

Orquestador Rocketbot: APIs Xperience

¿Qué es una API?

Una API (Interfaz de Programación de Aplicaciones) es un conjunto de reglas, protocolos y herramientas que permiten a diferentes sistemas de software comunicarse e interactuar entre sí, de manera estructurada. Las direcciones URL a través de las cuales se envían solicitudes y se recibe información de las APIs se denominan **endpoints**.

Requisitos Previos

- Acceso al Orquestador
- El usuario del Orquestador debe tener una [api Key generada](#).

Endpoints disponibles

Listar Proyectos

Devuelve un listado de los proyectos cargados en el Orquestador, con sus respectivos procesos y robots asignados.

Método URL

Parámetros

POST <https://www.myrb.io/dev/api/project/list>

Authorization: tipo Bearer Token; es el Token de acceso válido para autenticar al usuario (API Key asociada al perfil en el orquestador)

Avanzado

Ejemplos de requests

- **cURL**

```
curl --location --request POST 'https://www.myrb.io/dev/api/project/list' \
--header 'Authorization: Bearer <token>'
```

- **Python**

```
import requests
```

```
url = "https://www.myrb.io/dev/api/project/list"
```

```
payload = {}
headers = {
  'Authorization': 'Bearer <token>'
}
```

```
response = requests.request("POST", url, headers=headers, data=payload)
```

```
print(response.text)
```

Ejemplo de response

```
{
  "success": true,
  "data": [
    {
      "name": "test orquestador",
      "token": "JR0EYGNSEXEMTQWUP",
      "created_at": "2024-08-15 08:42:16",
      "id": 69,
      "description": null,
      "process_count": 1,
      "process": [
        {
          "id": 168,
          "project_id": 69,
          "token": "1KZQIJKTTR75CGWF",
          "name": "bot orquestador",
          "last_robot": {
            "id": 143,
            "name": "bot orquestador",
            "user_id": 67,
            "client_id": 3,
            "cron": null,
            "version": null,
            "version_rocketbot": null,
            "robot":
"myrb_clients/dev/project/69/process/168/robot/rocket.bot",
            "md5": "7af04cc5dbbea2e6ff1f27c183ef4772",
            "comment": null,
            "type": 1,
            "status": 0,
            "process_id": 168,
            "start_bot": "bot_orquestador",
            "created_at": "2024-08-21 09:37:50",
            "updated_at": "2024-08-21 09:37:50"
          },
          "backup": null,
          "trigger_process": null,
          "client": null,
          "robots": [
            {
```


Authorization:
tipo Bearer Token;
es el Token de
acceso válido para
autenticar al
usuario (API Key
asociada al perfil
en el
orquestador).
name: Nombre del
proceso al que se
asignará el robot.
startbot: Nombre
del robot como
aparece en Studio.
process_id: Token
del proceso. Hace
referencia al
token del proceso,
se puede obtener
desde el ROC ó
utilizando el
endpoint
/api/project/list.
project_id:
Identificador del
proyecto al que
pertenece el
proceso. Debe
obtenerse
previamente
utilizando el
endpoint
/api/project/list.
file: robot.bd

POST <https://www.myrb.io/dev/api/process/robot/update/data> multipart/form-data

Avanzado

Ejemplos de requests

- cURL

```
curl --location 'https://www.myrb.io/dev/api/process/robot/update/data' \  
--header 'Authorization: Bearer <token>' \  
--form 'name="process1"' \  
--form 'startbot="navegador"' \  
--form 'process_id="NDQGBJBSE4LDRSVA"' \  
--form 'project_id="56"' \  
--form 'file=@"/C:/Users/navegador.db"'
```

- Python

```
import requests
```

```
url = "https://www.myrb.io/dev/api/process/robot/update/data"
```

```
payload = {'name': 'process1',  
'startbot': 'navegador',  
'process_id': 'NDQGBJBSE4LDRSVA',
```

```
'project_id': '56'}
files=[
('file', ('navegador.db', open('/C:/Users/navegador.db', 'rb'), 'application/octe
t-stream'))
]
headers = {
  'Authorization': 'Bearer <token>'
}
```

```
response = requests.request("POST", url, headers=headers, data=payload,
files=files)
```

```
print(response.text)
```

Ejemplo de response

```
{
  "success": true,
  "data": {
    "robot": "myrb_clients/dev/project/56/process/152/robot/rocket.bot",
    "md5": "484871810fe6e7ed78b4bacb53a408b3",
    "start_bot": "navegador",
    "status": 0,
    "type": 1,
    "name": "process1",
    "user_id": 66,
    "process_id": 152,
    "client_id": 3,
    "updated_at": "2024-09-18 16:58:14",
    "created_at": "2024-09-18 16:58:14",
    "id": 145,
    "process": {
      "id": 152,
      "name": "process1",
      "token": "NDQGBJBSE4LDRSVA",
      "robot_id": 131,
      "config": null,
      "client_id": 3,
      "data": null,
      "status": 0,
      "minutes": null,
      "sent_email": null,
      "email": null,
      "trigger": null,
      "created_at": "2024-08-05 17:14:14",
      "updated_at": "2024-08-05 13:14:14",
      "project_id": 56,
      "form_id": null,
      "private": 0,
      "last_robot": {
        "id": 145,
```

```
    "name": "process1",
    "user_id": 66,
    "client_id": 3,
    "cron": null,
    "version": null,
    "version_rocketbot": null,
    "robot":
"myrb_clients/dev/project/56/process/152/robot/rocket.bot",
    "md5": "484871810fe6e7ed78b4bacb53a408b3",
    "comment": null,
    "type": 1,
    "status": 0,
    "process_id": 152,
    "start_bot": "navegador",
    "created_at": "2024-09-18 16:58:14",
    "updated_at": "2024-09-18 16:58:14"
},
"backup": null,
"trigger_process": null,
"client": {
    "id": 3,
    "name": "Academy",
    "owner": null,
    "parent": null,
    "type": 2,
    "status": 0,
    "created_at": "2022-09-05 11:07:44",
    "updated_at": "2023-01-03 11:04:21"
},
"robots": [
    {
        "id": 131,
        "name": "process1",
        "user_id": 66,
        "client_id": 3,
        "cron": null,
        "version": null,
        "version_rocketbot": null,
        "robot":
"myrb_clients/dev/project/56/process/152/robot/rocket.bot",
        "md5": "5594faf198e491c7868cf600204f39d2",
        "comment": null,
        "type": 1,
        "status": 0,
        "process_id": 152,
        "start_bot": "saludo",
        "created_at": "2024-08-05 17:14:14",
        "updated_at": "2024-08-06 10:38:46"
    },
    {
        "id": 145,
```



```
--data-urlencode 'process_id=CEDLU6FK79BOLPUM'
```

- **Python**

```
import requests

url = "https://www.myrb.io/dev/api/process/robots/start"

payload = 'process_id=CEDLU6FK79BOLPUM'
headers = {
    'Authorization': 'Bearer <token>',
    'Content-Type': 'application/x-www-form-urlencoded'
}

response = requests.request("POST", url, headers=headers, data=payload)

print(response.text)
```

Ejemplo de response

```
{
  "success": true
}
```

Obtener todas las queues

Devuelve un listado de las queues de todos los [formularios de Xperience](#) existentes en el Orquestador, pudiendo además filtrar por fecha de inicio y fin de las mismas.

Método URL	Formato	Parámetros
------------	---------	------------

Authorization:
tipo Bearer
Token; es el
Token de
acceso válido
para
autenticar al
usuario (API
Key asociada
al perfil en
el
orquestador).
from: Fecha de
inicio de la
búsqueda
(YYYY-MM-DD).
to: Fecha de
fin de la
búsqueda
(YYYY-MM-DD)

POST <https://www.myrb.io/dev/api/formData/all> application/json

Avanzado

Ejemplos de request

- cURL

```
curl --location 'https://www.myrb.io/dev/api/formData/all' \  
--header 'Authorization: Bearer <token>' \  
--header 'Content-Type: application/json' \  
--data '{  
  "from": "2024-06-9",  
  "to": "2024-07-16"  
'
```

- Python

```
import requests  
import json  
  
url = "https://www.myrb.io/dev/api/formData/all"  
  
payload = json.dumps({  
  "from": "2024-06-9",  
  "to": "2024-07-16"  
})  
headers = {  
  'Authorization': 'Bearer <token>',  
  'Content-Type': 'application/json'  
}
```

```
response = requests.request("POST", url, headers=headers, data=payload)

print(response.text)
```

Ejemplo de response

```
{
  "success": true,
  "data": [
    {
      "id": 8315,
      "form_name": "five9notas",
      "form_token": "6HNZNSYVG4W2MCS0",
      "created_at": "2024-07-10 11:04:55",
      "processed_at": "2024-07-10 11:06:26",
      "status": 1,
      "user_form_email": null,
      "locked_user": "Bot Integracion",
      "locked": 0,
      "form_id": 25
    },
    {
      "id": 8316,
      "form_name": "Clientify",
      "form_token": "F9J65RTNWWPMTYD8",
      "created_at": "2024-07-11 10:25:35",
      "processed_at": "2024-07-11 10:27:13",
      "status": 1,
      "user_form_email": null,
      "locked_user": "Robot Interno",
      "locked": 0,
      "form_id": 14
    }
  ]
}
```

Añadir datos a queue

Completa un formulario de Xperience y coloca los datos en cola para ser procesados (Debe tener activa la opción "send API" en el formulario).

