la ejecución no tiene por qué ser un error.

• PROCESADOS: Cuando el proceso termina de tratar un email, no antes, lo moveremos a la carpeta de procesados.

• ERRONEOS: En caso de error en la ejecución de algún email, se traslada a la carpeta erróneos para su tratamiento y aislamiento del resto de casos.

## Logotipo, nombre de la empresa Descripción generada automáticamente5. Relanzamiento

D.5.1 Siempre que sea posible, un proceso debe ser diseñado de manera que un relanzamiento no implique una intervención manual más allá de relanzar el mismo job en orchestrator (cambiando los argumentos de entrada si fuera necesario), suponiendo que pueden existir trabajos a medias que pueden interferir en su ejecución.

Deben controlarse en las colas las transacciones en progreso, comprobando la existencia de dichas transacciones e incluso cerrándolas si fuera necesario.

También debe controlarse el flujo de los archivos o emails en caso necesario.

Siempre que, por cuestiones de diseño, no sea posible automatizar las condiciones de relanzamiento y sean necesarias acciones manuales, deberán incluirse en los emails de notificación que envía el propio proceso en caso de interrupción.

D.5.2 Además, para optimizar las ejecuciones, los relanzamiento y reintentos deben de evitar repetir pasos que ya se han hecho. Para ello (dependiendo del proceso) podemos hacer uso del campo Progress del Item de Cola, o una columna similar en la tabla de BBDD asociada al proceso.

## 6. Variables

D.6.1 Al definir variables, tendremos que acotar correctamente siempre el scope. Hay que minimizar el uso de variables globales a todo un workflow, restringiendo el scope al mínimo

D.6.2: A la hora de extraer datos y almacenar en variables, no utilizar y arrastrar a lo largo del workflow múltiples variables simples para almacenar datos similares, usar arrays/listas/diccionarios según aplique. Por ejemplo, no crear 4 variables tipo “DatoA, DatoB, DatoC”, crear un único diccionario “Datos” y rellenar los valores

D.6.3 Las conversiones/manipulaciones de variables se harán en el sentido más corto posible, evitando la conversión intermedia a String si es posible. Por ejemplo: DateTimeA, queremos que el mes sea distinto -> Hacemos un AddMonths, NO hacemos DateTimeA.ToString.Replace y luego volvemos a convertir a DateTime. Este punto es especialmente aplicable para variables que vengan de un diccionario de objetos como el Config

D.6.4 Para los casos en los que se esté obteniendo un diccionario (o una variable equivalente a diccionario como JToken, JSON Object) es necesario comprobar que la “key” que vayamos a usar, exista, si no, dará excepción

## 7. Argumentos

D.7.1 Todos los procesos deben de incluir los siguientes argumentos (como mínimo) en Main.xaml:

* **Environment** (Obligatorio, string): DEV/PRE/PROD. Se usará para determinar a qué versión de las aplicaciones se accede.
* **Debug** (Obligatorio, bool): Se usará como variable genérica para ejecutar/no ejecutar ciertas partes del código si definimos que estamos en modo pruebas
* **StopTime** (Opcional, string dd/MM/yyyy HH:mm): Variable de control de hora de parada del proceso
* **ConfigPath** (Obligatorio, string): Ruta donde se aloja el fichero de Config
* **OrchestratorQueueFolder** (string): Vacío por defecto, sólo se usará en casos de procesos globales que tengan que acceder a recursos de otra carpeta
* **OrchestratorQueueName** (string): Vacío por defecto, sólo se usará en casos de procesos globales que tengan que acceder a recursos de otra carpeta, o para procesos en los que existan múltiples colas de entrada

D.7.2 Los argumentos de Main.xaml se deben de almacenar en la variable Config al inicializar el proceso. Dependiendo de si es obligatorio o no, tendremos dos construcciones:

**Argumento Obligatorio**: Check valor del argumento, si está vacío, lanzamos excepción, si no, guardamos valor en Config como Config(“NombreArgumento”) = Valor

**Argumento Opcional**: Check valor del argumento, si está vacío, utilizamos el valor por defecto alojado en el Config, si no, guardamos el valor como Config(“NombreArgumento”) = Valor

D.7.3: Evitar el uso de argumentos de tipo inout salvo que realmente sean necesarios

D.7.4: Los argumentos de credenciales se enviarán siempre como SecureString

D.7.5: El argumento “Config” (diccionario de configuración) se enviará a la mayoría de workflows, siendo la única excepción posible los workflows más elementales del proceso, en los que se enviará el dato directamente, por ejemplo:

Workflow “Apiges\_RealizarComprobaciones”: Workflow con mucha navegación y que a su vez invoca otros workflows: Se envía el Config

Workflow “ConvertirFechaLocalAUTC”: Worfklow elemental, se envían directamente los argumentos (p ej Fecha de entrada, fecha de salida)

