Buenas Prácticas en la creación de Bots

Para crear bots de una forma óptima se puede revisar el siguiente manual de Buenas Prácticas.

1. Robot
    a. Descripción en Comandos
    b. Nombre del Robot
    c. Variables
        a. Nombre de Variables
        b. Limpiar Variables
        c. Reglas de Variables
        d. Decodificación
        e. Variables heredadas Padre-Hijo
    d. Credenciales
    e. Validaciones
    f. Cerrar Procesos
2. Web
    a. Abrir Navegador
    b. Cerrar Navegador
    c. Cambiar a ventana
    d. Esperar por Objeto
3. Sistema
    a. Esperar
4. Scripts
    a. Ejecuta otro script Rocketbot
    b. Detener este Robot
    c. Detener todo y salir
5. Desktop
    a. Mover Mouse
    b. Mouse Click
    c. Alerta
    d. Enviar Tecla
6. Lógica
    a. IF
    b. While
7. XLSX
    a. Nuevo xlsx
    b. Abrir xlsx
    c. Contar Filas
    d. Obtener celda
8. Virtualización
    a. Buscar por Imagen
    b. Esperar por Imagen
    c. Tips
9. MySQL
    a. Configurar conexión MySQL

Video de ayuda Buenas Prácticas:

Robot

Buenas prácticas

Descripción en Comandos

Agregar descripciones en los comandos nos ayuda a comprender cuál es la acción específica que este realiza, lo que es de suma importancia a la hora de modificar un robot, ya sea para cambiar funcionalidades o identificar posibles errores.

Nombre del Robot

Nunca utilizar **espacios**, **ñ**, ni **caracteres especiales** en el nombre del robot, si el nombre fuera compuesto por varias palabras puede utilizarse el siguiente formato: snake_case, camelCase o **PascalCase**.

Variables

Nombre de Variables

En la definición de variables nunca se deben utilizar espacios, ñ, o caracteres especiales, en caso de utilizar palabras compuestas se pueden utilizar guiones bajos o comenzar la siguiente palabra con Mayúscula como por ejemplo: snake_case, camelCase o PascalCase.

Limpiar Variables

Es buena práctica limpiar las variables que no requieren valores fijos, al principio de nuestro bot, es recomendable utilizar el comando "**Limpiar Variable(s)**" del **módulo System++**, ya que si limpiamos una por una tendremos un comando por cada variable, lo que se pueden reducir a sólo uno con la siguiente instrucción:

Reglas de Variable

String

Cuando queremos trabajar con variables de tipo String en Rocketbot, debemos pasarlas a comillas dobles, ejemplo.

Tenemos una variable llamada **{texto}** la cual contiene lo siguiente:

Si queremos por ejemplo modificar parte del texto, podemos utilizar el método de string[1] llamado replace(), en este caso si solo pasamos la variable, nos arrojará un error de sintaxis:

Debemos indicar que la variable **{texto}** la trabaje como un string, y para eso debemos pasarla entre comillas dobles **"{texto}"**.

Si además nuestra variable es un string pero contiene saltos de líneas, entonces debemos pasarla entre triple comillas dobles y espacios **"""{texto}"""**, esto permite que nuestra cadena de texto utilice más de una línea, por ejemplo, ahora nuestra variable **{texto}** se ve de la siguiente forma:

[1] Revisar métodos de string

#	Nombre	Datos
1	texto	Esto es un nuevo texto con saltos de línea

Si volvemos a ejecutar el comando anterior, nos arrojará un error:

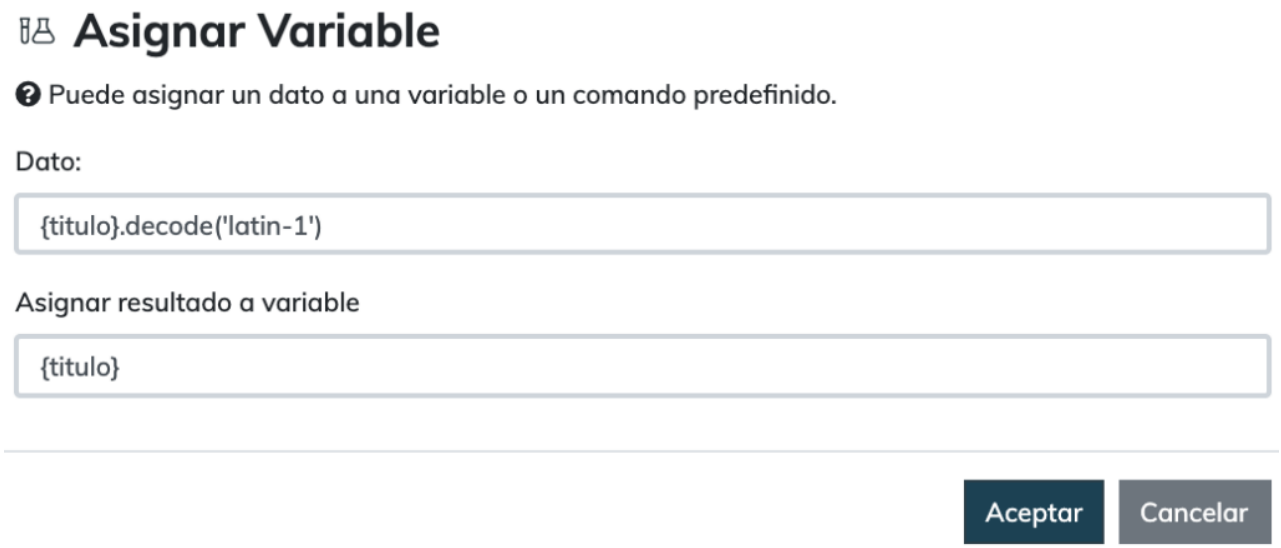
Al pasarla a triple comilla doble y espacios, ejecutará el método sin

problema:

Decodificación

Cuando tenemos variables codificadas, por ejemplo por haber obtenido el dato a través de JS, nos mostrará el texto de esta forma:

Para utilizar esta variable como STRING se debe utilizar {variable}.decode() o {variable}.decode('latin-1') en el comando **Asignar Variable**.



