Rocketbot Studio : Conflictos de Librerías

El empleo de diversas librerías en Rocketbot puede dar lugar a incompatibilidades si las versiones requeridas por los módulos personalizados no coinciden con las que ofrece la plataforma. Este informe examina las razones detrás de estos conflictos y sugiere soluciones para solucionarlos.

Importante

Es fundamental considerar la versión de Python utilizada al instalar librerías en Rocketbot. La versión 2020 de Rocketbot opera con Python 3.6.8 de 32 bits, mientras que a partir de la versión 2023 se utiliza Python 3.10 de 64 bits.

En caso de ejecutar módulos mediante un proceso externo, la librería correspondiente debe estar instalada en el entorno desde el cual se realiza la ejecución.
Esto permite operar dentro del entorno adecuado, evitar incompatibilidades y asegurar la correcta recepción de la información en Rocketbot.

Causas de los Conflictos

Los conflictos pueden surgir por diversas razones, como la existencia de diferentes versiones de librerías que no coinciden con las de Rocketbot. También pueden presentarse problemas de compatibilidad entre dependencias, ya que múltiples versiones de una misma librería pueden causar errores. Además, la instalación de versiones adicionales puede interferir con las librerías predeterminadas de Rocketbot.

Soluciones Recomendadas

Renombrar el Archivo de la Librería:

En casos donde se vea errores como el siguiente:

typing_extensions es una librería que viene compilada con Rocketbot y este error suele darse porque viene con una versión más antigua que la instalada para el script, por lo que al ejecutar otra librería que tiene como dependencia typing_extensions, rompe.

En estos casos se recomienda indicar la ruta hacia la lib que instalaron o si el error persiste como workaround se puede renombrar la carpeta de la librería importada o el nombre del archivo.py por uno diferente, por ejemplo en lugar de llamarlo **typing_extensions.py** lo llamaremos **new_typing_extensions.py**. En tu código deberás encontrar los imports que hagan tu scripts o las dependencias que importas cambiando lo siguiente:

import typing_extensions por **import new_typing_extensions as typing_extensions** (así se mantiene el nombre original)

Y también (teniendo en cuenta el error del ejemplo):

from typing_extensions import deprecated por **from new_typing_extensions import deprecated**

Con estos cambios, el script ejecutado por Rocketbot entenderá que no debe utilizar typing_extensions que viene nativamente con Rocketbot, sino **new_typing_extensions** instalado manualmente por nosotros.

Este ejemplo se puede aplicar tanto a archivos .py como a carpetas de librerías. Por ejemplo si queremos utilizar una nueva versión de la librería pandas, podemos descargarla

localmente, renombrar la carpeta a new_pandas y luego en nuestro script la importamos con ese nombre.

Uso de Procesos Externos:

En los casos donde la solución anterior no funcione, se sugiere ejecutar el proceso mediante popen o a través de un webhook. Esto permite el uso de la versión de Python y librerías del ambiente donde se ejecute en vez de usar los de Rocketbot.

Popen

Para ejecutar su script mediante un Popen realice lo siguiente en un comando Ejecutar Python:

```
from subprocess import Popen, PIPE
process = subprocess.Popen(
          ['python', 'ruta/a/su/script.py', 'arg1', 'arg2',
'argN'],
        stdout=subprocess.PIPE,
        stderr=subprocess.PIPE
    )
out, err = process.communicate()
print(out.decode())
```

En su script imprima en consola para obtener lo que necesita en Rocketbot.
En el ejemplo se ejecuta un script donde se le envían argumentos (que bien podrían ser datos de variables de Rocketbot). Luego en el script que está ejecutando toma los argumentos mediante: **sys.argv**
Esto trae cada valor de la lista que se le pasa al popen: **sys.argv[0]** traería python, **sys.argv[1]** ruta_script.py, **sys.argv[2]** arg1 y así según se necesite.
Es opcional para el caso en que se necesite enviar datos de Rocketbot al script.

Webhook

Al levantar un puerto local se queda en pausa a la escucha de algún evento POST o GET.

- Es un módulo que se descarga del market, el manual se encuentra en la carpeta example del módulo, en la ruta Rocketobot/modules/Webhook
- Puedes levantar un endpoint propio donde enviar información y que el robot escuche o usar ngrok para levantar uno (pasos en el manual).
- Al ejecutar el comando de 'Crear webhook', Rocketbot se pausará esperando que se consulte la url configurada en el comando, ya sea con una petición GET o POST.