## 8. Librerías y reutilización de código

D.8.1: Las aplicaciones a utilizar se mapearán en diversas librerías para hacer acciones comunes. En la librería de la aplicación se incluyen tanto workflows de acciones comunes (ej Login, Logout etc) como los elementos comunes utilizando Object Repository

D.8.2: cuando se realice un código de mantenimiento a un proyecto de librería, se deberá probar en todos los proyectos que lo empleen

D.8.3: el uso de snippets estará limitado a construcciones de “código puro” sin lógica de negocio, por ejemplo, transformación de variables, procesamientos de texto etc. Es importante evitar el uso de snippets ya que estos no se actualizan automáticamente en los procesos aunque se actualice el origen del Snippet

D.8.4: Todas las navegaciones en aplicaciones que impliquen lógica de negocio y no se puedan pasar a librería por algún motivo, se implementarán como un workflow independiente para poder reutilizar. No se podrá tener acciones a bajo nivel (clicks etc.) directamente en los workflows principales del proceso (Main, Process etc.)

## 9. Workflow Layout

Hay que comprender cuándo se debe utilizar cada uno de los distintos tipos de *Workflow Layout* de UiPath.

Se establece la siguiente jerarquía según la complejidad/alto nivel del workflow: State Machine -> Flowchart -> Sequence.

D.9.1: No se anidarán elementos de una jerarquía superior dentro de una inferior (ej. State Machine dentro de una Sequence)

D9.2: Se evitará más de un nivel de anidación para State Machine y Flowchart, es decir, en cada workflow sólo debe de haber un State Machine/Flowchart. Cualquier lógica que implique otro flowchart/state machine se hará mediante Invoke Workflow.

D9.3 Evitar “Sequences” anidadas innecesarias: Suele ocurrir al generar bucles e ifs, que se generan Secuencias dentro de Secuencias que no son necesarias.

D.9.4 En general, se reservará el uso de los Workflows tipo “sequence” para las acciones a más bajo nivel, no se debe de tener implementada lógica de flujos distintos de negocio en sequences con IFs.

## 10. Selectores

A partir de la versión 20.4 de UiPath Studio se introdujo lo que llamamos la *Modern Design Experience*. Para ello, tenemos una nueva herramienta llamada el *Object Repository*, que permite mapear de una forma semiautomática los controles en distintas vistas de las distintas aplicaciones y crear repositorios agrupados por vistas de controles reutilizables como si de objetos se tratase. Es un nuevo paradigma de programación en UiPath que sustituye de una forma conceptual al clásico selector y que permite mejorar la eficiencia a la hora de programar. Además, la *Modern Design Experience* crea selectores más robustos y flexibles.

D.10.1: cualquier proyecto nuevo que se desarrolle se hará empleando la Modern Design Experience y empleando librerías de controles de aplicaciones pre-creadas. En el caso que no la haya, se crearía una nueva.

D.10.2: Los fuzzy selectors sólo se emplearán en los casos en los que el selector normal no sea estable. En la mayoría de los casos, el problema que soluciona el Fuzzy selector se puede solucionar retocando el selector clásico para tener en cuenta las ligeras variaciones.

D.10.3 Los selectores deben de ser robustos y específicos. Es preferente utilizar variables en el selector antes que hacerlo totalmente genérico, por ejemplo:

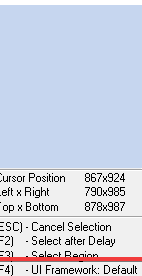
<web xxxx title=’Empleado 4848’> : Este selector es demasiado específico

<web xxxx title=’Empleado \*’> : Este selector funcionaría, pero puede dar falsos positivos

<web xxxx title=’Empleado {IdEmpleado} > : Este selector es robusto y específico para cada ejecución

D.10.4 Evitar usar el parámetro “idx” en un selector salvo que realmente no haya alternativa.

En ciertas ocasiones, dependiendo de la aplicación, podemos encontrar que el selector generador por defecto no aporta mucha información. En esos casos conviene siempre cambiar el “UI Framework” y generar de nuevo el selector. Pulsando F4 se puede cambiar el UI Framework cada vez que se intenta generar un selector:



Selector Clásico -> Imagen=CV Scope\* -> Fuzzy

\* El CV scope sólo se podrá usar en casos justificados, ver apartado de Actividades

## 11. Actividades

D.11.1: Se usará preferentemente actividades de la Modern Design Experience. El uso de actividades “Clásicas” deberá ser aprobado por el arquitecto para cada caso de uso

D.11.2 Las interacciones con aplicaciones (Clicks, Type etc.) Deben estar dentro de un contenedor “Use Application/Browser”.

D.11.3Las actividades de interacción con elementos (Click, Type) usarán preferentemente las propiedades de Simulate (O Chromium API en el caso de navegador Chrome)

D.11.4Los navegadores usarán preferentemente modo incógnito, salvo que por algún motivo la web consultada no funcione correctamente al usar el modo incógnito.