Quedará de esta forma y ya podemos utilizar la información correctamente:

Variables heredadas Padre-Hijo

Las variables del **PADRE** pueden ser utilizadas por todos los robots de la imagen, el **HIJO1** por ejemplo, no puede utilizar las variables de sus hermanos (**HIJO2–HIJO3**), el **NIETO1** puede usar las variables del **PADRE** y del **HIJO1**, pero no las del **HIJO2 – HIJO3**

Tiempos de ejecución (Esperar por Objeto)

En Studio (ambiente Desarrollo) el robot se ejecuta paso a paso con datos de debug que viajan al intérprete y regresan con datos de ejecución.
Esto hace que el comando tenga un tiempo de ejecución y milisegundos por cada comando para interpretar la respuesta. Significa que el robot en desarrollo es mucho más lento que en producción donde no necesita tiempos de debug.
Ej: En un robot extenso de manejo de Excel un robot tarda una hora en desarrollo pero solo 10 minutos en producción.
Para solventar esto y como parte de las buenas prácticas se debe usar los comandos **Esperar por Objeto**, de cada tipo de tecnología (Web, Desktop,

Virtualización, etc.).

Credenciales

Como buena práctica es importante utilizar un baúl de credenciales para obtener los datos y no dejarlos fijos.

Si tenemos Mac o Linux, podemos utilizar el siguiente módulo Credentials_, si estamos en Windows, debemos usar el siguiente: Windows Credential Manager.

Podemos encontrar más información acerca de cómo realizarlo, en la documentación de los manuales linkeados arriba.

Validaciones

Se deben validar todos los comandos y/o módulos que lo permitan, por ejemplo si es una conexión a una BD, Correo, FTP, Abrir un Archivo, etc. se debe retornar a una variable y tomar una decisión con un IF.

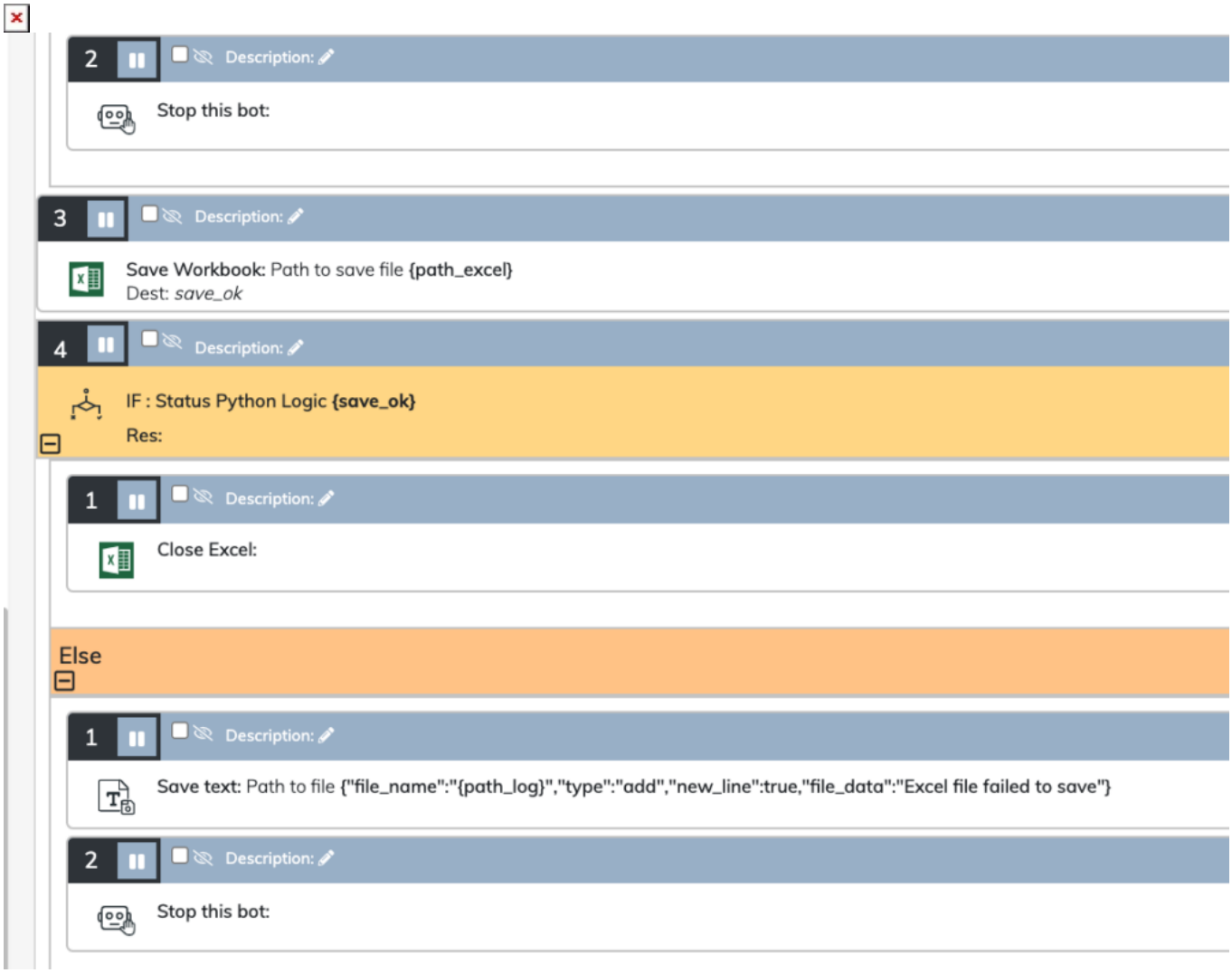
Esto nos permite poder controlar errores y tomar una decisión para que el robot no continúe ejecutándose cuando faltan partes del proceso o cuando no se realizó algo de forma correcta.

Ejemplo:

Crear un archivo excel y validar su correcta creación, si está ok, entonces se escribe en el archivo, sino, se deja un mensaje en el log y se detiene el robot.