Método URL

Formato

Parámetros

Authorization:
tipo Bearer
Token; es el
Token de
acceso válido
para
autenticar al
usuario (API
Key asociada
al perfil en
el
orquestador).
Dato1:
Dato2:
(Otros Datos):
Se pueden
incluir campos
adicionales
según el
diseño del
formulario.

https://www.myrb.io/dev/api/formData/addQueue/:token
POST **token:** Representa el token de formulario creado en multipart/form-data el
ROC Xperience.

Avanzado

Ejemplo de request

- **cURL**

```
curl --location 'https://www.myrb.io/dev/api/formData/addQueue/:token' \  
--header 'Authorization: Bearer <token>' \  
--form 'email="prueba@prueba.com"' \  
--form 'nombre="prueba1"'
```

- **Python**

```
import requests  
  
url = "https://www.myrb.io/dev/api/formData/addQueue/:token"  
  
payload = {'email': 'prueba@prueba.com',  
          'nombre': 'prueba1'}  
files=[  
  
]  
headers = {  
    'Authorization': 'Bearer <token>'  
}  
  
response = requests.request("POST", url, headers=headers, data=payload,  
                             files=files)  
  
print(response.text)
```

Ejemplo de response

```
{
  "success": true,
  "data": {
    "xperience":
"eyJpdjI6Inpid0RRVlV0bEczTW9DM1Z5SEFrV1E9PSIsInZhbHVlIjo1OVJCZHhHTDBwQ0p1Y2ps
VU9WZTYxbE5paFwvRTQ4cWlKSfZ3UUxGcVNUQXh4UTBx0FEwc01Ba0pvV1kwQklRdXByTE1JU2tWM
TMxZkx5Qk41SFNTNmdRPT0iLCJtYWMiOiI5YjFiNjAyMTRiODUwMjk3YjdjZGEwZjFmMjE4MjhiNG
I5MzA5MTViNmVhMTE4OTI0NmExNTgwYWE4YWJlYWNjIn0="
  }
}
```

Obtener todos los queues id

Devuelve un listado con los IDs de todas las queues con estado "Not Processed" de un formulario de Xperience existente.

Método URL

Parámetro

Authorization:

tipo Bearer Token; es el Token de acceso válido para autenticar al usuario (API Key asociada al perfil en el orquestador).

POST **https://www.myrb.io/dev/api/formData/get/:token:**
:token: Representa el token de formulario creado en ROC Xperience.

Avanzado

Ejemplos de request

- **cURL**

```
curl --location --request POST
'https://www.myrb.io/dev/api/formData/get/:token' \
--header 'Authorization: Bearer <token>'
```

- **Python**

```
import requests

url = "https://www.myrb.io/dev/api/formData/get/:token"

payload = {}
headers = {
  'Authorization': 'Bearer <token>'
}
```

```
response = requests.request("POST", url, headers=headers, data=payload)

print(response.text)
```

Ejemplo de response

```
{
  "success": true,
  "data": [
    {
      "id": 4734,
      "form_name": null,
      "user_form_email": null,
      "locked_user": null,
      "form_token": null,
      "form": null,
      "user_form": null,
      "locker": null
    },
    {
      "id": 4735,
      "form_name": null,
      "user_form_email": null,
      "locked_user": null,
      "form_token": null,
      "form": null,
      "user_form": null,
      "locker": null
    }
  ]
}
```

Cambiar de estado una queue

Permite modificar el estado de un queue.

Método URL

Formato

Parámetro

Authorization:
tipo Bearer
Token; es el
Token de
acceso válido
para
autenticar al
usuario (API
Key asociada
al perfil en
el
orquestador).
status: Un
número entero
que indica el
nuevo estado
de la cola. 0:
No procesado
1: Procesado
lock: Un nú 0:
No bloqueada
1: Bloqueada

POST **https://www.myrb.io/dev/api/formData/setStatus/:id:**
:id: Representa el ID de la cola que se desea
modificar. Este ID debe obtenerse previamente
utilizando el endpoint **/formData/get/:token.** multipart/form-data

Avanzado

Ejemplos de request

- **cURL**

```
curl --location 'https://www.myrb.io/dev/api/formData/setStatus/:id' \  
--header 'Authorization: Bearer <token>' \  
--form 'status="1"' \  
--form 'locked="0"'
```

- **Python**

```
import requests  
  
url = "https://www.myrb.io/dev/api/formData/setStatus/:id"  
  
payload = {'status': '1'}  
files=[  
  
]  
headers = {  
    'Authorization': 'Bearer <token>'  
}  
  
response = requests.request("POST", url, headers=headers, data=payload,  
files=files)  
  
print(response.text)
```

Ejemplo de response

```
{
  "success": true
}
```

Obtener datos de un formulario

Permite recuperar los datos de un formulario específico que se encuentra dentro de una queue determinada. Es útil para procesar los datos de un formulario una vez que ha sido añadido a la queue.

Método URL

Parámetros

Authorization:
tipo Bearer
Token; es el
Token de
acceso válido
para
autenticar al
usuario (API
Key asociada
al perfil en
el
orquestador).

https://www.myrb.io/dev/api/formData/getQueue/:id/:token
:id: Representa el ID de la cola. Este ID debe obtenerse
PREVIAMENTE utilizando el endpoint **/formData/get/:token**.
:token: Representa el token de formulario creado en ROC
Xperience.

Avanzado

Ejemplos de request

- **cURL**

```
curl --location --request POST
'https://www.myrb.io/dev/api/formData/getQueue/:id/:token' \
--header 'Authorization: Bearer <token>'
```

- **Python**

```
import requests

url = "https://www.myrb.io/dev/api/formData/getQueue/:id/:token"

payload = {}
headers = {
  'Authorization': 'Bearer <token>'
}

response = requests.request("POST", url, headers=headers, data=payload)

print(response.text)
```



```
--data-urlencode 'instance=66b1418f93213'
```

- **Python**

```
import requests

url = "https://www.myrb.io/dev/api/assets/get"

payload = 'name=toNotify&instance=66b1418f93213'
headers = {
    'Authorization': 'Bearer <token>',
    'Content-Type': 'application/x-www-form-urlencoded'
}

response = requests.request("POST", url, headers=headers, data=payload)

print(response.text)
```


Ejemplo de response

```
{
  "success": true,
  "data": {
    "id": 31,
    "name": "toNotify",
    "type": "password",
    "value": "prueba@prueba.com",
    "client_id": 3,
    "process_id": 0,
    "instance_id": 0,
    "created_at": "2023-03-21 17:12:35",
    "updated_at": "2023-10-25 12:21:49"
  }
}
```

[Orquestador Rocketbot – Xperience form editor](#)

Aquí se crean y editan los formularios de Xperience.

Form settings (propiedades del formulario)



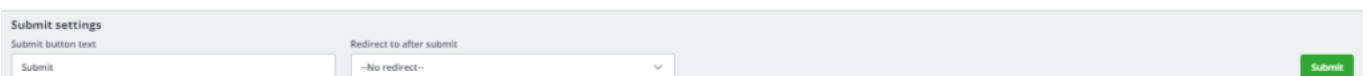
- Form name** Permite editar el nombre del formulario. Debe contener más de dos caracteres, solo caracteres alfanuméricos y espacios, y no debe estar compuesto solo por espacios. No pueden existir dos formularios con el mismo nombre.
- Public** Si está marcado como público, cualquiera que tenga el enlace del formulario podrá completarlo. Si está marcado como privado, solo los usuarios registrados con el formulario asignado podrán completarlo.
- Enabled** Permite editar el nombre del formulario. Debe contener más de dos caracteres, solo caracteres alfanuméricos y espacios, y no debe estar compuesto solo por espacios. No pueden existir dos formularios con el mismo nombre.
- reCaptcha** Cuando está activado, al intentar enviar un formulario, el usuario deberá completar un captcha.
- send API** Si está activado, permite completar el formulario mediante una API, sin necesidad de una interfaz gráfica.
- Form token** Indica el token asignado al formulario, el cual no puede ser editado. Al hacer clic en el ícono de copiado, este token se copiará en el portapapeles.