D.11.5El navegador a usar por defecto será Google Chrome

D.11.6: Será preceptivo priorizar el uso de los paquetes de actividades del feed oficial de UiPath, siempre que los mismos contengan la funcionalidad que necesitamos para nuestro desarrollo.

D.11.7: En el caso que no exista funcionalidad oficial por parte de UiPath, podremos emplear funcionalidad desarrollada por terceros, pero exclusivamente cuando la fuente esté almacenada en el feed del Marketplace de UiPath. El uso de librerías y actividades de este Marketplace deberá de ser aprobado por el arquitecto.

D.11.8: Limitar el uso de Invoke Code al mínimo posible. Sólo se debería de usar para hacer actividades de “programación pura” para, por ejemplo, transformar o adaptar datos en variables.

D.11.9: Se limitará el uso de las actividades de tipo CV (Computer Vision) para los casos en los que no haya otra opción. Dicho uso debe de ser aprobado por el arquitecto

D.11.10: Para cerrar aplicaciones web, el método preferido es un Use Application con “Open: Never, Close: Always” en lugar de un Kill process o un click a la “X” de la aplicación

## 12. Integración

Se establece la siguiente jerarquía de uso de actividades para integración con aplicaciones y servicios:

D.12.1 Actividades Integradas de UIPath Oficiales -> API (HTTP Request) -> Actividades Nativas de UIPath (por selector)-> Actividades nativas de UIPath (por Imagen)

Por ejemplo: Integración con Office 365: subir un archivo a un sharepoint y mover los que hubiera en una carpeta a otra distinta: Empezamos por el paquete oficial de actividades de O365 de UIpath, si nos faltara funcionalidad, intentamos usar la API oficial, si aun así nos falta funcionalidad, intentaríamos automatizar la aplicación (Web), primero por selectores, y luego por imágenes

## 13. Uso de BBDD

D.13.1 Los datos de cada transacción se almacenarán en el ítem de cola. Este ítem de cola se crea por el proceso colector. Además, se almacenarán los datos de negocio relativos a la transacción de forma paralela en una tabla en BBDD. Los datos estarán relacionados por medio del *ItemReference* de la cola, por lo que es necesario que este ItemReference sea un valor único.

D.13.2La conexión a BBDD se hace en la inicialización, y la variable de conexión se arrastra (dentro del Config) para usarla en las consultas necesarias a lo largo del proceso. Para usarla en las actividades simplemente la añadimos como *CType(in\_Config(“DBProject”), DatabaseConnection)*

D.13.3 Todos los procesos tendrán una tabla “Main” en la que se alojan los datos de negocio de la transacción. Además, podrán existir tablas adicionales de configuración.

D.13.4Toda la interacción por parte del robot con dichas tablas se hará mediante Procedures. Los procedures podrán devolver un DataTable (en la mayoría de los casos) o un valor simple (Bool, string, int)

No está permitido hacer Queries SQL directamente desde el robot.

Los procedures deberán de limitar siempre el número de resultados, tanto de filas como de columnas, para optimizar el funcionamiento del robot y BBDD

## 14. Excepciones y Error Handling

Las excepciones, siempre que empleemos el ExtendedREFramework, estarán controladas por el propio Framework. No obstante, tenemos que distinguir entre excepciones de sistema o de negocio, si es que hubiera diferencia entre ellas, y documentarlas en el log apropiadamente.

D.14.1: El “mensaje” de la excepción será única y exclusivamente el código de error. Este código de error se traducirá en el mensaje correspondiente a la hora de gestionar la excepción

D.14.2: Como buena práctica en el uso de esta actividad *Try Catch*, y especialmente para tratar de gestionar excepciones no contempladas, conviene poner en la parte *Catch* el tipo concreto de excepción si sabemos cuál es más probable (si estamos con ficheros, por ejemplo, la System.IO.IOException es la más probable), pero si existe la probabilidad de que haya otros tipos que no conozcamos, hay que poner, al final la System.Exception para que capture cualquier otra excepción. De ese modo, podríamos tratar a cada una de ellas de forma individualizada. Si no respetamos el orden descrito y ponemos System.Exception arriba, nunca pasará por el resto de las excepciones, ya que System.Exception es la más genérica. Los tipos de excepciones más comunes son:

* NullReferenceException
* IndexOutOfRangeException
* ArgumentException
* SelectorNotFoundException
* ImageIOperationException
* TextNotFoundException
* ApplicationException

Si precisamos de ejecutar siempre un código de reciclado o de puesta a punto de la aplicación o el proceso una vez terminemos de ejecutar el código o el tratamiento de la excepción, podremos emplear la parte *Finally* que a continuación de la ejecución del bloque realizará las tareas que queramos.