Luego realizamos el guardado del archivo, y se valida, si se guardó de forma correcta, entonces cerramos la aplicación, sino, se deja un mensaje en el log

y se detiene el robot.



Cerrar Procesos

Todos los comandos y/o módulos que permitan cerrar conexiones, se deben agregar al proceso, por ejemplo, cerrar una conexión a una BD, a un FTP, Correo, Cerrar un archivo Excel, XML, etc.

Es necesario siempre cerrar todo lo que se abre con Rocketbot, ya que esto nos permite liberar recursos y evitar errores de ejecución, si el comando o módulo no permite cerrar la conexión o los procesos no son cerrados de forma correcta, se pueden utilizar los módulos killProcess y/o killApp para terminar de matar procesos zombies.

Ejemplo de un error:

Problema

No se realizó la ejecución de bots desde el Orquestador dado que un proceso no finalizó.

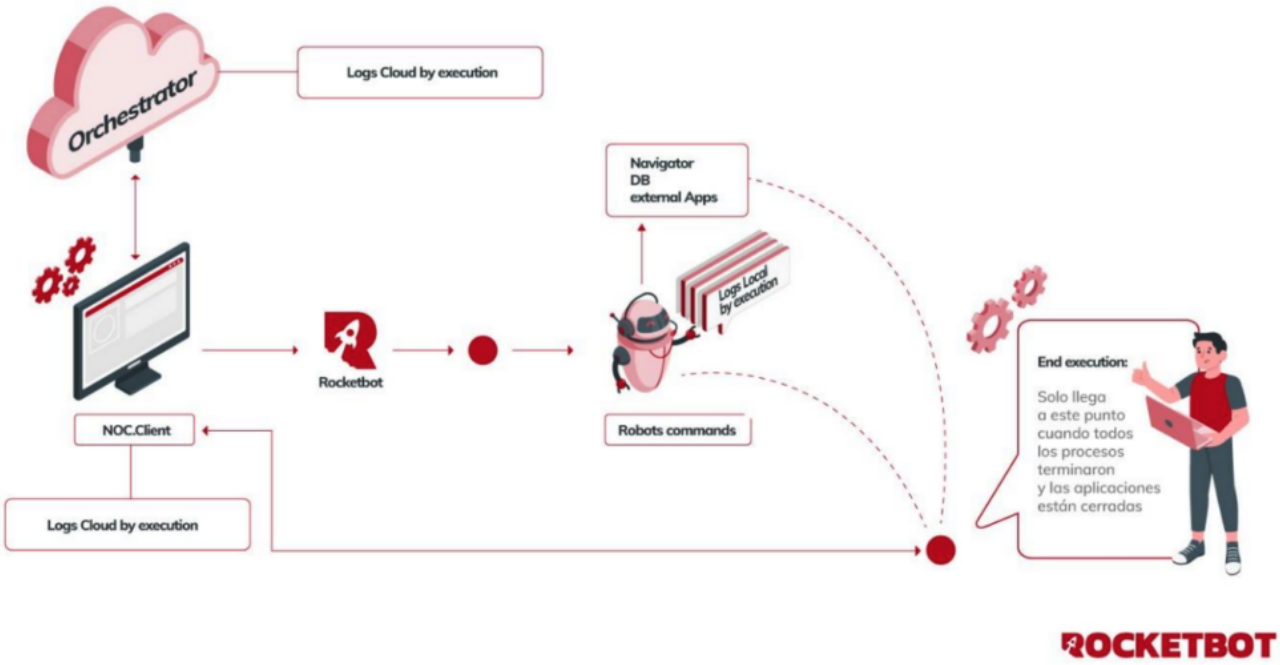
Causa

No llega la respuesta de proceso finalizado al Orquestador. Esto ocurre

cuando se trabaja con un proceso que queda en segundo plano o no finaliza al terminar el robot. Por ejemplo, abrir alguna aplicación.

Solución

Agregar el comando killProcess y/o killApp al final del flujo para que cierre cualquier proceso zombie que pudiera estar activo e impida notificar al Orquestador el término del proceso.



Web

Abrir navegador

Opciones:

- **Selecciona navegador:** Chrome/Firefox/IE
- **URL de Servidor:** Indicar la url de la página a la cual queremos acceder (Si no se elige ninguna abre por defecto https://www.google.com). Admite variables
- **Modo de Apertura:** Síncrona/Asíncrona.

Sincrónico es el método normal de apertura, el cual espera para continuar con los comandos, no se debe cambiar.

Buenas prácticas

Es óptimo agregar la url en una variable, pensando en escalabilidad, si en algún momento se requiere que la variable sea dinámica o si necesita cambiar,

lo ideal es que Rocketbot lea algún archivo de configuración y setear el valor a la variable para no estar abriendo y modificando el robot.

Error común

Los navegadores son controlados por web drivers, estos se encuentran dentro de la carpeta ***Rocketbot/drivers/{sistema operativo}/{browser}***, los mismos deben ser actualizados si se actualiza la versión del navegador y el robot comienza a presentar problemas.

Los drivers los puedes encontrar en https://www.seleniumhq.org/download/

Error de driver al intentar abrir navegador, esto sucede cuando el navegador se actualiza, para solucionarlo se debe actualizar el driver según la versión del browser.

Chrome: https://chromedriver.chromium.org/downloads
Mozilla: https://github.com/mozilla/geckodriver/releases/tag/v0.29.0

Descargar y descomprimir el driver, copiar el archivo en la siguiente ruta:

Rocketbot/drivers/{sistema operativo}/{browser}

Ej: Rocketbot/drivers/win/chrome

Cerrar Navegador

Nos permite cerrar el navegador abierto con Rocketbot, cierra todo incluyendo las pestañas abiertas.

Buenas prácticas

Siempre que realicemos un robot de tipo Web, y ya no necesitemos utilizar más el navegador, debemos agregar el comando **Cerrar Navegador**, ya que esto además de cerrar el Browser, mata el proceso del driver abierto con Rocketbot. El cual se abre en cada ejecución, por lo que si no se mata y se acumulan demasiados, podría consumir bastante recurso de la máquina.