En el contexto de conflicto de librerías, un workaround utilizando Webhook es crear un .bat de la siguiente manera:

C:/Python312/python.exe "C:\Users\user\Desktop\archivo.py"

Cambiando el path que está entre comillas por el path hacia el script a ejecutar y el primer path por el path hacia el ejecutable de Python en el ambiente a utilizar.

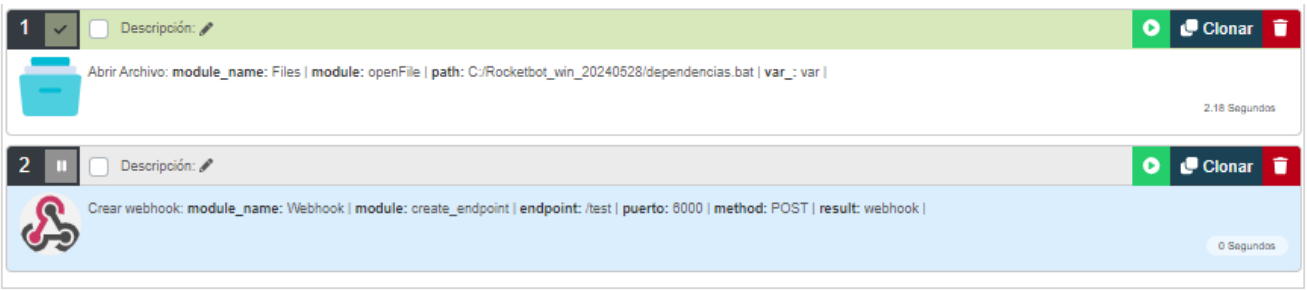
Dentro del archivo.py que contiene el script a ejecutar se tiene que agregar al principio las importaciones de request y un sleep que dará tiempo a ejecutar el comando de webhook. Al final del script se tiene que agregar la petición post para recibir la información mediante webhook al endpoint levantado:

```
import requests
from time import sleep
sleep(10)
(acá va el código)
requests.post("http://127.0.0.1:6000/test",
data={"var1":"data1", "var2": "data2"})
```

Cambiando 127.0.0.1:6000 por el host y puerto elegido. El parámetro data en este ejemplo es opcional, se puede utilizar

si se requiere enviar información a Rocketbot.

En el bot se tienen que crear los siguientes comandos uno seguido del otro:



El comando de webhook se tiene que ejecutar dentro de los segundos declarados en el sleep del archivo.py, en este ejemplo se tiene que ejecutar antes de los 10 segundos.

Rocketbot Studio : Migrar bots con Virtualizacion

Pasos a Seguir para la Migración de Bots de Virtualización

Virtualización en la Máquina Destino

Es recomendable realizar la virtualización directamente en la máquina o computadora final. Esto se debe a que la resolución de pantalla puede variar, lo que podría requerir ajustes en la configuración, similar a la máquina donde se realizó la virtualización inicialmente.

Consistencia de Imágenes de Referencia

Es posible que la imagen utilizada como referencia no sea la misma que la que se encuentra en la computadora de destino. En tal caso, será necesario capturar nuevamente las imágenes de referencia, incluyendo focos , como ejemplo, el comando “hacer clic en imagen” , etc.

Compatibilidad de Resolución y Zoom

Si la máquina en la que se trabaja es compatible tanto con la resolución como con el zoom utilizados al momento de capturar las pantallas en la referencia de virtualización, se deben seguir las siguientes consignas para asegurar una migración exitosa de los bots:

- **Registro de Comandos**
  Al grabar un comando en el entorno de virtualización, las imágenes se almacenarán en una carpeta específica dentro del directorio de Rocketbot, bajo el nombre correspondiente al bot, en la ruta Rocketbot/robots. Esta carpeta es crucial, ya que las imágenes se utilizan como comandos en Rocketbot Studio, tales como “hacer clic en imagen” y “esperar por imagen”.

- **Migración del Archivo .db**
  Para completar la migración del bot, es imprescindible transferir también el archivo .db. Este archivo es un componente básico del proceso de migración.

Consideraciones Finales

Asegurarse de seguir meticulosamente los pasos anteriores es esencial para lograr una migración exitosa de los bots de virtualización.