Si dentro de algún bloque de tipo *Try Catch* tenemos una excepción que debe bloquear la ejecución dada su severidad, podemos emplear la instrucción *Rethrow* que envía la excepción detectada “hacia arriba” y la propaga para que se capture en el siguiente bloque de código, fuera del *Try Catch* actual. Esto es especialmente importante para las excepciones de negocio y las de sistema que sean bloqueantes, pues es necesario que la excepción suba para que el Framework la gestione y pase al siguiente item

D.14.3: Solo emplearemos la propiedad *ContinueOnError* cuando sea estrictamente necesario, ya que ésta hará que cualquier excepción que se capture en el bloque de código concreto sea ignorada deliberadamente. Como buena práctica recomendamos limitar mucho su uso y deberíamos atenernos a supuestos como pueden ser:

* Durante el *Data Scraping*, ya que la actividad lanza una excepción en la última página si no se localiza el selector del botón siguiente.
* Cuando no nos interese capturar el error concreto, sino que el bloque de código lleve la lógica conveniente para controlar los problemas
* Cuando tenemos una secuencia opcional, por ejemplo, un popup de aviso que no salga siempre, y que, si no sale, no interrumpa el resto de la ejecución

D.14.4: El uso del *Global Exception Handler* estará limitado a procesos en los que, por criticidad temporal, sea necesario realizar algún tipo de alerta o aviso (mail, orchestrator, teams) en el preciso instante en el que se produzca un fallo.

En los siguientes casos se añadirá un **Raise Alert** para notificar en Orchestrator:

D.14.5 El framework contempla la posibilidad de parar la ejecución del proceso si encontramos más de N excepciones de sistema consecutivas. Dicho valor se aloja en el Config del proceso. El framework envía de forma automática un mail con la siguiente estructura si se supera el límite de excepciones:

**Subject**: [Nombre del Proceso]: Fallo masivo en [Nombre de aplicación]

**Body**: Durante la ejecución del proceso [Nombre del Proceso] se han producido [límite de errores] excepciones de sistema en la aplicación [Nombre de aplicación] por lo que se ha procedido a abortar el proceso. Por favor, compruebe la procedencia de estas excepciones y en caso de que fuese necesario recuerde eliminar las ejecuciones oportunas del proceso del scheduler.

### Excepciones inicialización

Se inicia un contador antes de cada login. Si durante el login se produce algún error, se debe reintentar hasta alcanzar un límite establecido. Este error se produce antes de recoger un ítem de la cola, por tanto, dicho ítem no se verá afectado.

Una vez superado el límite de reintentos, la ejecución se parará enviando el siguiente correo al responsable de la aplicación.

Subject: [Nombre del Proceso]: Fallo en el acceso a [Nombre de aplicación]

Body: No se ha logrado acceder a [Nombre de aplicación] tras alcanzar [límite de errores] intentos por lo que se ha detenido la ejecución del proceso [Nombre del Proceso]. Por favor, póngase en contacto con el administrador de la aplicación para resolver el problema cuanto antes.

### Excepciones por ítem

Las excepciones de negocio no se reintentarán de forma automática, pero existirá la posibilidad de forzar un reintento directamente en la cola a través de Orchestrator

Las excepciones de sistema se reintentarán el número de veces definido en la cola.

Existe una posibilidad adicional para el manejo de excepciones por ítem que deberá ser aprobado en cada caso. Las excepciones no se reportarán al instante, sino que el ítem se pospondrá para intentarlo otra vez más tarde, hasta un límite (de tiempo, o reintentos totales)

## Logotipo, nombre de la empresa Descripción generada automáticamente15. Cuentas de correo

D.15.1 Las actividades de correo necesitan un tratamiento especial ya que no se puede confiar completamente en los servidores. Introducirlas en un Try/Catch o Retry Scope para asegurar reintentos en caso de excepciones.

D.15.2 El "Account" en las actividades de Correo también es un elemento muy importante para especificar, ya que podría haber más de una configuración de cuenta en una máquina determinada.

## 16. Uso de APIs y HTTP Request

D.16.1 Las llamadas de API se harán en un Workflow específico para cada endpoint. No se podrán hacer más de una llamada (HTTP Request) en el mismo workflow.

D.16.2 Todos los workflows de API deberán de comprobar los status code devueltos y lanzar excepciones según el resultado

D.16.3 La construcción de los “Bodys” (JSON) para llamadas de API “desde 0” se harán dando valores a un diccionario que luego se serializa (Utilizando Newtonsoft.Json.Serialize). **No se debe de construir un string concatenado**

D.16.4 Es posible utilizar un template (XML o JSON, según API) el cuál se lee y luego se reemplazan los valores para generar los parámetros de la llamada. En este caso, NO se debe de modificar el texto directamente sino que se debe almacenar el template en la clase adecuada (XmlDocument/JObject) y manipularla.

## Logotipo, nombre de la empresa Descripción generada automáticamente17. Macros Excel

D.17.1 Sólo se hará uso de Macros Excel para los casos en los que previamente a la implementación del proceso TO BE exista dicha Macro y su implementación por medio de actividades de UIPath sea muy complicado.