> Todo lo que se abre con Rocketbot, es óptimo que se cierre al final del proceso o cuando ya no se utilice más, ej: navegador, conexión a BD, conexión a email, Excel, etc.

Cambiar a ventana

Cuando la página que estamos controlando abre otra ventana (al clickar un botón, enlace, etc.) este comando nos permite cambiarnos a la nueva pestaña y dejarla en foco para poder trabajar con los elementos.

En este ejemplo tenemos en foco la primera pestaña (google),

si queremos hacer un click en la página de Rocketbot (pestaña 2), por ejemplo al botón Descargar, obtendremos un error como este, donde nos indica que nose encuentra el elemento.

Debemos entonces primero cambiarnos de ventana y una vez quede en foco ya podremos realizar el click. Podemos cambiarnos indicando el handle o una de las siguientes opciones las cuales indican la posición de las pestañas abiertas:

Secuencia:

Buenas prácticas

Cada vez que nos cambiamos a otra ventana debemos utilizar el comando **Esperar por Objeto** con su respectivo **IF** de validación, ya que estamos ingresando a otra página y debemos asegurarnos de que cargó correctamente.

Esperar Por Objeto

Este comando permite esperar por un elemento en la web, dándole segundos máximos de espera hasta que nos retorna una respuesta (True o False, si lo encontró o no respectivamente). Se debe utilizar siempre que abrimos el navegador para confirmar si la página realmente cargó, también cuando clickeamos algún enlace o botón y nos abre otra pestaña, en general siempre que "cambiemos" de página, para así validar que cargó y los elementos ya existen.

Opciones:

- **Dato a buscar:** Valor del identificador del elemento.
- **Tipo de dato:** Tipo de identificador del elemento.
- **Esperar Antes:** Cantidad de segundos antes de que el comando "Esperar por Objeto" se ejecute. Es una espera fija, la cantidad de segundos que definamos se respetarán si o si, por lo cual se recomienda que sean tiempos bajos, ojalá no más de 2-5 segundos.
- **Acción Esperar máx:** Cantidad de segundos máxima que esperará el elemento. Es una espera variable, si ingresamos 10 segundos, no significa que los esperará si o si, sino que como máximo esperará esa cantidad, si nuestro elemento aparece antes en la página, entonces obtendremos antes nuestra respuesta y se continuará con el siguiente comando, en cambio, si nuestro

elemento no aparece, estará consultando hasta que se cumplan los 10 segundos y luego nos retornará un False en nuestra variable.
- **Esperar Después:** Cantidad de segundos que esperará luego de que se ejecute el comando “Esperar por Objeto” y continúe con el siguiente. Es una espera fija, la cantidad de segundos que definamos se respetarán si o si, por lo cual se recomienda que sean tiempos bajos, ojalá no más de 2-5 segundos.
- **Asignar resultado a variable:** Variable donde retornaremos el valor de si encontró o no el elemento dentro de los segundos asignados (True o False).

En este caso se está esperando 10 segundos a que aparezca el input de Google para realizar una búsqueda.

Buenas prácticas

Esperar Antes y **Esperar Después**: Son tiempos fijos, y son campos opcionales, en caso de utilizarlos, se deben indicar tiempos bajos (ej: 2-5 segundos), ya que utilizar esperas fijas no es buena práctica.

Acción Esperar máx: Es un tiempo variable, y es un campo obligatorio, siempre debe ir un valor ya que sino se indica, el comando no estará cumpliendo su función, como es un tiempo variable y no fijo, a diferencia de los dos inputs indicados arriba, el tiempo indicado debe ser más amplio.

Siempre que utilizamos **Esperar por Objeto** al inicio de nuestro flujo, se debe utilizar el comando **IF** del menú lógica, ya que tendremos una respuesta (True o False) en nuestra variable, en este ejemplo, llamada **{wait}**, con esa respuesta debemos tomar una decisión, qué hacer si es **True** (seguir el flujo normal) o qué hacer si es **False** (arrojar excepción, cerrar navegador, escribir en un log, informar por correo, etc.)

La secuencia sería la siguiente:

- Abrir la página de google.
- Esperar 10 segundos máximo a que el input de búsqueda cargue en la página.
- Tomar una decisión con el IF
  - True: Clickear el input de búsqueda, enviar el texto a buscar y realizar un Enter.
  - False: Cerrar el navegador y detener el robot.

Sistema

Esperar

Hace esperar al sistema por N segundos y luego sigue ejecutando los comandos.

Utilizado para casos especiales, **no se deben utilizar esperas fijas** ya que solo aumentarán el tiempo de ejecución del robot, para Web se debe utilizar **Esperar por Objeto** y para Desktop con Virtualización se debe utilizar **Esperar por Imagen**.

Scripts

Ejecuta otro script RocketBot

Este comando nos sirve para llamar un robot desde otro, donde podemos compartir variables y dividir tareas entre cada robot.

Nos mostrará todos los robot que tenemos en nuestra base de datos para ejecutarlos.

Buenas prácticas

Como buena práctica siempre es recomendado seccionar el flujo (proceso), en varios **mini-bots**. Esto facilita el poder modificar el proceso, revisar dónde está fallando, corregirlo y reutilizar comandos.

Ejemplo:

Un robot principal (**padre**), que abra un navegador con una url, luego un mini robot (**hijo_descarga**), que descargue un archivo y otro mini bot (**hijo_mail**) que tome el archivo y lo envíe por mail.

Robot **“padre”:**

1. Abre el navegador con una URL específica.
2. Maximiza la ventana.
3. Ejecuta un robot hijo (**hijo_descarga**).
4. Cierra el navegador.

Variables:

N/A

Robot “**hijo_descarga**”:

1. Cuenta la cantidad de archivos existentes en la carpeta de Descarga.
2. Realiza click en el botón Descargar de la página abierta.
3. Espera hasta que la descarga haya finalizado.
4. Ejecuta un robot hijo (hijo_mail)

Variables:

Robot “**hijo_mail**”:

1. Configura un correo Gmail.
2. Envía un correo adjuntando el archivo descargado en el robot anterior (utilizando la variable {path} del robot hijo_descarga).