Botones de Visualización / Guardado:



- Visualización** Redirige al usuario a la vista de completado del formulario, mostrando lo que verá el usuario final al completar el formulario.
- Guardado** Permite guardar el formulario. Si ocurre un error durante el guardado, se mostrará un mensaje de error y el formulario no podrá ser guardado.

Submit settings (propiedades del completado del formulario):



- Submit button text** Aquí se indica el texto que tiene que tener el botón de completado de formulario. Por defecto es Submit.
- Redirect to after submit** Permite al usuario elegir qué ocurrirá después de completar el formulario. Por defecto, no hay redirección. El usuario puede seleccionar la opción URL para redirigir a otra dirección.
- Target** Esta opción aparece si el usuario eligió URL en "Redirigir luego de completar". Aquí se ingresa la URL a la que el usuario será redirigido después de completar el formulario.

Botón de agregar elemento, guardar



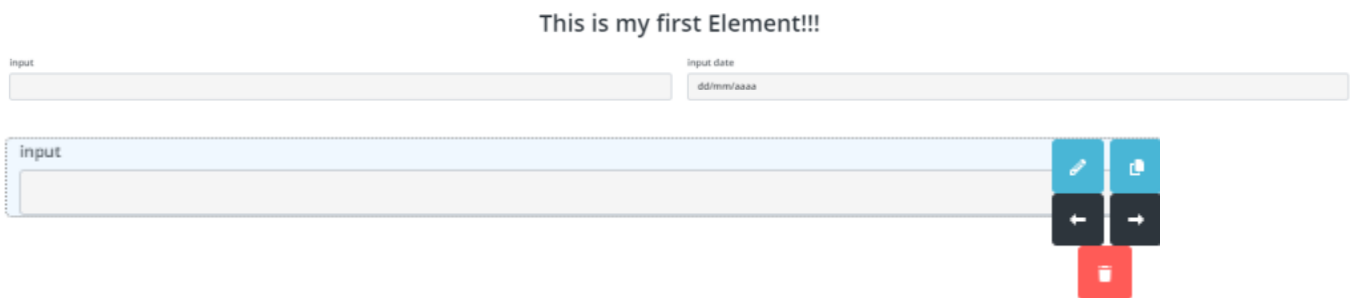
- Add element** Abre el modal de creación de elemento (ver sección de crear/editar elemento).
- Save** Guarda el formulario. Si surge algún error durante el guardado, se mostrará un mensaje de error y el formulario no podrá ser guardado.

Pestaña de Editor



Visor de elementos:

Aquí se pueden ver y modificar todos los elementos del formulario.



Pasando el mouse por sobre cada elemento aparecerá su menú de opciones.

Desde este visor las distintas acciones que se pueden realizar sobre los elementos son:

Mover elemento: Arrastrándolo con el mouse o utilizando las flechas del menú de opciones. “->” mueve el elemento una vez hacia la derecha, “<-” mueve el elemento una vez hacia la izquierda.



Editar elemento: Al hacer clic en el ícono del lápiz en el menú de opciones. Abre el modal de edición del elemento (ver sección de crear/editar elemento).



Clonar elemento: Al hacer clic en el ícono de copiar en el menú de opciones, se clonará el elemento (ver sección de clonar elemento).



Borrar elemento: Al hacer clic en el botón rojo con el ícono de cesto de basura en el menú de opciones, el elemento será eliminado.

Crear / Editar formulario

Los formularios se crearán/editarán mediante un modal.

New element

Element type (*)	Element id (*)	Bootstrap col class (*)
--Select form element type--		--Select col class--
Element title		
Element style		
Element required	Css class	
No		

(*) Required fields

Add Close

Al editar un elemento, se mostrará un modal similar, pero de color azul y con los campos del elemento completados con sus valores correspondientes. Tiene exactamente los mismos campos que un modal de crear elemento.

Edit element

Element type (*)	Element id (*)	Bootstrap col class (*)
Title	12	col-md-1
Element title		
1		
Element style		
Element required	Css class	
No		

(*) Required fields

Save Close

Al clonar un elemento, se abrirá un modal similar al de creación, con todos los campos llenos con los valores del objeto original, excepto el ID, que estará vacío. No se puede repetir el ID del elemento original. Al clonarlo, el clon aparecerá al lado (o abajo, si no cabe en la fila) del elemento original.

Si el elemento original tenía custom functions asignadas a eventos, su clon

las tendrá asignadas de forma similar.

Atributos obligatorios:

Están marcados con un (*); el elemento no puede ser creado/editado si no están completados.

Tipo	(ver tipos de elemento disponibles)
Id	No puede repetirse entre elementos. Es obligatorio ingresar manualmente un id en todos los elementos salvo en Title, Paragraph, Image y Space. En estos casos si se crea/edita el elemento sin un id, se le asignará uno creado aleatoriamente.
Bootstrap col class	Indica el ancho que tendrá el elemento en el formulario, variando de 1 a 12 unidades. Una fila en un formulario puede contener hasta 12 unidades.

Atributos comunes a todos los elementos, no obligatorios

css class	La clase css que tendrá el elemento en el formulario.
style	El estilo inline que tendrá el elemento en el formulario.
required	Si el elemento es requerido para poder completar el formulario.

Atributos específicos a ciertos elementos

Element title	La etiqueta del elemento. Disponible en todos menos en Paragraph.
Content	Disponible en Paragraph. Contenido del párrafo.
Pattern data	Disponible en Entrada y Contraseña. Indica el patrón que debe tener el elemento. Si no se cumple, el formulario no puede completarse.
File size y file type	Disponible en File. File size indica el tamaño máximo en bytes que puede tener el archivo subido, y file type es un selector con el tipo de archivo requerido. Si se indica un tipo no se podrán subir otro tipo de archivos que no sea el tipo indicado.
Max y Min value	Disponible en Range. Indica el valor mínimo y máximo.
Options (Image)	Disponible en Image, Aquí se ingresa la url o el código base64 de la imagen que se desea mostrar.
Options (Select)	Disponible en Select. Hay dos opciones, agregar las opciones manualmente o traerlas desde una API. Para agregarlas manualmente, ingrese un Text (Texto) y un Value (Valor) en los campos correspondientes, luego presione el botón verde "+" para añadir el par Texto-Valor. Ambos campos deben estar completos para que la adición sea exitosa. En la tabla se mostrarán todas las opciones ingresadas. Cada opción tiene botones para mover hacia arriba, mover hacia abajo y borrar. Los botones de flecha permiten cambiar la posición de las opciones, mientras que el botón rojo elimina la opción seleccionada. También puede traer valores a través de una API. Ingrese la URL donde se hará la petición, el nombre de la llave que trae el texto de cada opción en "Text Key" y el nombre de la llave que trae el valor de cada opción en "Value Key".

Events (eventos)

En los elementos tipo Input, Date, Number, Password, Textarea, Checkbox, File, Range el footer del modal tendrá un botón Show Events. Clickearlo expandirá un grupo de selectores, volverlo a clicar los esconderá.

Estos eventos ocurren cuando se realiza una determinada acción en cada elemento HTML del form. Estos son: *onclick*, *onchange*, *onblur*, *onmouseover*, *oninput*. Para más información sobre estos eventos ver https://www.w3schools.com/tags/ref_eventattributes.asp.

Event	Custom function
On click	<input type="text" value="-- No function --"/>
On change	<input type="text" value="-- No function --"/>
On blur	<input type="text" value="-- No function --"/>
On mouse over	<input type="text" value="-- No function --"/>
On input	<input type="text" value="-- No function --"/>

Cada evento tendrá seleccionado un selector, en el se podrá elegir una custom function (función customizada) creada por el usuario, para asignarla a ese evento. No es obligatorio asignarle funciones a cada evento.

Para ver cómo funcionan los eventos, dirigirse a *Events y custom functions*.

Tipos de elemento disponibles

Title (título)	Heading de página.
Paragraph (párrafo)	Un bloque de texto.
Input	Un input para ingresar texto.
Date (fecha)	Un input para ingresar una fecha.
Number (número)	Un input para ingresar valores numéricos.
Password	Un input para ingresar una contraseña.
Textarea (Área de texto)	Un input para ingresar área de texto.
Checkbox	Un checkbox individual.
File (archivo)	Un input para seleccionar un archivo en el dispositivo desde el que se ve el formulario.
Range (rango)	Una barra en la que se seleccionan valores ubicados entre un valor mínimo y un valor máximo.
Select	Un selector de valores ingresados manualmente o traídos desde una api.
Space (espacio)	Un espacio vacío. En el editor se ve como una caja de líneas punteadas, pero en el visor del formulario se verá como un espacio vacío.
Image (imagen)	Una imagen, se puede traer mediante una url o mediante un código de base64.
Signature (Firma)	Un canvas donde el usuario puede dibujar su firma. En el editor se visualizará como un rectángulo de bordes sólidos.
QR Reader (Lector de QR)	Un lector de QR. En el editor se verá como una imagen estática.
Barcode EAN Reader (Lector de código de barras)	Un lector de código de barras. En el editor se verá como una imagen estática.