Una correcta transferencia de los componentes no solo garantizará que el bot funcione adecuadamente en su nuevo entorno, sino que también optimizará su rendimiento y facilitará futuras actualizaciones.

Adoptar un enfoque riguroso en este proceso es clave para minimizar cualquier inconveniente y asegurar un despliegue eficiente.

En caso de no entender algún concepto relacionado con la virtualización en Rocketbot Studio, se recomienda consultar la documentación : Documentación de Virtualización en Rocketbot.

Studio Rocketbot – Variables internas

Rocketbot Studio cuenta con variables nativas que puedes utilizar para acceder a información rápidamente. Estas variables pueden ser utilizadas desde cualquier comando y no necesitas crearla en la sección de variables o generar una acción para generarlas.

Variable	Descripción
%rocketbot_last_status%	dict. Almacena el estado del último comando ejecutado. {‘status’: ‘True’, ‘message’: ‘setvar ‘, ‘img’: ”, ‘vars’: [], ‘ifs’: [], ‘extra’: [], ‘time’: ‘0.0040669441223144 53’}

Variable	Descripción
%log_path%	str. Almacena la ruta del archivo log de la ejecución en curso
%base_path%	str. Almacena la ruta de la carpeta de rocketbot
%date%	str. Almacena la fecha y hora actual. El formato es YYYY-MM-DD HH:MM:SS.ms