Variables:

Detener este robot

Detiene la ejecución del robot actual y continúa con los siguientes, para utilizarlo solo se agrega el comando.

Buenas prácticas

Si tenemos una validación IF por ejemplo, para validar que la página cargó correctamente, este comando nos serviría para no continuar con las demás instrucciones. En caso de que algo falle, se puede detener el robot.

Ejemplo:

En este caso al caer en el Else, detendrá el Robot actual, por lo que no continuará con las siguientes instrucciones como la ejecución del bot hijo llamado Excel.

Detener todo y salir

Este comando detiene el Script en curso y sus padres.

Buenas prácticas

Misma función que el comando anterior, con la diferencia de que este detendrá tanto el robot actual como sus robots padres.

Desktop

Mover Mouse

Mueve el mouse hasta la posición indicada por coordenadas X e Y (ej: 1045,207).

Para obtener las coordenadas podemos utilizar el comando **Obtener coordenadas del mouse**.

Buena práctica

Este tipo de instrucciones no se deben utilizar en procesos **Web**, son para utilizarlas en automatizaciones de aplicaciones de escritorio, sin embargo si requerimos mover el mouse para realizar algún click, es mejor utilizar el comando **Click en Imagen**, el cual busca la imagen en toda la pantalla, por lo tanto si el objeto cambia de posición lo encontrará igualmente, no así **Mover Mouse**, que solamente irá a una posición específica.

Mouse Click

Hace click en donde está posicionado el Mouse, si queremos que lo realice en un lugar específico, debemos utilizar primero el comando **Mover Mouse** para indicarle la posición y una vez esté donde lo queremos, utilizamos **Mouse Click**.

Podemos utilizar:

- **left** -> para hacer click con el **botón izquierdo**
- **right** -> para hacer click con el **botón derecho**
- **middle** -> para hacer click con el **botón central del Mouse**

Buena práctica

Este tipo de instrucciones no se deben utilizar en procesos **Web**, son para utilizarlas en automatizaciones de aplicaciones de escritorio, sin embargo si

requerimos realizar algún click, es mejor utilizar el comando **Click en Imagen**, el cual busca la imagen en toda la pantalla, por lo tanto si el objeto cambia de posición lo encontrará igualmente, no así **Mouse Click**, que solamente lo hará en una posición específica.

Alerta

Este comando muestra un cartel de mensaje. Agregamos el comando: Mostrará la ventana de alerta: También le podemos pasar una variable para que nos muestre su valor. Nos mostrará la ventana de alerta con el valor que tiene la variable que ingresamos.

Buena práctica

Las alertas son sólo cuando estamos desarrollando un robot, pero se deben eliminar de un robot final, ya que éstas pausan la ejecución y el proceso no continuará hasta que sea aceptada.

Enviar Tecla

Al igual que el comando **Enviar Texto Web** éste sólo acepta una opción, si se escribe un texto (o una variable con texto) no se puede agregar una tecla especial, o no funcionará. Al revés lo mismo, si queremos enviar una tecla especial, no podemos también pasar un texto.

Buenas prácticas

No utilizar este comando en automatizaciones **Web**, en ese caso se debe utilizar **Enviar Texto Web**, ni en aplicaciones Office (Excel, Word, Powerpoint) para ellas se deben utilizar los menús o módulos respectivos.

En el input de Texto, podemos concatenar un texto y una tecla, de la siguiente forma:

Si se necesita repetir la misma tecla varias veces, también se puede indicar de esta forma:

Podemos encontrar más información respecto a combinaciones a utilizar en el siguiente link: pywinauto además podemos utilizar teclas predefinidas con el el módulo Keyboard.

Lógica

IF

La sentencia if se utiliza para ejecutar un bloque de código si, y sólo si, se cumple una determinada condición. Por lo tanto, IF es usado para la toma de decisiones.

Buenas prácticas

Nunca se debe dejar un IF vacío y poner solo la lógica en el Else, esto no es permitido en Python por lo tanto producirá fallos en nuestro robot.

Ejemplo:

La forma correcta es la siguiente, donde el Else si puede ir vacío:

While

Con la sentencia While podemos ejecutar un ciclo mientras se cumpla una condición, lo cual nos permite ejecutar instrucciones múltiples veces.

Buenas prácticas

La condición debe estar relacionada con una variable que tenga un cambio en la condición para que en algún momento esta se cumpla y se termine el ciclo, sinó se corre el riesgo de generar un bucle infinito.

XLSX

Si trabajamos con este menú no es necesario tener instalado Microsoft Excel, ya que no trabaja con la aplicación, abre o crea un archivo en segundo plano y no veremos que este se abra, todo pasará por debajo.

Nuevo xlsx

Nos permite crear un nuevo archivo XLSX en memoria, podemos indicar un Identificador (puede ser un número, letra o palabra) en el caso de que necesitemos crear y trabaja con más de un archivo xlsx abierto por Rocketbot, luego si necesitamos ir moviéndonos entre uno u otro, lo tendremos que realizar a través de su ID, también podemos indicar una variable, donde obtendremos un True si se puedo crear un nuevo archivo o False en caso contrario.

Buenas prácticas

Retornar a una variable para validar si se pudo abrir de forma correcta o no y con eso tomar una decisión a través del comando IF.

Ejemplo:

Abrir xlsx

Nos permite abrir un archivo XLSX en memoria, debemos indicar la ruta donde se encuentra, y al igual que el comando anterior también podemos indicar un Identificador en el caso que necesitemos abrir más de un archivo, además de la variable donde obtendremos un True si se pudo abrir de forma correcta, o un False en caso contrario.

Buenas prácticas

Retornar a una variable para validar si se pudo abrir de forma correcta o no y con eso tomar una decisión a través del comando IF.

Ejemplo:

Contar Filas

Este comando nos permite obtener la cantidad total de filas con información del archivo xlsx. Por defecto contará las filas de la primera hoja, si necesitamos trabajar con otra, primero debemos cambiarnos con el comando Cambiar de Hoja.