D.17.2 En las invocaciones a macros indicar:

* + Log de inicio de lanzamiento indicando:
    - El fichero Excel que contiene la macro que se invoca
    - Nombre de la macro y parámetros que se pasan
  + Log de fin de lanzamiento
  + Invocación en modo VISIBLE si la macro proporciona alguna información útil.

## Logotipo, nombre de la empresa Descripción generada automáticamente18. Excel y Datatables

D.18.1 Para los casos en los que sólo hay que leer o escribir un datatable completo de/en un Excel conocido, utilizar las actividades del grupo Workbook. Reservar el uso de las actividades del Excel Scope para aquellos casos en los que necesitemos cambiar formatos, transformar o identificar el número de hojas usar macros, o en general, leer datos de un Excel en el que el formato no se asemeja directamente a un Datatable

D.18.2 Para seleccionar celdas o Rango dentro de un Excel utilizar el Select Range.

D.18.3Si se quiere automatizar visualmente algo de Excel, se debe priorizar los Hotkeys a los clicks. Sin embargo, automatizar algo visualmente en Excel debería ser siempre la última opción. Es preferible utilizar una macro de VBA antes que automatizar visualmente

D.18.4 Revisar que las fechas se filtren correctamente cuando se escriben en un Excel. Especial cuidado con el formato DD/MM/AAAA y MM/DD/AAAA, porque puede no dar error de fecha, pero no estar tratándolo correctamente. Por ejemplo: 01/02/2022, en cualquiera de los 2 formatos es correcto. Para evitar estos problemas, especialmente al escribir, es recomendable generar una plantilla de Excel “vacía” en la que ya se ha forzado previamente el formato de las celdas.

D.18.5Cuando se tenga que leer un fichero Excel de **solo lectura** y se utilice la actividad de workbook, habrá que hacer una copia en la carpeta de ejecuciones y leerlo desde ahí, para evitar problemas de fichero en uso.

D.18.6Es recomendable que las columnas donde se van a almacenar números (con o sin decimales) estén en formato “general”. Las columnas que vayan a almacenar fechas es recomendable que estén en formato “fecha” de Excel.

D.18.7Es recomendable hacer siempre un Filter Datatable para quitar valores que no tengan sentido, principalmente, valores de filas vacías.

D.18.8En la medida de lo posible, evitar formatos complicados para las tablas de Excel, especialmente formatos con celdas combinadas y/o tablas tipo “árbol” en las que hay campos vacíos pero que se sobreentiende que son “los del valor de arriba”

## 19. Multimáquina

D.19.1 Los procesos de tipo consumidor se desarrollarán de forma que puedan lanzarse de forma simultánea en múltiples máquinas. Esto también es aplicable para los procesos de tipo colector en los que se prevea un tiempo de funcionamiento elevado

* Para garantizar que un proceso funcione correctamente en distintas máquinas de forma simultánea, es necesario garantizar que se disponen de suficientes credenciales para las aplicaciones. Algunas aplicaciones permiten multisesión, pero en el caso de aplicaciones que no admiten multisesión es necesario tener una credencial distinta por cada máquina. En el caso de APIs también hay que tener en cuenta si hay límites de llamadas por tiempo (rate limit) si se comparte la conexión/usuario entre varias máquinas.
* Para los procesos de tipo colector es necesario garantizar que no se están introduciendo transacciones repetidas en la cola. La implementación dependerá de las aplicaciones y lógica de negocio del proceso

D.19.2 Los procesos de tipo Reporte sólo se deberán lanzar en una máquina de forma simultánea

## 20. Logs

D.20.1 Se debe de emplear la actividad *Log Message* indicando la severidad en la propiedad *Log Level* y el mensaje que queremos adjuntar a lo largo del proceso. No se puede usar la actividad WriteLine

Los índices de severidad serán:

* Fatal: cuando el sistema no pueda recuperarse o reintentar, por ejemplo, cuando una aplicación no abre. Un Log de tipo Fatal implicaría que hay que parar por completo la ejecución lo antes posible
* Error: cuando se trate de fallos no recuperables, como podría ser un dato no esperado y que el robot pueda continuar con el siguiente elemento.
* Warn: serán advertencias por las que el robot puede continuar, pero que recomendemos revisar el origen del problema. Ejemplo sería un dato que aun estando mal, se puede emplear.
* Info: estos serán los típicos logs de información, como por ejemplo cuando se entra o sale de un proceso, o para indicar el tipo de caso que se está procesando si existieran varias posibilidades.
* Trace: en este caso solo serán trazas de desarrolladores, las cuales no queremos que salgan en el proceso cuando se encuentre en producción.

D.20.2: los mensajes **no pueden contener información sensible en ningún caso y bajo ningún concepto**.

D.20.3: eliminar todas las trazas innecesarias del desarrollador para evitar el exceso de logs, hay que buscar un equilibrio mínimo donde la información que almacenemos sea suficiente para comprender qué ha pasado en caso de error. Esto es complicado de discernir, pero se puede hacer si se analiza el caso que estamos cubriendo.