%day%	str. Retorna el día al momento de usar la variable
%month%	str. Retorna el mes al momento de usar la variable
%year%	int. Retorna el año al momento de usar la variable
%hour%	int. Retorna la hora al momento de usar la variable
%minute%	int. Retorna el minuto al momento de usar la variable
%second%	int. Retorna el segundo al momento de usar la variable
%milisecond%	str. Almacena la parte de los milisegundos de la hora al momento de usar la variable

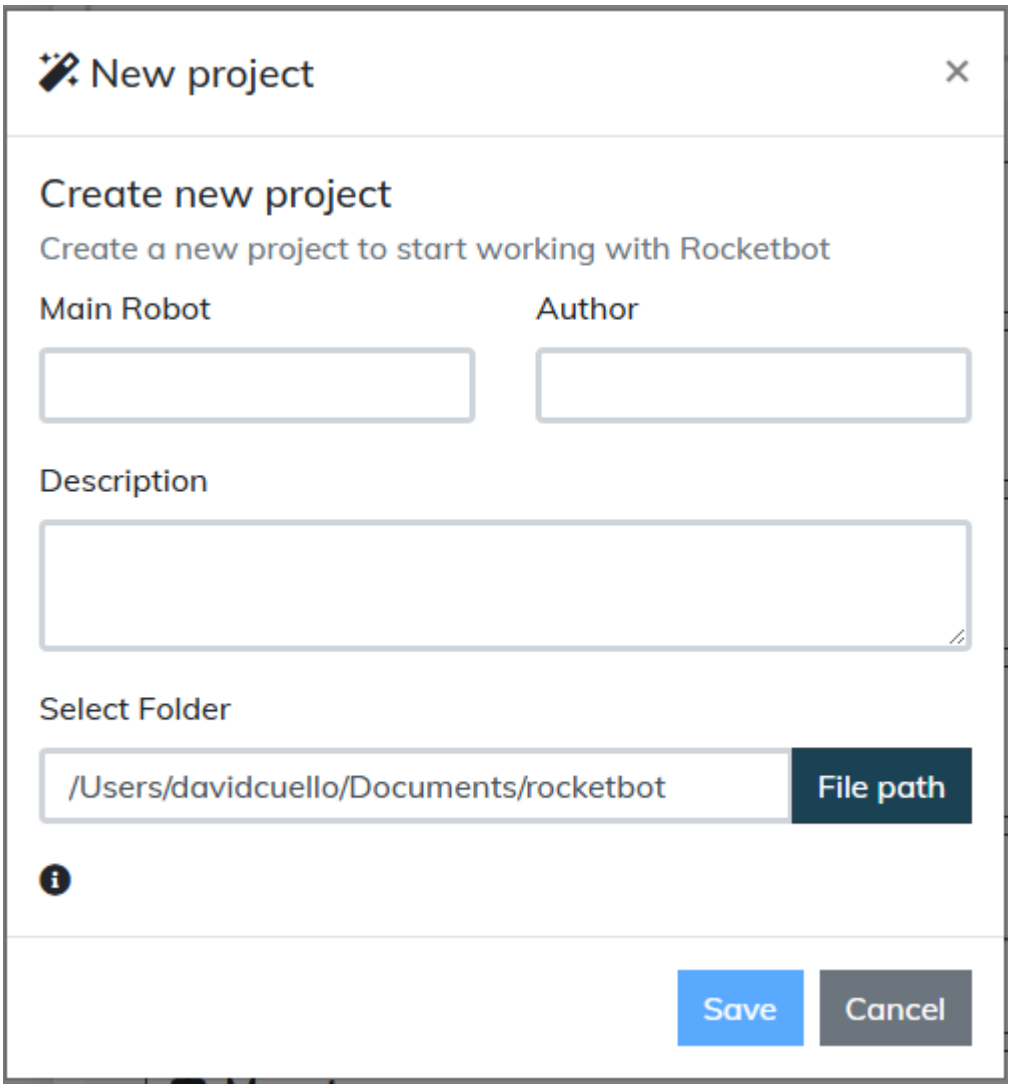
Variable	Descripción
%machine%	str. Almacena el nombre de la máquina donde se está ejecutando el proceso
%tab%	str. Contiene un tab
%newline%	str. Contiene un salto de línea o nueva linea
%enter%	str. Contiene un salto de línea o nueva linea
%osname%	str. Almacena el nombre del sistema operativo
%username%	str. Almacena el nombre de usuario de la computadora
%db_path%	str. Almacena la ruta de la base de datos
%project_path%	str. Almacena la ruta de la carpeta de la base de datos. Para versiones superior a la 2024, corresponde a la carpeta del proyecto
%robot_name%	str. Contiene el nombre del robot
%production%	bool. Retorna True si la ejecución es en producción. En caso contrario, retorna False