Buenas prácticas

Siempre luego de Abrir un archivo xlsx, debemos contar la cantidad de filas que tiene, ya que en el caso de que la cantidad sea variable y en otra ocasión el archivo tenga más o menos filas, es decir, en otra ocasión el archivo tenga más o menos filas tendríamos que estar modificando el comando **Obtener celda** de forma manual, lo cual no es óptimo, por lo tanto siempre se debe trabajar con una variable que contenga la cantidad de filas.

Obtener celda

Este comando nos permite obtener una celda o un rango de celdas y almacenar la información en una variable, al obtener un rango nos devolverá una lista o una lista de listas.

Buenas prácticas

Cuando trabajamos con un rango (ej: A1:F3) no debemos pasar un número fijo al final de este, sino, debemos utilizar el comando **Contar Filas** y asignar el valor a una variable, esta variable es la que se debe pasar al final del rango, ej: **A1:F{filas}**, esto en el caso de que la cantidad de datos en el Excel sea variable, para no estar modificando el comando cada vez que la cantidad de filas cambia, se debe utilizar esta buena práctica.

Virtualización

Este menú es nuestra última opción para realizar una automatización, el orden de prioridad sería:

1. Comandos nativos y/o módulos para Web, Office, Email, Base de datos, SAP, FTP, Archivos, etc.
2. Si necesitamos controlar una aplicación de escritorio, debemos primero intentar con los grabadores, Desktop Recorder o Java Recorder, si ninguno de los dos nos sirve, entonces la última opción será Virtualización ya que se deben considerar ciertas cosas como por ejemplo:
   a. La resolución del PC afectará al bot, por ejemplo, si se desarrolla un bot en X computador y luego se pasa al computador del cliente o donde se ejecutará finalmente, lo más probable es que no funcione correctamente, ya que se tienen resoluciones y componentes distintos.
   b. Las imágenes siempre deben estar enfrente para que el bot las reconozca, esto quiere decir que una persona no puede utilizar el computador al mismo tiempo que el robot, por lo tanto no se podrán realizar ejecuciones en paralelo.
   c. La pantalla no se puede bloquear, por lo que eso se debe desactivar o realizar ajustes para que no suceda.

> **Considerar:** Al trabajar con Virtualización, al sacar una captura de pantalla con los comandos del menú que lo permiten, siempre tomará la pantalla principal por defecto.

Buscar por Imagen

Busca una imagen de referencia en la pantalla y almacena el resultado en una variable retornando True si la encontró dentro del tiempo definido o False en caso contrario.

Buenas prácticas

Al igual que el comando **Esperar por Objeto** para la parte web, en Virtualización debemos utilizar este comando para asegurarnos de que el elemento con el cual queremos trabajar, existe, y luego tomar una decisión

con un **IF**, si nos retorna **True**, osea que sí lo encontró, entonces dentro de ese bloque realizamos nuestro flujo, si nos retorna **False**, osea que no lo encontró entonces debemos decidir qué hacer.

Esperar por Imagen

Espera una imagen de referencia en la pantalla y almacena en una variable si está visible o no.

Realiza la misma función que el comando **Buscar por imagen.**

Buenas prácticas

Al igual que el comando **Esperar por Objeto** para la parte web, en Virtualización debemos utilizar este comando para asegurarnos de que el elemento con el cual queremos trabajar, existe, y luego tomar una decisión con un **IF**, si nos retorna **True**, osea que sí lo encontró, entonces dentro de ese bloque realizamos nuestro flujo, si nos retorna **False**, osea que no lo encontró entonces debemos decidir qué hacer.

Tips

Windows

Estos son algunos casos donde los comandos de imagen
no funcionen correctamente, por lo tanto se debe revisar resolución o ejecutar Rocketbot como administrador.

1. **Si no realiza los click en las imágenes**
    a. Instalar lo siguiente:
        i. NET Framework: Download .NET Framework | Free official downloads
        ii. Visual C++: Descargas más recientes compatibles de Visual C++
    b. Revisar resolución
        i. Si la configuración de pantalla está aumentada, dejarla al 100% y probar.

> **Importante:** Si se realizan cambios de escala, puede ser necesario cerrar sesión y volver a abrir para que los cambios se guarden.

2. Si realiza click en ciertas imágenes pero no en todas.

Por ejemplo realiza click en elementos del escritorio o barra de tareas pero no en una aplicación de escritorio interna, además al intentar abrir la aplicación con Rocketbot aparece lo siguiente:

Ejecutar Rocketbot como administrador y probar.

3. Si realiza click en otra parte

Especificar alguna referencia, por ejemplo si tenemos dos elementos iguales, debemos marcar como referencia algo que no se repita y como foco el lugar donde realizar el click.

Por ejemplo acá seleccionamos como referencia el label Username el cual no se repite, y el foco es el input, lo que hará será calcular las coordenadas desde la imagen de referencia hasta la indicada en foco y hará click a lo que se encuentre en esa posición.

> Si sigue realizando el click en otra parte, y se está trabajando desde un escritorio remoto, debemos cambiar la resolución antes de ingresar al servidor.