D.20.4: Todos los workflows deben incluir un Log de tipo Info al principio del workflow y al final del estilo “Iniciando Workflow XXX” y “Terminando Workflow XXX”. En la versión más reciente de uipath, podemos utilizar directamente las propiedades de Invoke Workflow para ello

## 21. Comentarios y anotaciones

El uso de Comentarios en el código es obligatorio

D.21.1: El código debe estar debidamente explicado por medio de comentarios y anotaciones. Es especialmente importante añadir comentarios cuando se realiza una acción en un punto en el que se prepara algo para el resto del proceso y/o a simple vista parece que no tiene relación con el paso siguiente, por ejemplo: Al principio hacemos un Get Text para coger una Fecha, y esa Fecha no se usa en el resto del proceso hasta el último paso en el que se rellena en un campo de la aplicación.

D.21.2: todos los workflows deben llevar una anotación explicativa al principio. Esta anotación debe indicar qué hace el workflow, desde qué estado lo hace (ej. Desde qué menú en qué aplicación) y qué entradas y salidas (argumentos) tiene. Es especialmente importante en librerías ya que luego aparecen como anotación de la actividad, por ejemplo:

Text

Description automatically generated

Las anotaciones de principio de workflow siempre deben seguir la siguiente estructura:

[Descripción simple]

PRE: [Estado anterior, ej “ventana XX abierta”]

POST: [Estado posterior, ej “campos ventana XX rellenos con datos”]

INPUTS: [lista de argumentos de input con descripción]

OUTPUTS: [lista de argumentos de output con descripción]

SETTINGS: [lista de valores internos de configuración, SÓLO si es aplicable al workflow, principalmente para workflows de librerías]

D.21.3: Es obligatorio añadir siempre una pequeña anotación embebida en las siguientes actividades:

* **If**: Explicación simple de la condición. La única excepción es cuando la condición del If es una única variable booleana (por ejemplo, Element Exists inmediatamente seguido de If)
* **Regex**: Explicación (con palabras) de lo que se busca (Y/O reemplaza)
* **Filter Datatable**: Condiciones de filtrado
* **Assigns con LINQ**: Explicación (con palabras) de lo que se está filtrando/haciendo
* **Flow Decision**: Igual que el “If”

## 22. Configuraciones

D.22.1 En todos los procesos es necesario hacer uso del *Config.xlsx*

D.22.2 En la pestaña de *Assets* pondremos una clave y un nombre de *Asset* que tendremos que consumir desde el Orchestrator. Lo que finalmente nos vendrá será el valor almacenado en Orchestrator del *Asset*. La OrchestratorFolder irá vacía para assets específicos al proceso (=se coge el asset de la misma carpeta de ejecución), o el nombre de la carpeta “Global” si fuera un valor global

D.22.3 Las credenciales irán incluidas en la hoja de Credentials. El framework recogerá las credenciales al iniciar la ejecución y guardará los valores en la misma variable Config con la siguiente estructura:

Nombre del asset: CredencialMailRPA

Config(“CredencialMailRPA\_User”): String

Config(“CredencialMailRPA\_Pass”): SecureString

D.22.4: Los valores configuraciones elementales que se deban de cambiar a menudo (una vez a la semana o más) deben estar como Asset y no dentro del fichero de Config para facilitar el cambio directamente desde Orchestrator

D.22.5 Los valores de configuración de tipo relacional (tablas) se alojarán como una tabla de BBDD. El nombre de los Procedures usados para interactuar con dichas tablas estará en el Config.

D.22.6: No debemos tener contraseñas en el fichero de Config. Las credenciales se deben gestionar como Assets de Orchestrator

D.22.7: El fichero de Config **no** puede estar como ruta local de la máquina (Ej. C:\FicherosConfig\ConfigProceso1.xlsx), se alojará en una ruta compartida. Además, el proceso SIEMPRE debe tener un Config válido en Data\Config.xlsx. Este config será equivalente al Config de desarrollo, con los parámetros necesario para poder lanzar el proceso en desarrollo.

## 23. Control de versiones

D.23.1: Es obligatorio el uso de Git en cada proceso. Cada paquete de UIPath tendrá su propio repositorio)

D.23.2: La rama main/master alojará siempre el código que está funcionando en producción en ese momento. Cualquier nuevo desarrollo se hará sobre una rama de desarrollo con nombre dev-[Cambio]

D.23.3: el desarrollador ha de hacer como mínimo un *Push* diario de su trabajo, bajando previamente la última versión disponible del código fuente y verificando que no haya conflictos. De este modo siempre tendremos disponible la última versión del código si sucediera algo.