Studio Rocketbot – Iniciar un proyecto

Este documento explica como crear un proyecto en Studio que pueda ser ejecutado en cualquier ambiente que cuente con una licencia de Rocketbot Studio.

Nuevo proyecto

Para crear un nuevo proyecto, desde Studio ve a Inicio y haz click en el botón Nuevo Proyecto y verás el siguiente modal.

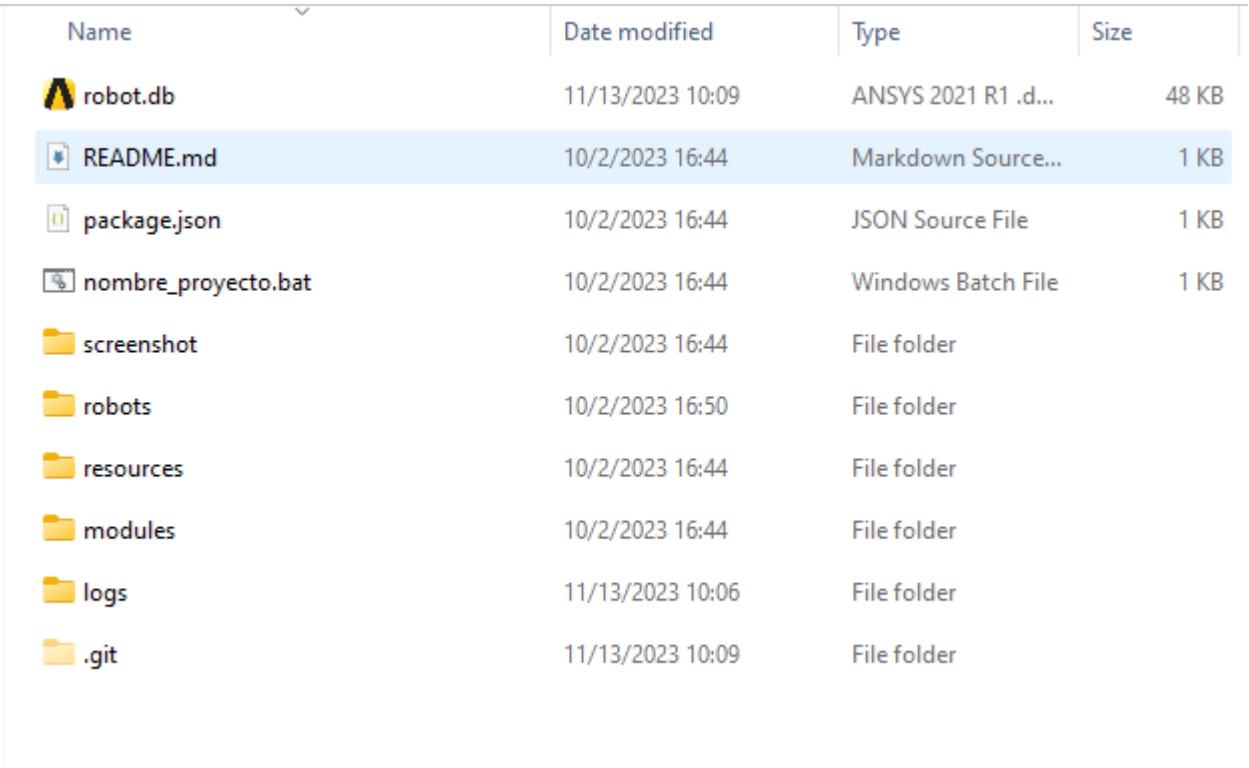


En el modal de nuevo proyecto, completa los siguientes campos:

- Robot principal: Especifica un nombre para tu robot principal y tu proyecto. Este nombre debe ser único para cada proyecto, ya que se utilizará para crear una carpeta con este nombre
- Descripción: Especifica una descripción que resuma lo que quieres hacer con este proyecto.
- Author: Indica el nombre del autor del proyecto
- Seleccionar carpeta: Selecciona la ubicación en la que quieres crear el proyecto. La hubicación predeterminada donde se crean los proyectos es en %USERPROFILE%\Documents\rocketbot

Estructura de carpetas y archivos

Al completar los datos y crear el proyecto, se crearán los siguientes archivos y carpetas:



- robot.db: Base de datos SQLite creada por defecto donde

se almacenará tu robot. En esta base de datos contendrá todos los subrobots que sean creados y podrás llamarlos desde el comando Ejecuta otro script Rocketbot

- README.md: Archivo markdown con el nombre del proyecto y la descripción. Puedes editarlo para agregar una documentación acerca de como ejecutar tu robot

- package.json: Contiene información acerca de tu robot. Este archivo contiene datos como: nombre, descripción, versión, autor, licencia, nombre del robot principal.

 Puedes incluir además las versiones de los módulos que utilice tu robot.

```
{
  "name": "Rocket",
  "description": "",
  "version": "1.0.0",
  "main": "run_main",
  "author": "",
  "license": "MIT",
  "modules": [
    {"name": "AdvancedExcel", "version": "13.2.1"}
  ]
}
```

- Archivo batch: Este archivo contiene la instrucción de consola para ejecutar tu robot. Está asociado al ambiente y rutas donde se creó el proyecto.

```
cd C:\rocketbot
rocketbot.exe -start=Rocket -db="%USERPROFILE%\Documents\rocketbot\Rocket\robot.db"
```

- screenshot: Carpeta donde podrás almacenar las capturas de pantallas que realiza tu robot

- robots: Carpeta con las base de datos de robots como servicio que utiliza tu robot. En esta carpeta puedes agregar robots de otros proyectos para ser utilizados en tu robot

- Resources: Carpeta donde deberás alojar los archivos que tu robot necesita para trabajar y los archivos que genera.

 Puedes agregar subcarpetas si lo necesitas, como carpeta para descargas, plantillas, archivos de configuración, etc

- modules: Carpeta con los módulos que utilizará tu robot. Esta carpeta te permite tener diferentes versiones de los módulos para diferentes proyectos.