Takephoto (Foto)

Un capturador de foto. En el editor se verá como una imagen estática.

Pestaña de Javascript



Aquí se puede customizar la experiencia del usuario del formulario agregándole código Javascript.


Librerías CDN

Se puede agregar código Javascript de terceros mediante [librerías CDN](#).

En *URL to CDN* (Dirección a CDN) se puede ingresar la dirección a la librería deseada. Con el botón Add (Agregar) se agrega a la lista de CDNs agregadas.

No se puede agregar la misma dirección dos veces. Si se lo intenta aparecerá un mensaje de error y no se permitirá agregar la librería.

Add Javascript libraries to give more fascinating options to your Rocketbot Xperience Form View

URL to CDN		
<input type="text" value="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.30.1/moment.min.js"/>	Add	
Library already added		
#	Javascript Lib CDN	
1	https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.30.1/moment.min.js	

En la tabla se listarán todas las librerías asignadas, con su dirección y su número de orden en que fueron ingresadas, de primera a última ingresada. El botón rojo con el cesto de basura eliminará la librería seleccionada de la lista.

Código manual

Aquí se puede ingresar manualmente código personalizado. A la izquierda se encuentra un menú desplegable con Global, Events y Custom functions. A la derecha se encuentra un editor de código asociado a la sección elegida en el menú. El área del menú que este en ese momento seleccionada estará resaltada en color azul en el menú.

Add your own Javascript code to give more functionality to your Rocketbot Xperience

- ▼ Global
- Code
- ▶ Events
- ▶ Custom functions

```
JS  
1 code goes here...
```

Global

Es código Javascript que afecta a todo el formulario. Cuando se ingresa a la pestaña de Javascript está expandido y seleccionado por defecto. En caso que se quiera volver a acceder, en el menú clicar *Global* para expandirlo y luego clicar en *Code*.

Events (Eventos)

Bajo este menú se listarán, primero los eventos particulares del formulario, y luego los eventos asociados a los elementos del formulario.

Form events (Eventos del formulario):

Son los eventos ligados a todo formulario.

- load** Cuando un formulario se termina de cargar en el navegador.
- submit** Cuando un formulario es completado luego de clicar el botón de enviado.
- data_received** Si el formulario está asociado a un robot, se ejecuta cuando llegan datos desde este a través del comando send data to Xperience.

Clickeando en cada uno de ellos se puede acceder a su editor de código.

Events (eventos)





- ▼ Global
- Code
- ▼ Events
 - ▶ formEvents
 - ▶ select_element_ID (select)
 - ▶ file_element_ID (file)
 - ▶ check_ID (checkbox)
- ▶ Custom functions

En este menú se listan todos los elementos creados del form cuyo tipo tenga eventos asociados (Input, Date, Number, Password, Textarea, Checkbox, File, Range). En el menú se listarán por su ID, con el tipo de elemento entre paréntesis.

▼ select_element_ID (select)
click
change
blur
mouseover
input

Clickear cada uno de ellos los expandirá y se podrán ver listados todos los eventos asociados. Clickear cada evento accederá a su editor de código.

Custom functions (Funciones customizadas)

▼ Custom functions
+
function_1  
function_2  

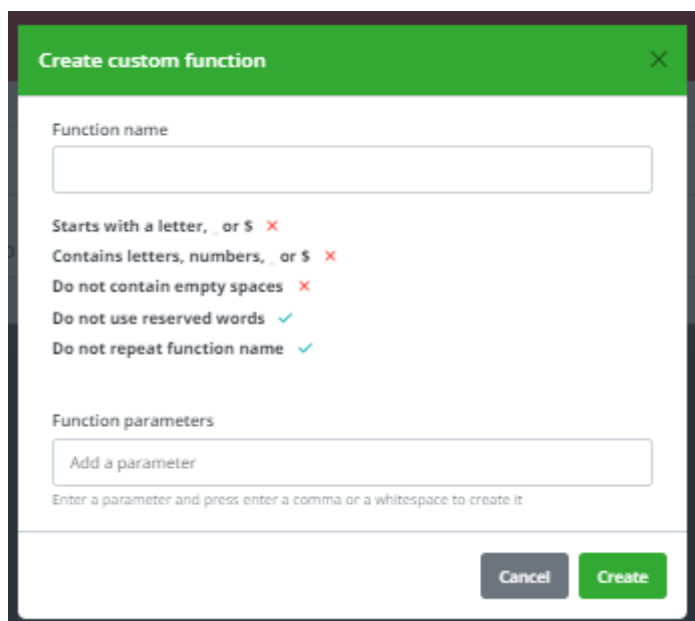
Aquí se listarán por nombre todas las funciones creadas. Clickeando el botón de “+” al principio de la lista abrirá el modal de creación de función.

Cada función posee un nombre, ningún o varios parámetros y su código. Las funciones no pueden tener nombres repetidos y éstas y los nombres de parámetros deben seguir las reglas de nomenclatura de Javascript.

- Deben empezar con una letra, _ o \$
- No pueden contener espacios en blanco
- No pueden ser [palabras reservadas](#).

Clickear una función en la lista accederá a su editor de código. Hacerlo en el botón de edición abrirá el modal de edición, donde se podrá editar el nombre y los parámetros de la función seleccionada. En el botón de borrado permitirá, previa confirmación, borrar la función seleccionada.

Crear/editar función



Function name

Function name is already in use or invalid

Function name (nombre de función): No puede estar vacío, debe seguir las reglas de nomenclatura de Javascript y no puede estar repetido. Si el nombre es repetido o inválido aparecerá un mensaje de error y no dejará crear/editar la función.

Function parameters (parámetros de función): Es optativo, una función puede no tener parámetros. Siguen las reglas de nomenclatura Javascript y una función no puede tener dos parámetros con el mismo nombre. Para agregar un parámetro, se escribe su nombre y luego se ingresa presionando la tecla Enter, la tecla Space o la tecla ,. Para borrar un parámetro, se clickea la x al lado de su nombre.

Si se intenta ingresar un parámetro inválido o repetido, aparecerá un mensaje de error y no se podrá agregar.

Function parameters

Invalid tag(s): 12param

Enter a parameter and press enter a comma or a whitespace to create it

Function parameters

Invalid tag(s): param1

Duplicate tag(s): param1

Enter a parameter and press enter a comma or a whitespace to create it

Asignación y uso de custom functions

Declarando la función en el editor de código deseado la llamará para que se ejecute.

Por ejemplo se crea una función `function_1()` y en su código ingresamos `console.log("hello from function_1")`.



Se va hasta Events -> formEvents -> load y en su editor se ingresa `function_1()`.

Add your own Javascript code to give more functi



Cuando un usuario abra un formulario, cada vez que este cargue, se llamará a la función `function_1` y se ejecutará el código asociado a esta, en este caso un `console.log`.

Hay dos formas de asignar custom functions. Una es ingresarlas manualmente, como se vio en el ejemplo anterior, y funciona tanto para código global, eventos del formulario y eventos de los elementos. La otra forma, que solo sirve para eventos de elementos, es asignar las funciones a través del modal de crear/editar elemento (*ver sección crear/editar elemento*).

Por ejemplo, en el modal de edición de un elemento tipo File, le asigno `function_1` al evento `onclick`.

The image shows a modal titled 'Edit element' for a 'File' type. It contains several configuration fields:

- Element type (*): File
- Element id (*): file_element_ID
- Bootstrap col class (*): col-md-12
- File size (in bytes): [empty]
- File type: all
- Element title: file input
- Element style: [empty]
- Element required: No
- Css class: [empty]

Below these fields is a table for assigning custom functions to events:

Event	Custom function
On click	function_1
On change	-- No function --
On blur	-- No function --
On mouse over	-- No function --
On input	-- No function --

At the bottom of the modal, there are buttons for 'Hide events', 'Save', and 'Close'.

Cuando se vinculan funciones de esta manera, al ir hacia el editor de código del evento, la función se declara al final del código previamente existente y se bloquea el editor. Solo se puede ver el código, sin posibilidad de edición.

```
File onClick(event)
▲ The code cannot be edited because a custom function has been assigned to it. Enable code editor
1 console.log("hello")
2 //previous code here
3
4 function_1()//Function added automatically
5
```

Si se quiere volver a editar manualmente el código, se puede hacer clic en *Enable code editor* (Habilitar editor de código). El editor volverá a estar activo, pero si se regresa al modal de edición del elemento, se puede ver que la función fue desvinculada del evento.