Por ejemplo, el foco (donde hacer el click) es el ícono de Rocketbot, pero el click lo realiza en el punto rojo:

Configuración:

4. Si sigue sin encontrar las imágenes y es un Windows Enterprise (win 7 o win 10), revisar si está la siguiente .dll sino, agregarla:

- DLL: [api-ms-win-downlevel-shlwapi-l1-1-0.dll]

5. Virtualización en Windows server 2012 mediante rdp

En caso que las funciones de virtualización para “click en imagen” o “esperar por imagen” no funcionara correctamente podemos revisar lo siguiente:

Que no esté habilitada la opción de cambiar el nivel de escala:

Adicional a esto verificar que esté instalada las características dentro de “Interfaz de usuario e Infraestructura” en especial Experiencia de escritorio:

macOS

Permiso denegado al hacer OCR

Solución:

1. Abrir el terminal e ir a la ruta indicada en el error, ej: **cd /Users/usuario/Desktop/Rocketbot/drivers/mac/tesseract**
2. Escribir el comando **ls-l** para visualizar los permisos.
3. Veremos que en este caso no tiene permisos de ejecución:

4. Le damos los permisos con uno de los siguiente comandos:

- ejecución para todos los usuarios: **chmod 777 tesseract**
- ejecución sólo para el owner: **chmod 744 tesseract**

- Volvemos a ejecutar un **ls -l** para revisar los cambios:

- Ahora ejecutamos nuevamente el comando de OCR

MySQL

Configurar conexión MySQL

Para conectarnos con MySQL, debemos indicar la URL del servidor, puerto, nombre de la BD, usuario y contraseña, además podemos retornar el resultado a una variable donde obtendremos un True si la conexión fue exitosa, o un False, en caso contrario y podemos utilizar un ID en caso de que nos conectemos a varias Bases de datos MySQL.

Buenas prácticas

Siempre agregar una variable para retornar el status de conexión y con eso validar a través del comando IF para tomar una decisión.

Consulta MySQL

Realiza una consulta MySQL (Select, insert, delete, update), el resultado lo

retornamos en una variable para posteriormente trabajar con la información.

Buenas prácticas

En el ambiente de Desarrollo de Rocketbot las consultas se deben realizar con límites y no traer todos los registros, ya que si son demasiados, Rocketbot puede detenerse por consumo de memoria, esto para realizar pruebas, ya en el ambiente de producción igualmente es recomendable seccionar la cantidad de registros consultados.

Email

SMTP – IMAP

Configurar Servidor

Configura un servidor SMTP e IMAP con SSL para enviar y recibir correo. Al abrir el comando, por defecto vienen los servidores y puertos de gmail, debemos añadir el correo y su contraseña.

Buenas prácticas

Variables:

Es buena práctica, utilizar variables tanto en el nombre de usuario como en el caso de la contraseña y no escribirlos directamente en el comando. También debemos recordar siempre borrar las credenciales si por alguna razón tuvieras que compartir el robot.

Conexión:

Siempre se debe validar la correcta conexión de la cuenta de correo, en el caso de los **módulos**, éstos permiten validar la conexión retornando el resultado a una variable, por lo tanto esto se debe validar con un IF.

Ejemplo: 

Una vez que ya no utilizaremos nada respecto a correos, debemos cerrar la conexión con el comando “**Cerrar Conexión**” del módulo que estemos utilizando.

Módulos:

Se recomienda utilizar los módulos específicos para los distintos tipos de servidores de correo, pero si se requiere utilizar uno corporativo o que por el momento no tenga su respectivo módulo, es óptimo utilizar este comando para configurar y complementarlo con el módulo Email Advanced en vez de utilizar los comandos nativos que son más limitados.

Listar todos los email IMAP

Este comando nos permite obtener una lista de todos los email que coincidan con el filtro[2] indicado, debemos retornar el resultado a una variable para trabajar con la información.

Buenas prácticas

Una vez que obtenemos el resultado de listar email es necesario decodificar los datos.

[2] Filtros de email

La decodificación se realiza con el comando Asignar Variable y la función de Python decode().

Ejemplo:

Quedará de la siguiente forma:

> Si queremos realizar el decode para toda la lista obtenida, debemos recorrerla con un For.

Listar email nuevos IMAP

Este comando nos permite obtener una lista de todos los email que coincidan con el filtro[3] indicado, la diferencia con el comando anterior es que este nos traerá los id's sólo de los correos que no hayan sido leídos aún. Debemos retornar el resultado a una variable para trabajar con la información.

[3] Filtros de email

Al igual que el comando anterior, también debemos decodificar los id's entregados.

Integración con Aplicaciones

Excel

Si trabajamos con este menú si es necesario tener instalado Microsoft Excel,

ya que trabaja con la aplicación, abre o crea un archivo en primer plano y veremos que este se abre.

Nuevo Libro

Nos permite crear un nuevo archivo Excel, podemos indicar un Identificador (puede ser un número, letra o palabra) en el caso de que necesitemos crear y/o trabaja con más de un archivo Excel abierto por Rocketbot, luego si necesitamos ir moviéndonos entre uno u otro, lo tendremos que realizar a través de su ID, también podemos indicar una variable, donde obtendremos un True si se puedo crear un nuevo Excel o False en caso contrario.

Buenas prácticas

Retornar a una variable para validar si se pudo abrir de forma correcta o no y con eso tomar una decisión a través del comando IF.

Ejemplo:

Abrir archivo

Nos permite abrir un archivo Excel existente, debemos indicar la ruta donde se encuentra, y al igual que el comando anterior también podemos indicar un Identificador en el caso que necesitemos abrir más de un archivo, además de la variable donde obtendremos un True si se pudo abrir de forma correcta, o un False en caso contrario.

Buenas prácticas

Retornar a una variable para validar si se pudo abrir de forma correcta o no y con eso tomar una decisión a través del comando IF.

Ejemplo:

Guardar Archivo

Este comando guarda el archivo excel en la ruta que especifiquemos (usando el botón buscar o pasando una variable), es posible guardar el resultado de la operación en una variable, donde obtendremos un True si fue exitosa o un False en caso contrario.

Buenas prácticas

Retornar a una variable para validar si se pudo guardar de forma correcta o

no y con eso tomar una decisión a través del comando IF.

Ejemplo:

Contar Filas

Este comando nos permite obtener la cantidad total de filas con información del archivo Excel. Por defecto contará las filas de la primera hoja, si necesitamos trabajar con otra, primero debemos cambiarnos con el comando **Cambiar de Hoja**.

Considerar que sólo contará la cantidad en la columna A, por lo tanto si necesitamos cambiar de columna podemos utilizar el comando Contar Filas del módulo Advanced Excel.

Buenas prácticas

Siempre luego de Abrir un archivo Excel, debemos contar la cantidad de filas que tiene, ya que en el caso de que la cantidad sea variable, es decir, en otra ocasión el archivo tenga más o menos filas tendríamos que estar modificando el comando **Obtener celda** de forma manual, lo cual no es óptimo, por lo tanto siempre se debe trabajar con una variable que contenga la cantidad de filas.