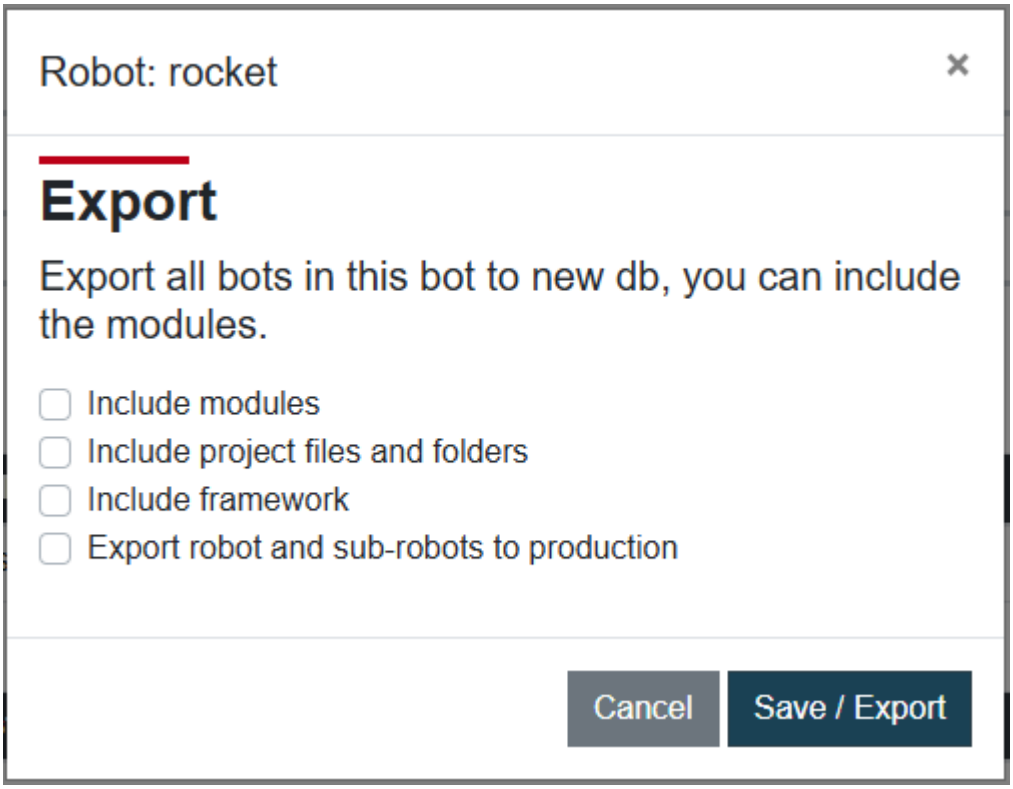
- logs: Carpeta donde se almacenarán los logs generados por la ejecución de cada comando de tu robot. Para que los robots se almacenen en esta carpeta, debes indicarlo en la configuración de tu robot, en caso contrario, los almacenará en la carpeta de rocketbot.

- .git: Carpeta generada por git al generar un nuevo repositorio. Esto te permitirá usar desde studio los comandos de git para el control de versiones

Exportar un proyecto

Para exportar un proyecto debes ir a la pestaña Robot del ambiente de desarrollo y dar click en el botón ***Exportar robots***

a Db. Dependiendo de lo que necesites, deberás marcar alguno de las casillas:



- Sin marcar casillas: Si no se marca ninguna casilla, se creará solo una base de datos de tu robot con la última versión de cada uno de los subrobots.
- Incluir módulos: Al marcar esta casilla, se creará una carpeta comprimida con todos los módulos que utiliza tu robot
- Incluir archivos y carpetas del projecto: Al marcar esta casilla, se creará una copia de la carpeta de tu proyecto en la ruta indicada luego de dar click en **Guardar/Exportar**
- Incluir Framework: Al marcar esta casilla, se exportará ademas una versión reducida de rocketbot que solo permite ejecutar los robots. Con esta versión reducida, puedes ejecutar tu robot sin permitir entrar a Studio
- Exportar robots a producción: Al marcar esta casilla, la base de datos creada se encriptará y no se podrá ver el contenido del robot desde Studio.

⚠ Si marcas la casilla Exportar robots a producción, no selecciones una base de datos existente. Al encriptar los datos, no podrás volver a ver ni editar el contenido y podría perder tu robot

Diferencia con versiones anteriores

Esta documentación está basada en las funcionalidades de Rocketbot Studio 2024.05.28 o superior. Versiones tienen diferencias en la creación de proyectos y se deben configurar manualmente para replicar a las ultimas versiones

Versión 2023

La estructura de carpetas al crear un proyecto fue integrada en la versión Rocketbot Studio 2023.03.30 pero con ligeras diferencias

Carpetas y archivos

package.json	Este archivo no es generado en la versión 2023
modules	Esta carpeta se utiliza para almacenar las versiones de los módulo, pero el robot no los utiliza en la ejecución
logs	Esta carpeta era utilizada para almacenar los logs generados intencionalmente con el robot. No se almacenaban los logs generados por la actividad del robot
robots	Esta carpeta era utilizada para almacenar copias de seguridad de la base de datos original. No se podían utilizar los robots como servicios

Menú de exportar

La versión 2023 no cuenta con la opción de exportar incluyendo archivos y carpetas del proyecto ni incluir el framework

Versión 2020 e inferior

Las versiones inferior a la 2023 no cuentan con la funcionalidad de crear proyecto. Para estas versiones, se debe crear una base de datos y luego generar la estructura de carpetas de forma manual.