Pestaña de CSS style (estilos de CSS)



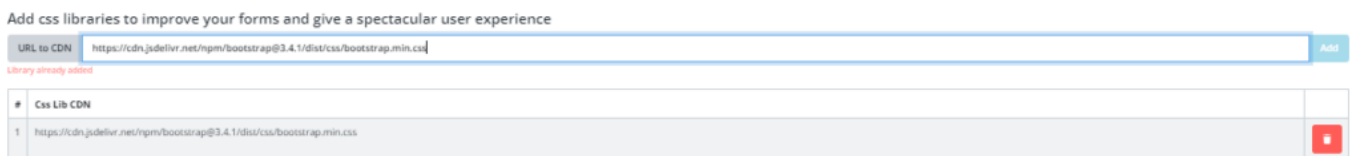
Aquí se puede customizar el estilo y la experiencia de usuario del formulario agregándole estilos de CSS.

Librerías CDN

Se pueden agregar hojas de estilo de terceros mediante [librerías CDN](#).

En *URL to CDN* (Dirección a CDN) se puede ingresar la dirección a la librería deseada. Con el botón Add (Agregar) se agrega a la lista de CDNs agregadas.

No se puede agregar la misma dirección dos veces. Si se lo intenta aparecerá un mensaje de error y no se permitirá agregar la librería.



En la tabla se listarán todas las librerías asignadas, con su dirección y su número de orden en que fueron ingresadas, de primera a última ingresada. El botón rojo con el cesto de basura eliminará la librería seleccionada de la lista.

Editor de estilo de CSS

En este editor se pueden definir manualmente los estilos para la vista del formulario, especificando los selectores cuyos estilos de desean modificar.



Add css libraries to improve your forms and give a spectacular user experience

URL to CDN	https://cdnjs.cloudflare.com/ajax/lib.min.css
#	Css Lib CDN
1	https://cdn.jsdelivr.net/npm/bootstrap@3.4.1/dist/css/bootstrap.min.css

```

1  h1 {
2    font-size: 25px;
3  }
4
5  .form-label {
6    color: red;
7  }
8
9  #first-name-input {
10   border: 1px solid black;
11 }

```

En el ejemplo de la imagen superior, se definió para todos los elementos <h1> del form un tamaño de fuente de 25 píxeles, para todos los elementos con clase form-label un color rojo y para los elementos con ID first-name-input un borde sólido negro de 1 píxel de ancho.

Pestaña de code (código)



Aquí se puede ver y editar el código de los elementos del form, descargar el código .json de las variables, descargar un robot prefabricado que manipule el formulario, y descargar el código .json del formulario.

Visor/editor de código

En el visor se puede ver el código .json de todos los elementos del form en orden de primero a último, con todos sus atributos.

```

JSON code of your form
[{"type":"select","id":"1","container_class":"col-md-12","class":"","label":"","style":"","required":"","options":{"apiurl":"https://jsonplaceholder.typicode.com/todos","api_text_key":"title","api_value_key":"id"},"type":"file","id":"file","container_class":"col-md-12","class":"","label":"file input","style":"","required":"","accept":"","type":"signature","id":"12","container_class":"col-md-12","class":"","label":"signature","style":"","required":"","type":"checkbox","id":"check","container_class":"col-md-12","class":"","label":"checkbox","style":"","required":""}]

```

Presionar el botón verde a la derecha habilitará el modo de edición. En el código se formateará de forma que su visualización y edición sea sencilla, como un elemento por bloque. Volver a clicar el botón cerrará el modo de edición, cambiará el formato a texto plano y guarda todos los cambios realizados.

Si se introduce código incorrecto, aparecerá un mensaje de error y al cerrarse el editor los cambios realizados desde la última vez que se habilitó el editor de código no se aplicarán.

Botón de Download vars (descargar variables)

Crea todas las variables necesarias para que el robot autocomplete con los datos enviados al queue, genera y descarga un archivo .json, con cada uno de los elementos del form, siendo su atributo *name* el ID del elemento.

Botón de Download robot prefab (Descargar robot prefabricado)

Crea y descarga un robot en formato .json para cargarlo en Rocketbot Studio.

Botón de Export form (exportar formulario)

Crea un archivo .json con el formulario y lo descarga, con sus elementos, librerías y código Javascript/CSS, eventos y funciones personalizadas.

AutoCreator

AutoCreator es un asistente de Rocketbot para la creación de Robots para la automatización de procesos y Formularios para la plataforma Rocketbot Xperience.

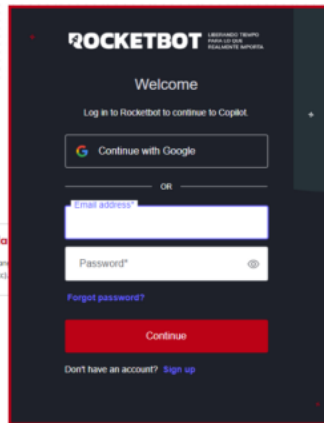
Para ingresar existen dos maneras:

- Directamente desde el link <https://copilot.rocketbot.com/#/>
- Mediante el Addon correspondiente desde Rocketbot Studio

Boost your efficiency

With Rocketbot Autocreator you will be able to create robots much faster than manually. Autocreator will help you create the necessary steps to automate your process, you can add drag and drop commands directly in Rocketbot Studio with very little effort.

Start creating your robot now
[Click to Start](#)
Version 1.7.0 beta preview



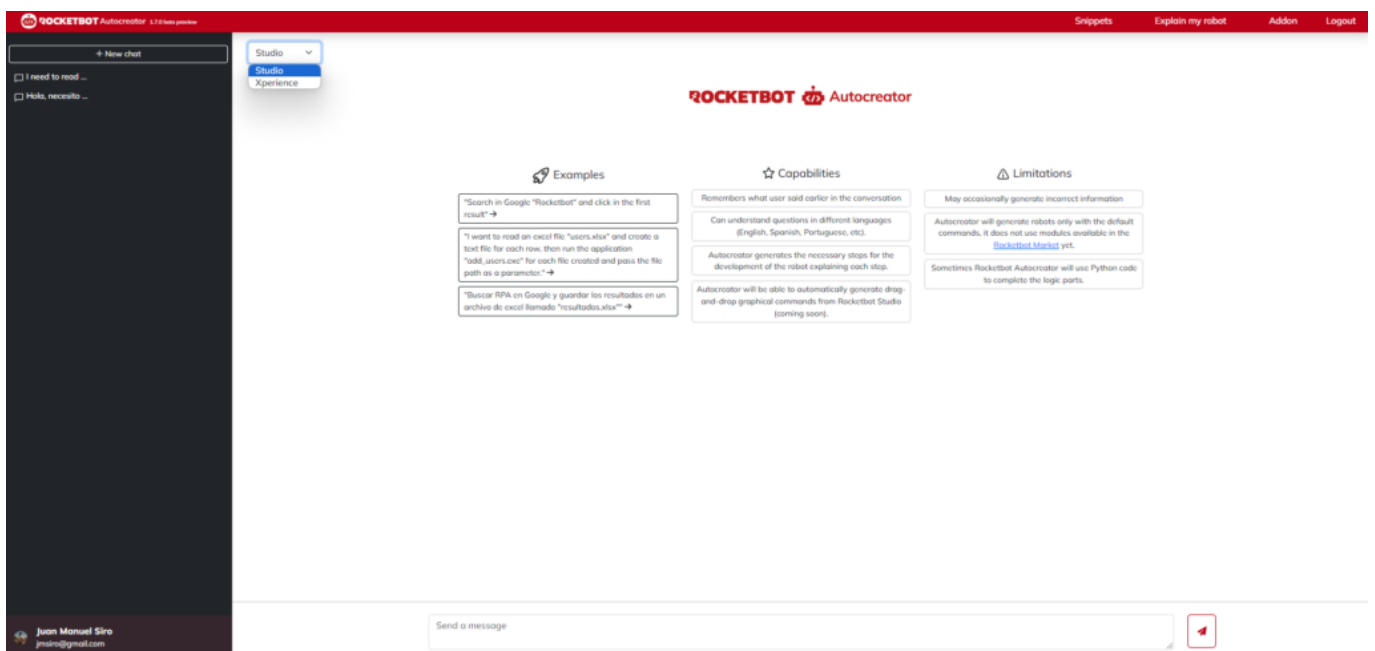
Memory
Remembers what user said earlier in the conversation

Multilingual
Can understand questions in different languages (English, Spanish, Portuguese, etc.)

Ready for Rocketbot Studio
Autocreator will be able to automatically generate drag-and-drop graphical commands

Al ingresar realizar el login desde el botón Click to Start.

Una vez realizado el login, será redirigido al menú de chats. Allí puede elegir la plataforma con la cual desea que la IA lo asista: **Studio** o **Xperience**.