D.23.4: Se hará merge a main sólo cuando se haya puesto en producción y se confirme que funciona correctamente. El desarrollador creará la pull request e incluirá a los validadores obligatorios para que acepten la pull request

D.23.5: es esencial utilizar un comentario descriptivo a la hora de hacer commit. El comentario debe resumir los cambios que se han hecho. Ejemplo de comentario:

Autor: Jorge JCV

Aplicaciones: N/A

Versión: 2.0.0

Comentario: Actualizada lógica de filtrado de archivos Excel en carpeta compartida

**D.23.6**: Para llevar la trazabilidad de los commits que implementan cambios/defectos/evolutivos, se añade un labels al commit con el código del cambio (CMBXXXX, DFXXXX, EVOXXXX)

## 24. Rutas y URLs

D.24.1 Todas las Rutas y URLs tendrán la estructura “Ruta Base” + “Endpoint”. Las rutas se construirán a partir de estos dos componentes, alojados en los Assets correspondientes. La construcción de la ruta se hará en el InitAllSettings del framework

## 25. Workflow Analyzer

D.25.1 Se usará Workflow Analyzer con las reglas por defecto para validar errores comunes en el código, principalmente, nombres repetidos/por defecto de actividades, variables sin usar etc.

## QueueItems

D.26.1 Todos los procesos usarán ítems de cola para tener trazabilidad de las ejecuciones del proceso. Además, el transaction ítem contendrá (En el SpecificContent, y, si fuera necesario, Output), los parámetros identificativos para cada transacción

## 27. Triggers

D.27.1 Los triggers se crearán en función del proceso. Por defecto, los colectores se lanzarán por tiempo, y los consumidores se lanzarán por nº de ítems de cola, pero se tendrá que analizar el lanzamiento en cada caso.

El proceso de reporte se lanzará por tiempo, salvo en procesos en los que sea necesario que se lance inmediatamente después de terminar el consumidor. En estos casos, el proceso consumidor podrá lanzar directamente el Job del proceso reportador.

## 28. Assets

D.28.1 Se guardarán en assets:

* Credenciales
* Valores simples de configuración global como URLs
* Valores comunes a varias ejecuciones del mismo proceso (información “para el siguiente proceso”)

En los dos primeros casos, se crearán a mano, y los robots los consumirán directamente a través del REFramework

Para el tercer caso, los robots podrán escribir el valor (Set Asset) a lo largo del proceso. Por ejemplo, un asset de “Fecha Última ejecución” que escriba el robot al terminar su proceso, y que nos puede servir para filtrar ítems a partir de esa fecha

D.28.2 Todos los Assets y Credenciales deberán asignarse globalmente para todos los robots, y, si fuera necesario, se añadirán valores por máquina, por ejemplo para utilizar múltiples credenciales en distintos robots. El nombre de la credencial es la misma, pero se configura para que cada robot reciba la correcta.

D.28.3Los assets y credenciales que sean de uso común para cualquier proceso se deben almacenar en la carpeta base “RPA CoE”.

## 29.1. Publicación de procesos - SemVer

D.29.1 En el entorno de Preproducción la publicación de los procesos quedará a discreción del desarrollador, ya que se trata de un entorno precisamente para realizar cuantas pruebas sean necesarias.

La versión del proyecto que el desarrollador puede cambiar exclusivamente sin autorización es la Rev, es decir, el tercer número de la versión siendo *Major.Minor.Rev* el formato que adoptaremos para cada uno de los procesos, empezando por la versión 1.0.0 de todos ellos. Para un cambio de versión Major o Minor, tendrá que solicitar al CoE autorización.

Cambiaremos la versión *Rev* para cada sucesiva versión de pruebas que estemos publicando como desarrolladores.

Cambiaremos la versión *Minor* si los cambios de la versión anterior a la actual tienen algún impacto interesante y el CoE lo acepta, o bien el número de versiones *Rev* se ha disparado por encima de las 99. Esta es la única excepción para los desarrolladores. Por ejemplo, si tenemos la versión 1.0.99 y vamos a crear una nueva versión, ésta sería la versión 1.1.0.

Cambiaremos la versión *Major* exclusivamente bajo órdenes del CoE cuando el código que se haya añadido a la anterior versión del desarrollo tenga impacto suficientemente grande como para actualizar dicha numeración. Por ejemplo, si de la versión anterior a la actual se ha cambiado el comportamiento de alguno de los componentes por completo, se actualizará la versión *Major*, por ejemplo: si estamos en la 1.7.9, pasaríamos a la 2.0.0. Además, cualquier cambio que implique que las configuraciones anteriores no son válidas también implica un cambio de Major.

Ejemplo de X.X.Cambio : Arreglamos un selector que ha cambiado

Ejemplo de X.Cambio.X : Mejoramos la gestión de una excepción enviando un mail informativo y añadiendo detalles de la excepción

Ejemplo de Cambio.X.X : Pasamos de recibir el input desde un FTP a recibirlo por mail.

D.29.2 Es **obligatorio añadir *Release Notes***a todas y cada una de las versiones que hagamos, para tener trazabilidad en caso de problemas. La responsabilidad de añadir dichas *Release Notes* recae sobre el desarrollador que está publicando.