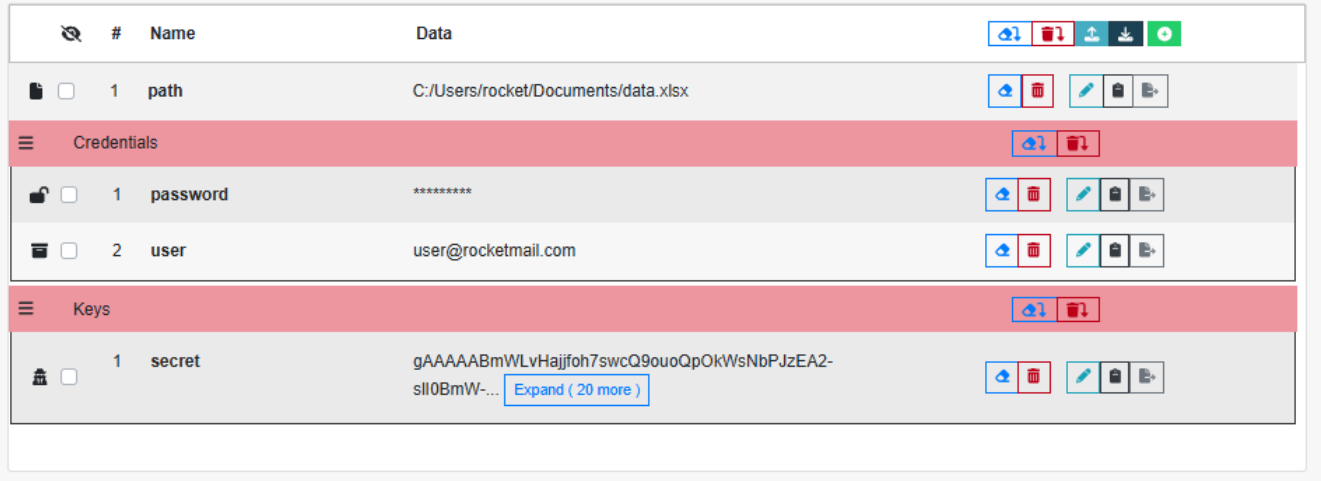
Para exportar una base de datos, puedes elegir entre dos botones en la pestaña robot. Exportar robots y subrobots, y exportar robots y subrobots a producción

---

Studio Rocketbot – Pestaña variables

La pestaña variables en Studio te permite crear, modificar y organizar las variables del robot.