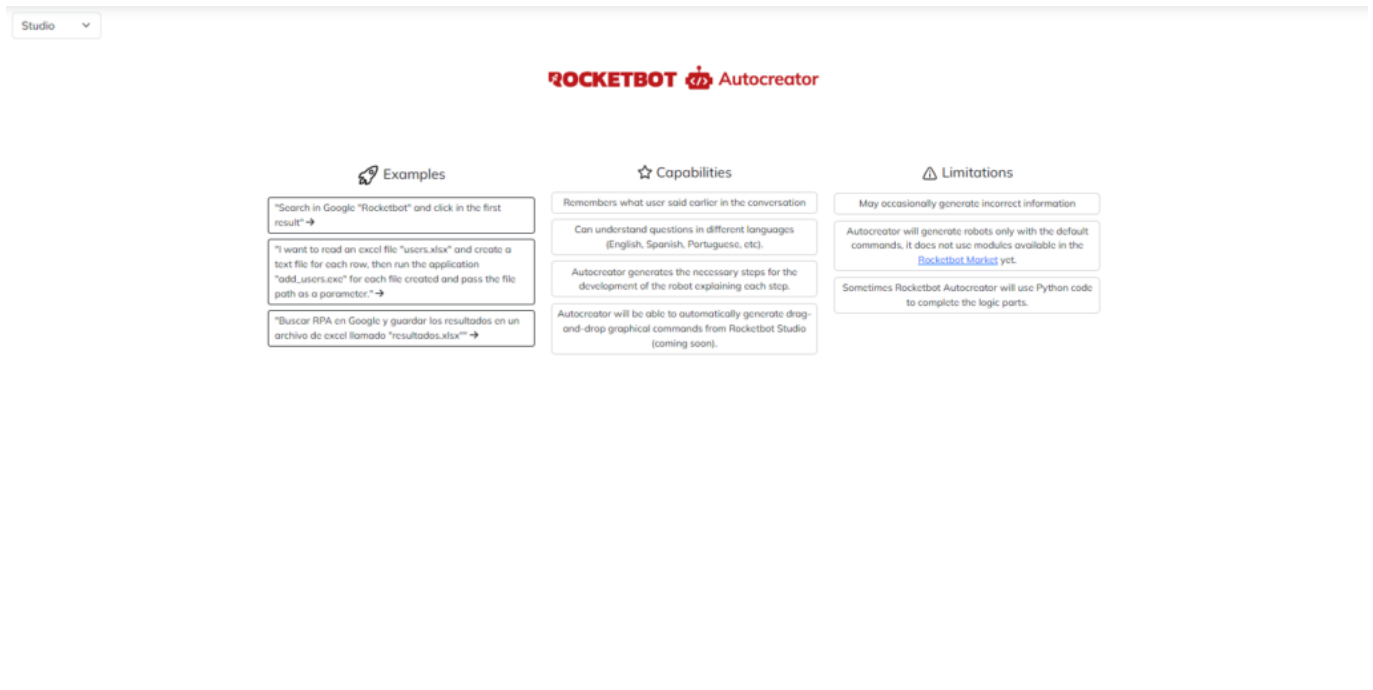


Si selecciona **Xperience**, a la izquierda del recuadro del chat, se habilitará el botón para cargar la imagen sobre la cual la IA deberá basarse para crear el formulario.

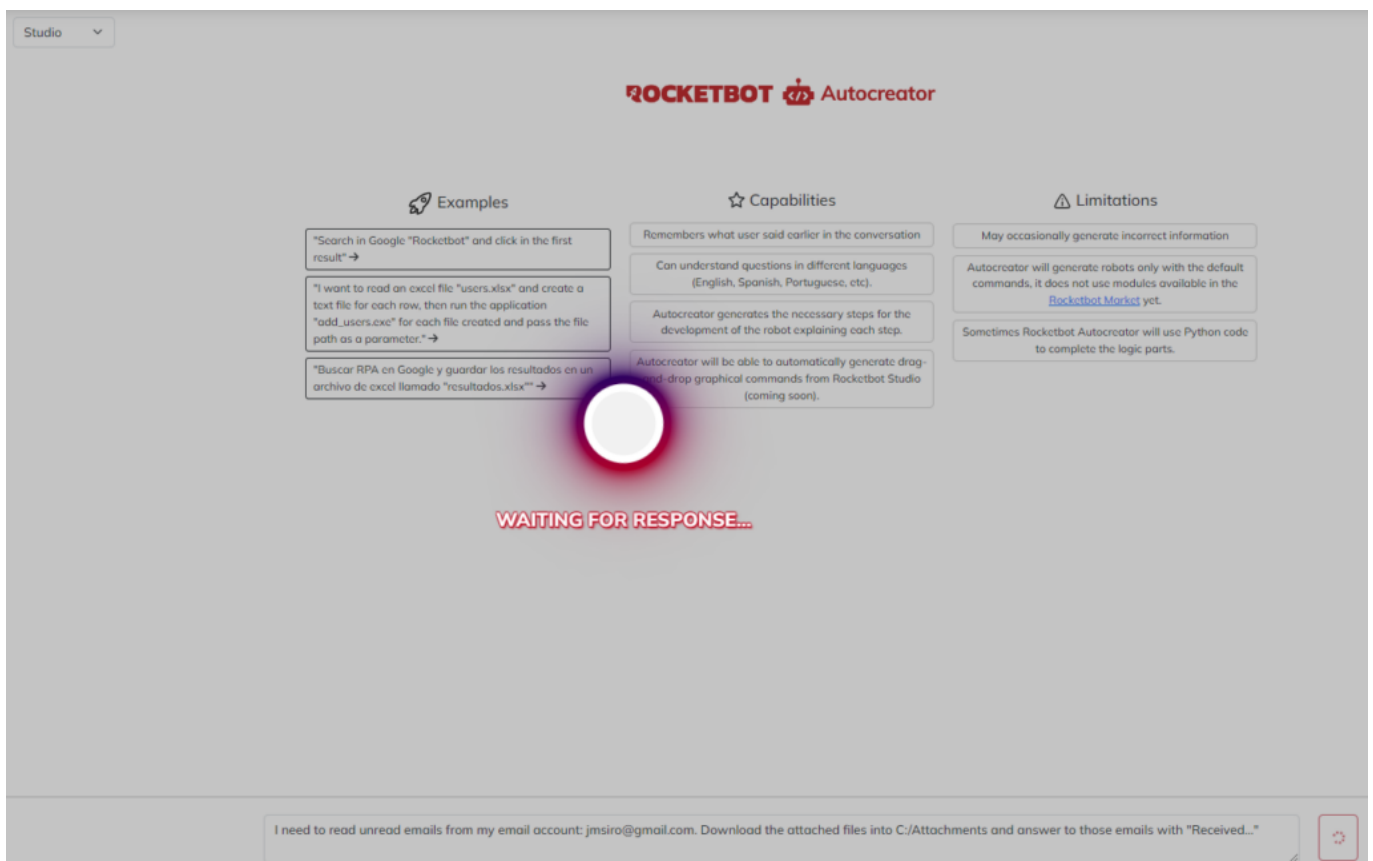


Crear un Robot – ██████████ :

Seleccionar **Studio** y escribir la solicitud.



Enviar la consulta y aguardar la respuesta de la IA.






La IA responderá con el listado de pasos que interpreta necesarios para cumplir con la solicitud. El usuario dispondrá de 2 opciones:

- Generar el robot tomando los pasos propuestos como base.
- Solicitar a la IA que genere una nueva serie de pasos.

I need to read unread emails from my email account: jmsiro@gmail.com. Download the attached files into C:/Attachments and answer to those emails with "Received..."

 I need to read unread emails from my email account: jmsiro@gmail.com. Download the attached files into C:/Attachments and answer to those emails with "Received..."

< 1 / 1 >  **Proceso**  

Paso a paso sugerido

Entorno	Detalle
Gmail	Configurar conexión con la cuenta de correo electrónico jmsiro@gmail.com
Gmail	Leer correos electrónicos no leídos
Gmail	Descargar los archivos adjuntos de los correos no leídos en la carpeta "C:/Attachments"
Gmail	Responder a cada correo no leído con el mensaje "Received..."

Generate bot Regenerate steps


 Regenerate


Send a message



Seleccionar la opción deseada y esperar la respuesta de la IA.


I need to read unread emails from my email account: jmsiro@gmail.com. Download the attached files into C:/Attachments and ans


 I need to read unread emails from my email account: jmsiro@gmail.com. Download the attached files into C:/Attachments and answer to those ema

< 1 / 1 >  **Proceso**

Paso a paso sugerido

Entorno	Detalle
Gmail	Configurar conexión con la cuenta de correo electrónico jmsiro@gmail.com
Gmail	Leer correos electrónicos no leídos
Gmail	Descargar los archivos adjuntos de los correos no leídos en la carpeta "C:/Attachments"
Gmail	Responder a cada correo no leído con el mensaje "Received..."