Tanto las release notes como los commits que se hagan en Git seguirán la siguiente plantilla:

**Autor**: Nombre + mail corporativo

**Aplicaciones**: Aplicaciones afectadas por esta reléase.

**Versión**: Versión del paquete uipath

**Comentario**: Comentario explicativo con los cambios

# Anexo: Recomendaciones generales

A1 La resolución de las máquinas se establece en 1920x1080. Todos los desarrollos se harán teniendo en cuenta esa resolución, esto es especialmente importante para los casos de automatización por imagen. Para garantizar que el desarrollador esté usando una máquian con dicha resolución, se utiliza mRemoteNG con la resolución previamente forzada

A2 La opción "Empty Field" en algunas actividades siempre debe estar marcada. Hace que la escritura en el campo establecido sea más estable. Esta opción puede que no funcione correctamente en algunos tipos de campos, por lo que es posible que sea necesario hacer previamente una secuencia manual de borrado del campo

A3 En las actividades de interacción con controles de interfaces de usuario utilice siempre el método “Simulate” /”Chromium API” (si es Google Chrome) siempre que sea posible. Utilice “WindowMessages” como segunda opción. Con esto las automatizaciones funcionan en background. “Hardware Events” es la que ofrece menos rendimiento y se ha de ejecutar en foreground.

A4 Para algunas propiedades en las actividades como Click, Type into, Get Text, Element Exists, etc. que son importantes para proporcionar robustez en el código, puede cumplirse:

 Cambiar la interacción del elemento de "Interactivo", por defecto, a "Completo". La actividad esperará a que el elemento UI esté completamente cargado antes de interactuar con él.

 Comprobar la propiedad de clic antes de escribir en las actividades proporciona robustez, lo que asegura escribir en el campo correcto de la caja de texto.

A5 No tener código que no se use, tanto a nivel de workflows que nunca se invoquen, como secuencias completas que estén comentadas dentro del código

A6 Evitar múltiples Assign Activities pues afectan mucho al rendimiento. Usar Diccionarios o Datatables para almacenar información

A7 El uso de Multiple Assign estará limitado a asignar valores ya conocidos por ejemplo a un diccionario o datarow. Es decir, debemos evitar hacer cálculos o concatenar métodos grandes en un multiple assgn ya que, si diera excepción, no se sabe cuál de las líneas de múltiple assign ha causado el problema.

A8 No crear múltiples IF anidados pues disminuyen significativamente la legibilidad. Utilizar otras actividades como ELSE IF o SWITCH, o implementar la lógica mediante un flowchart o state machine. En general, no se podrán tener más de 3 IFs anidados.

A9 No se deben de anidar más de 2 bucles (for each), si fuera necesario anidar más se hará como otro workflow.

Todos los bucles While deben de tener una condición de salida excepcional por contador o por tiempo para evitar bucles infinitos

A10 Evitar tildes, “ñ” y espacios en las secuencias en proyectos de UiPath. Esto es aplicable para todos los elementos de programación: Variables, Argumentos, nombres de workflows y archivos, etc.

A11 No dejar condiciones vacías. Por ejemplo, dentro de un if, si va por el else y no hace nada, poner un log también.

A12 En el caso de usar una actividad Switch, se usará el case Default preferentemente para el caso de excepción no controlada (valor no esperado).

A13 Se evitará el uso de la actividad “Delay”: Para no perjudicar el rendimiento de una solución automatizada, es recomendable la utilización de actividades que retrasan el proceso hasta que un elemento o imagen es encontrado, en vez de usar retrasos incondicionales (por ejemplo, un retraso siempre de 10 segundos). Esto también robustece la acción, ya que no se ejecutan etapas antes de tiempo y no aparecen, por tanto, errores.

A14 Uso del Retry Scope para las actividades que interactúan con Orquestador (colas y variables globales) o BBDD: Ante la posibilidad de desconexiones momentáneas con Orquestador se usará esta actividad para posibilitar varios reintentos.

A15 Si el proceso precisa de un navegador, preferiblemente utilizar el mismo navegador que utilice el usuario. Si el navegador es Chrome, utilizar Chromium api. Preferiblemente se lanzará el navegador siempre en Incógnito

A16 Maximizar la ventana al abrir un browser: Para que la ventana en el escritorio robot se asemeje lo máximo posible a la ventana con la que hemos trabajado durante el desarrollo utilizaremos la actividad de Maximize Window. Para otras aplicaciones también podamos mandar el comando “Windows+up” mediante la actividad “Send Hotkey”.

A17 Tiempos definidos en los Assets: En caso de necesitar modificar los tiempos de búsqueda, es recomendables usar los valores de los Asset definidos como Global\_Wait\_XXL, Global\_Wait\_XL, Global\_Wait\_L, Global\_Wait\_M, Global\_Wait\_S, Global\_Wait\_XS, Global\_Wait\_XXS.