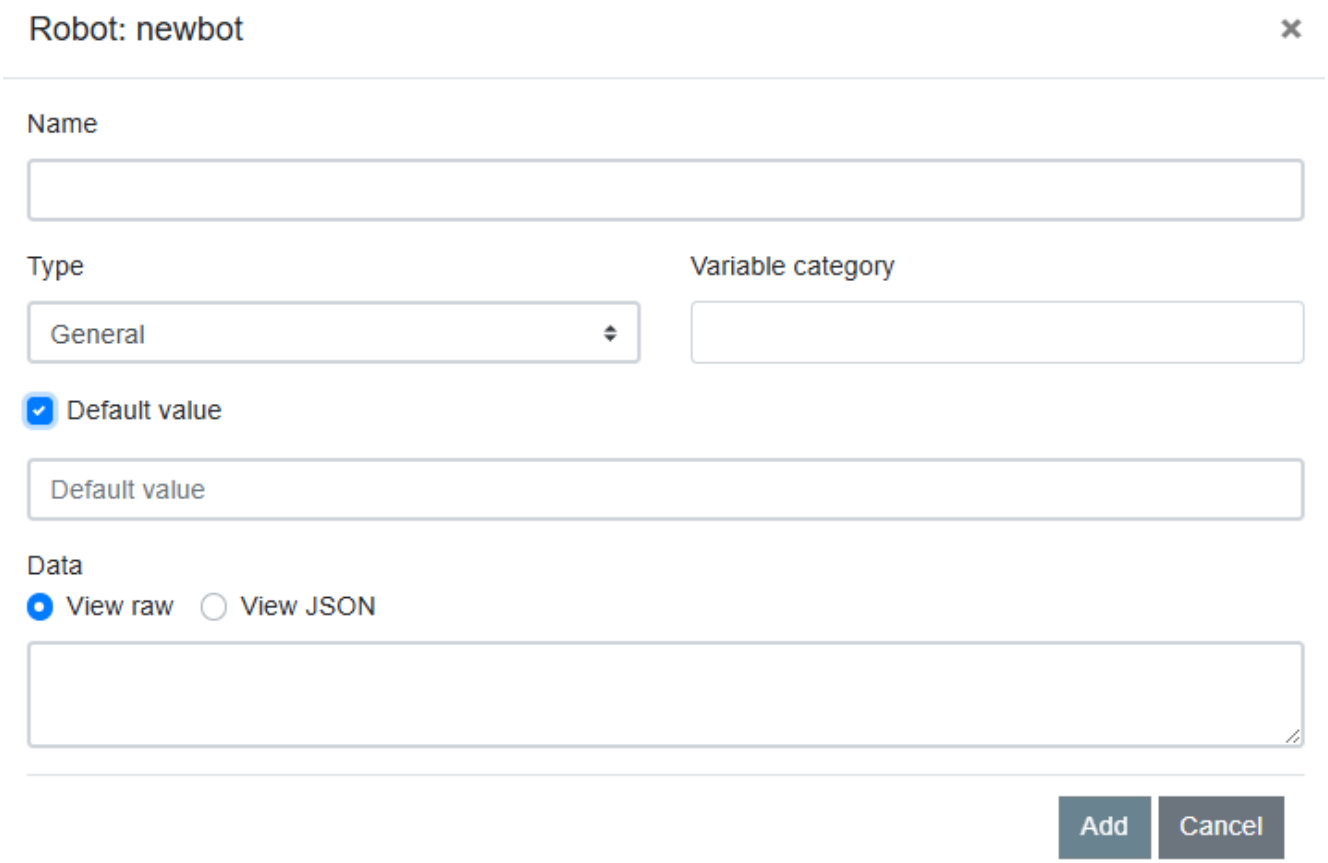
Acciones



Secciones	Descripción
Desactivar ()	Indica si la variable está activa o no. Si la casilla está marcada, el robot asumirá que la variable no debe ser usada
Nombre	Nombre de la variable
Datos	Contenido de la variable en el momento que se observa
Limpiar ()	Vacía o vuelve al valor por defecto de la variable
Eliminar ()	Elimina la variable de la lista de variables pero no de los comandos que la utilicen
Editar ()	Muestra el modal para editar la variable
Copiar ()	Copia al portapapeles el contenido de la variable o las variables si hay más de una seleccionada
Exponer ()	Convierte la variable a una variable expuesta o variable de entrada. Para más información, ir Expose

Crear variables

Para crear una nueva variable para tu robot, debes dar click en el botón (+) verde y completar los datos de la ventana emergente



Campo	Descripción
Nombre	Nombre único de la variable. Obligatorio. El nombre no puede repetirse en el mismo robot y solo puede contener caracteres alphanuméricos o guiones
Tipo	Permite elegir como manejar la visualización de la variable. Opcional. Las opciones disponibles son: – General – Password: Modificará el input a tipo password para ocultar los datos – File: Modificará el input a file para poder seleccionar archivos fácilmente – Folder: Modificará el input a tipo folder para poder seleccionar carpetas – Encrypted: Encriptará los datos para que no vuelvan a ser visibles

Campo	Descripción
Categoria	Categoría en donde se agruparán las variables. Opcional. Si el nombre de la categoría no existe, se creará un nuevo grupo
Valor Default	Valor por defecto que se asignará cuando se limpien los datos. Opcional. Si la casilla no está marcada, el valor por defecto será vacío Cuando se de click en alguno de los botones para limpiar los datos, en lugar de vaciar la variable, se asignará el valor plano por defecto, si la variable es de tipo encrypted, no la encriptará
Dato	El valor que contiene la variable en el momento de crearse. Opcional. A diferencia del campo default, acá se ve el valor de la variable en un estado puntual y si algún comando la modifica, no puede volver al estado incial.