 Regenerate steps






SENDING DATA...

La IA responderá con el código del robot generado.

I need to read unread emails from my email account: jmsiro@gmail.com. Download the attached files into C:/Attachments and answer to those emails with "Received..."

 I need to read unread emails from my email account: jmsiro@gmail.com. Download the attached files into C:/Attachments and answer to those emails with "Received..."




<1/1>  **Proceso**  

Paso a paso sugerido

Entorno	Detalle
Gmail	Configurar conexión con la cuenta de correo electrónica jmsiro@gmail.com
Gmail	Leer correos electrónicos no leídos
Gmail	Descargar los archivos adjuntos de los correos no leídos en la carpeta "C:/Attachments"
Gmail	Responder a cada correo no leído con el mensaje "Received..."

Here is the Rocketbot code to read unread emails from the specified email account, download the attachments to a local folder, and respond to those emails with a message:

Paso	Descripción
1	Configure email account for IMAP and SMTP.
2	Get all unread email IDs.
3	Check if there are unread emails.
4	Iterate over each unread email.
5	Read the email and download attachments.
6	Send a reply with "Received..."
7	Save status of email reply.

 Code Structure  



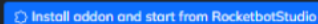
```
#robot_name: email_reader_responder
#description: Read unread emails, download attachments, and respond with "Received..."

vars = {
  'email_user': 'jmsiro@gmail.com',
  'email_password': '*****', # Replace with the actual password
  'attachments_path': 'C:/Attachments',
  'unread_emails': [],
  'email_id': '',
  'reply_message': 'Received...',
  'send_status': False
}

# Configure email account for IMAP and SMTP
```

Send a message

El usuario podrá descargar el robot en formato JSON o agregarlo directamente a Rocketbot Studio.

 Code Structure  

```
#robot_name: email_reader_responder
#description: Read unread emails, download attachments, and respond with "Received..."

vars = {
  'email_user': 'jmsiro@gmail.com',
  'email_password': '*****', # Replace with the actual password
  'attachments_path': 'C:/Attachments',
  'unread_emails': [],
  'email_id': '',
  'reply_message': 'Received...',
  'send_status': False
}

# Configure email account for IMAP and SMTP
email.configemail(setp='smtp.gmail.com', setp_port=587, imap='imap.gmail.com', imap_port=993, ssl=True, imap_ssl=True, user=vars['email_user'], password=vars['email_password'])

# Get all unread email IDs
vars['unread_emails'] = email.getunreademail(command='UNSEEN')

# Check if there are unread emails
if len(vars['unread_emails']) > 0:

  # Iterate over each unread email
  for vars['email_id'] in vars['unread_emails']:

    # Read the email and download attachments
    email.reademail(id=vars['email_id'], file=vars['attachments_path'])

    # Send a reply with "Received..."
    vars['send_status'] = email.sendemail(to=vars['email_user'], subject='Re: ', msg=vars['reply_message'], file='')

    # Check if email was sent successfully and write status
    if vars['send_status']:
      # Status can be logged or used as needed
      pass
    else:
      # Handle failed email send, if necessary
      pass

else:
  # If no unread emails, log or perform other actions as needed
  pass
```

Para enviarlo directo a Rocketbot Studio, deberá abrir AutoCreator desde el addon correspondiente.

Con **Create bot** la IA responderá la solicitud de manera independiente a los comandos que se posean en el robot abierto en Rocketbot Studio.

Con **Add commands** la IA responderá la solicitud teniendo en cuenta los comandos que ya posea el robot desde donde se llame AutoCreator.



Create your bot with AutoCreator.

Create bot

Add commands to your bot.

Add commands

```
Code Structure Generate JSON Create Bot

#robot_name: email_reader_responder
#description: Read unread emails, download attachments, and respond with "Received..."

vars - {
  'email_user': 'jesiro@gmail.com',
  'email_password': '*****', # Replace with the actual password
  'attachments_path': 'C:/Attachments',
  'unread_emails': [],
  'email_id': '',
  'reply_message': 'Received...',
  'send_status': False
}

# Configure email account for IMAP and SMTP
email.configemail(setp='setp.gmail.com', setp_port=587, imap='imap.gmail.com', imap_port=993, ssl=True, imap_ssl=True, user-vars['email_user'], password-vars['email_password'])

# Get all unread email IDs
vars['unread_emails'] - email.getunreademail(command='UNSEEN')

# Check if there are unread emails
if len(vars['unread_emails']) > 0:

  # Iterate over each unread email
  for vars['email_id'] in vars['unread_emails']:

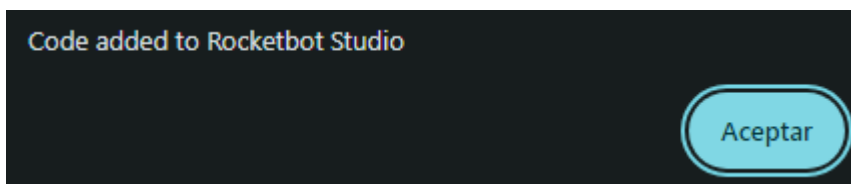
    # Read the email and download attachments
    email.reademail(id=vars['email_id'], file=vars['attachments_path'])

    # Send a reply with "Received..."
    vars['send_status'] - email.sendemail(to=vars['email_user'], subject='Re: ', msg=vars['reply_message'], file='')

    # Check if email was sent successfully and write status
    if vars['send_status']:
      # Status can be logged or used as needed
      pass
    else:
      # Handle failed email send, if necessary
      pass

else:
  # If no unread emails, log or perform other actions as needed
  pass
```

Presionar **Create Bot** para agregar los comandos en el robot abierto en Studio. Una vez completada la operación, un Alerta aparecerá en el navegador indicando que los comandos han sido agregados.



Al volver a Rocketbot Studio se encontrará con los comandos y las variables generados por la IA creadas

The screenshot displays the Rocketbot Studio interface. At the top, there are navigation tabs for Home, New, Tests, Run, and Stop. Below this, a menu bar includes Events, Variables, Robot, History, Robots in Project, and Expose. The main workspace shows a 'Sequence of commands - AC_Test' with five numbered steps:

- Open browser: Select browser: chrome ("url": "url")
- Wait for object: ... xpath object: //input@class= 'wait_time': 10; wait_for: visible
- Send keys to web: Text data: //input@class= 'data_type': xpath: /text: [search_term]; special: ENTER
- Wait for object: ... xpath object: //div@class= 'wait_time': 10; wait_for: visible
- Click: Click on the first 20 results

Below these steps, there are three more commands for data extraction and storage:

- Click: Click on the name of the product
- Click: Click on the price of the product
- Add data to array: array_ [product_name] + value_ [product_name] + var_ [product_name]

On the right side, the 'Rocketbot Copilot' sidebar is active, showing a message: 'Rocketbot AutoCreator has opened in a new tab.' Below this, there are buttons for 'Create bot' and 'Add commands to your bot'.

Crear un Formulario Xperience – Paso a paso:

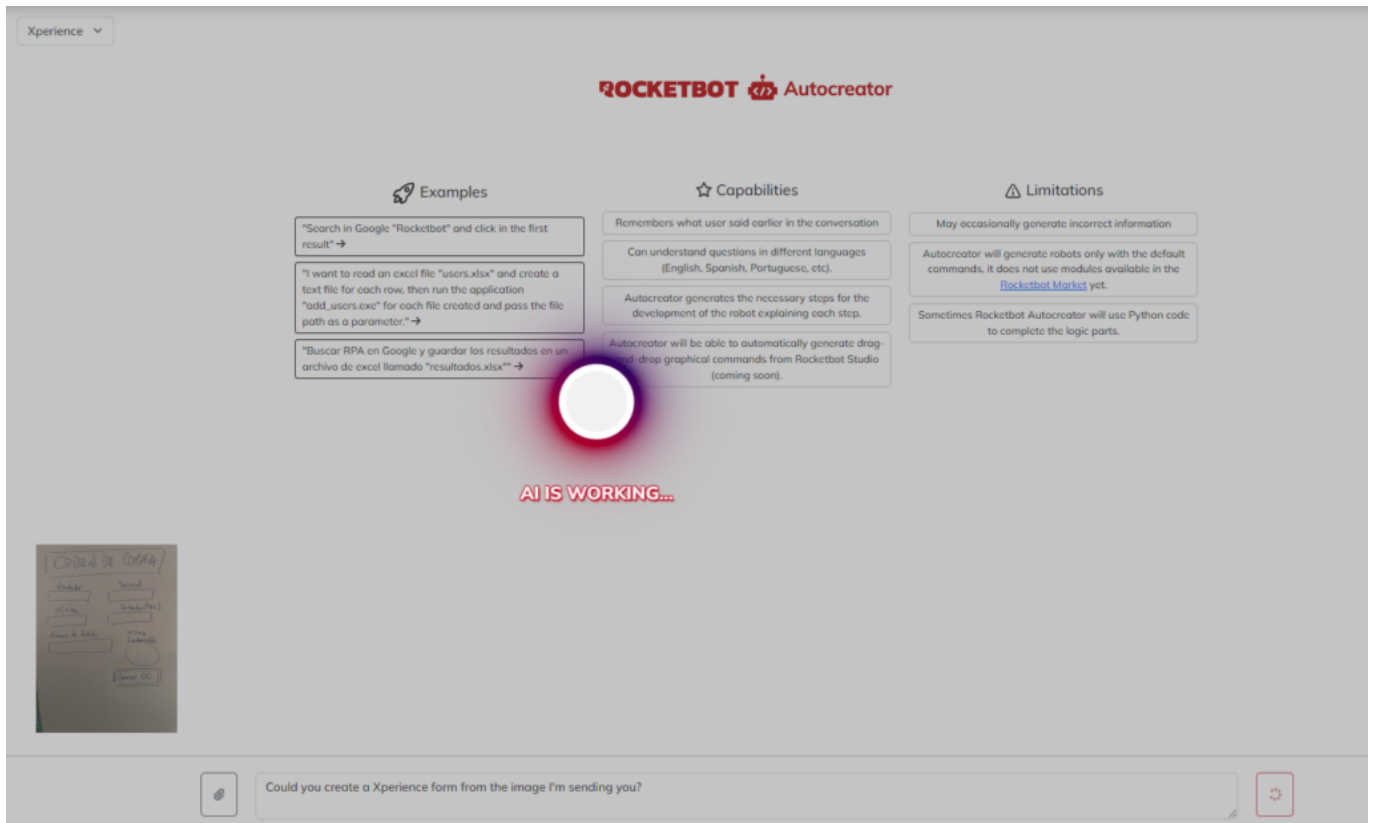
Seleccionar **Xperience**, presionar el botón con el clip para seleccionar la imagen y enviar la solicitud.

The screenshot shows the Rocketbot Autocreator interface. At the top, the 'ROCKETBOT Autocreator' logo is visible. Below the logo, there are three main sections:

- Examples:**
 - "Search in Google "Rocketbot" and click in the first result" →
 - "I want to read an excel file "users.xlsx" and create a text file for each row, then run the application "add_users.exe" for each file created and pass the file path as a parameter." →
 - "Buscar RPA en Google y guardar los resultados en un archivo de excel llamado "resultados.xlsx" →
- Capabilities:**
 - Remembers what user said earlier in the conversation
 - Can understand questions in different languages (English, Spanish, Portuguese, etc).
 - Autocreator generates the necessary steps for the development of the robot explaining each step.
 - Autocreator will be able to automatically generate drag-and-drop graphical commands from Rocketbot Studio (coming soon).
- Limitations:**
 - May occasionally generate incorrect information
 - Autocreator will generate robots only with the default commands. It does not use modules available in the [Rocketbot Markup](#) yet.
 - Sometimes Rocketbot Autocreator will use Python code to complete the logic parts.

At the bottom of the interface, there is a text input field with a question: "Could you create a Xperience form from the image I'm sending you?". To the right of the input field is a red arrow button.

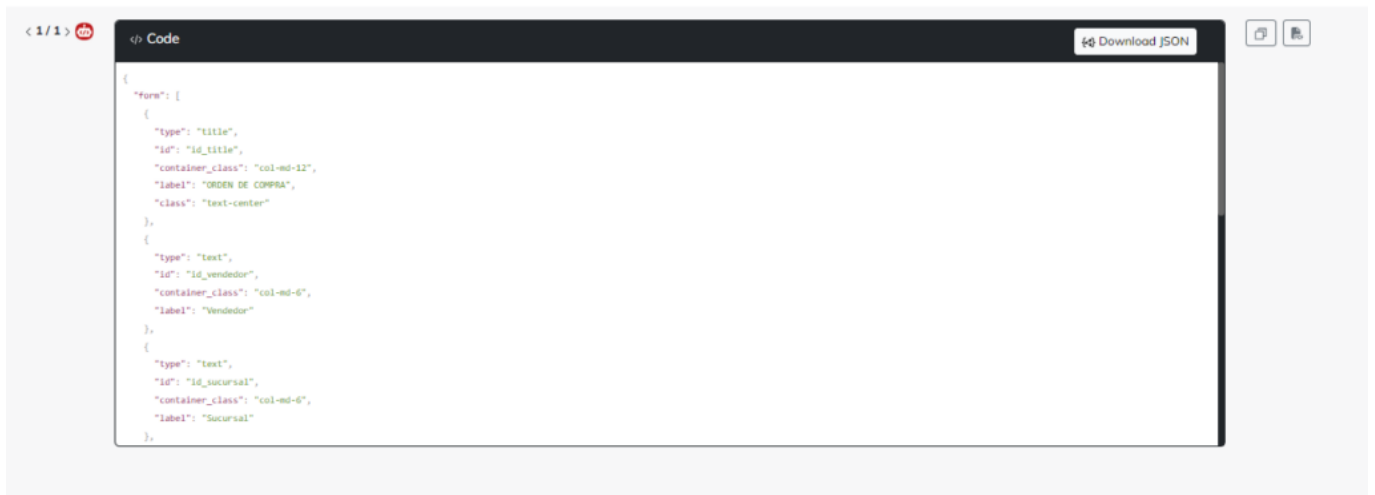
Enviar la solicitud y aguardar la respuesta de la IA.



La IA responderá con el código del formulario generado.

☐ Could you create a Xperience form from the image I'm sending you?

🤖 Could you create a Xperience form from the image I'm sending you?



Descargar **JSON** y crear un nuevo formulario en el **Orquestador** desde archivo.

Create a new Form



Create an amazing form to collect data easily.

Form name:

testAutoCreator

- Start from scratch
- Import form from file
- Based on other form (coming soon)

form.json

Browse



CLOSE

CREATE FORM

Crear formulario y publicar.



ORDEN DE COMPRA

Vendedor

Sucursal

Cliente

Contacto (TEL)

Número de Pedido

Firma Responsable

Submit Settings

Submit button text:

Generar OC

Redirect to after submit:

-- No redirect --


Generar OC

[Como recibir información desde un](#)

robot en formularios Xperience

La funcionalidad principal de [Xperience](#) es enviar los datos ingresados por el usuario y enviarlos al robot para que utilice esa información, pero algunas veces necesitamos que el robot sea quien envíe información de vuelta al formulario para mostrar información al usuario.

En el formulario

1. Crear un formulario o ir a un formulario existente
2. Habilitar la opción **Send API**
3. En el menú Javascript, agregar la siguiente librería → [xperience.js](#)

4. En la consola de Javascript agregar lo siguiente

```
rocket.on('start', (xperience, data) => {  
  console.log('DATA: ', data) // esta línea te permite ver el resultado en la  
  consola del formulario  
  //Acá el código inicial, lo que envía el robot la primera vez  
})
```

```
rocket.on('change', (xperience, res) => {  
  if (res.success) {  
    console.log('RES: ', res) // esta línea te permite ver el resultado en la  
    consola del formulario  
    //Acá la información que estará enviando el robot posteriormente  
  }  
});
```

En el robot

1. Crear una variable llamada **xperience**. Esta variable la llenará el comando **Get Form queue data** del módulo [Xperience](#)
2. Utilizar el comando **Return Message to Xperience** para enviar un mensaje de vuelta al formulario

☐ Puedes enviar cualquier tipo de dato permitido por Rocketbot y este se enviará como un string.

Es importante hacer un `JSON.parse` desde el formulario al recibir los datos.

```
result = JSON.parse(res.data.replaceAll('"', ''));
```

3. Utilizar el comando **Send File to Xperience** para enviar un archivo y alojarlo en Xperience

□ Puedes obtener el archivo mediante postman con el siguiente endpoint de lectura:

```
GET: /api/form/extraFile/{xperience}/{token}
```

Siendo {xperience} la variable generada en el punto nro 1, y {token} el resultado del comando Send File to Xperience

Ejemplo

El ejemplo te permite mostrar una ventana de alerta con los datos retornados desde el robot

□Para que te funcione el código, tienes que agregar esta url a los CDN:
//cdn.jsdelivr.net/npm/sweetalert2@11

```
rocket.on('change', (xperience, res) => {
  if (res.success) {
    console.log('RES: ',res) // esta línea te permite ver el resultado en la
    consola del formulario
    if(res.data){
      swal.fire({
        title: 'Estos son los datos enviados desde el robot ' + res.data,
        icon: 'info',
      });
    }
  }else{
    swal.fire({
      title: 'Ocurrió un error al solicitar datos',
      icon: 'error'
    });
  }
});
```