Obtener celda

Este comando nos permite obtener una celda o un rango de celdas y almacenar la información en una variable, al obtener un rango nos devolverá una lista o una lista de listas.

Buenas prácticas

Cuando trabajamos con un rango (ej: A1:F3) no debemos pasar un número fijo al final de este, sino, debemos utilizar el comando **Contar Filas** y asignar el valor a una variable, esta variable es la que se debe pasar al final del rango, ej: **A1:F{filas}**, esto en el caso de que la cantidad de datos en el Excel sea variable, para no estar modificando el comando cada vez que la cantidad de filas cambia, se debe utilizar esta buena práctica.

Cerrar Excel

Este comando cerrará todos los Excel abiertos, independientemente de si fueron abiertos por Rocketbot o no, ya que mata la aplicación de Excel.

Si se requiere cerrar solo el archivo abierto pro Rocketbot, podemos utilizar el comando Cerrar XLSX del módulo Advanced Excel.

Buenas prácticas

Una vez que dejemos de utilizar el Excel debemos cerrarlo, sino el proceso quedará abierto y al acumular demasiados podría afectar en algo el funcionamiento correcto del bot.

Logs en Rocketbot

Los logs se encuentran en la ruta:
Rocketbot/logs/YYMMDD/logfile_YYMMDD_HHMMSS.log

Estructura de Logs de Rocketbot

- Fecha: formato yyyy-mm-dd
- Hora: formato H:M:S.f
- mod/core: ej: Rocketbot, script, system
- Tipo de log: ej: INFO, ERROR, EXCEPTION
- Comando/módulo: ej: clcikWeb, for, alert, module – {module_name: “System++”}

Ejemplo: 

Visualizar errores en Rocketbot

En el Studio, cuando un comando falle, se mostrará en color rojo y además indicará el texto del error, también podemos complementar la información revisando la consola de Rocketbot, la cual nos irá mostrando la información en tiempo real, o también revisando el archivo generado en la carpeta logs.

Si obtenemos un error en algún módulo, en la consola podremos ver la línea exacta donde ocurrió.

Ejemplo:

Recomendaciones para generar Logs.

Revisar en el momento de desarrollo todos los errores posibles ya que en el Studio la interfaz muestra todos los errores en forma gráfica. En producción no existe la interfaz de desarrollo por lo que no se muestra ningún error en forma gráfica, por lo tanto se recomienda programar acciones para reportar anomalías en las ejecuciones.

Se recomienda incluir información como Fecha, Hora, BOT, Acción y Resultado de la acción, que sean compatibles con las buenas prácticas de monitoreo y supervisión a través de visualizadores externos como Power BI, Kibana, etc. a través de correo electrónico, mensajes de texto, etc.

Robots en Paralelo

El siguiente cuadro grafica, cuándo y en qué condiciones se pueden ejecutar robots en paralelo.

Tipo de aplicación a automatizar	Posibilidad de implementar Bots en paralelo	Requisitos	Solución Alternativa
Sistemas WEB internos	**SI**	El sistema debe permitir correr múltiples sesiones por equipo	
Sistemas WEB externos	**SI**		
Bases de datos	**SI**		
APIs	**SI**		
Correo	**SI**		
SAP con GUI Scripting	**SI**	Habilitar en SAP GUI Scripting y una sesión por Bot	
Aplicación Java Cliente Servidor	**Probable**	-La aplicación debe permitir correr múltiples sesiones en la misma máquina. -Cada BOT requiere tener una cuenta de acceso. -La aplicación cliente debe ser una aplicación Java(basada en swing y awt) que se pueda grabar en Java Recorder	
Aplicación Windows Cliente Servidor	**SI**	-La aplicación cliente debe permitir correr múltiples sesiones en la misma máquina. -Cada BOT requiere tener una cuenta de acceso. -La aplicación cliente debe ser una aplicación Windows(basada en C# o .net)que se pueda grabar con Desktop Recorder Rocketbot	
Microsoft Word	**SI**	Solo si los BOTs trabajan sobre Archivos Word diferentes	

Microsoft Excel	**Si**	Solo si los BOTs trabajan sobre Archivos Excel diferentes	
Aplicación que obligue al uso de tratamiento de imágenes(OCR) o virtualización para su automatización	**No**	No se puede automatizar en paralelo sobre una misma máquina	Usar VMs sobre un mismo Servidor o equipos físicos. Con licencias individuales Onpremise S(Si requiere usar Xperience).

Escritorio Remoto

No se recomienda trabajar desde el escritorio remoto a menos que sea necesario, por ejemplo si es una web y podemos acceder desde fuera, entonces lo óptimo es realizar el bot desde nuestros computadores, si en cambio el sistema que queremos automatizar está en las dependencias del cliente, entonces tendremos que conectarnos.

Al trabajar desde un Escritorio Remoto podemos presentar lentitud en las acciones, si a esto sumamos conexión a través de alguna VPN y una velocidad de internet más lenta, entonces la lentitud puede aumentar. Esto afecta al tiempo de desarrollo de un bot, ya que al intentar mover un comando a otra posición, el delay de esta acción puede tardar.

Buenas prácticas

Si presentamos lentitud en las acciones, se recomienda primero que todo, definir bien el proceso para no estar modificando cada vez y tener que arrastrar los comandos desde el final hacia la posición deseada, también dividir aún más el flujo para trabajar con mini bots que contengan una cantidad no mayor a 40 instrucciones.

Para arrastrar comandos y empezar a subirlos, se debe posicionar entre el término del menú superior y el inicio del Dashboard (marcado en el recuadro rojo de la imagen de abajo)

Al dejar el comando en esa posición, sin soltar el mouse, empezará a hacer scroll y nuestra instrucción irá subiendo.

Y en el caso contrario, si queremos bajar una instrucción, el scroll lo realizará posicionando el comando entre el final del dashboard y el inicio del footer