

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"
```


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)
```

Ejemplo de response

```
{
  "success": true,
  "data": {
    "id": 8728,
    "data": "{\"Dato1\":\"prueba\",\"Dato2\":\"prueba1\"}",
    "user_form_email": null,
    "xperience":
"eyJpdiI6InNxMXhRekhVUUJrWVptcGpzOUM1aXc9PSIsInZhbnVlIjoiY1NoMGtpcW5udk5vTmZx
VlVYZFg1V29CUGU4SDB1UVUwWuXWVYVjYVNmM2FVNkE2OE5MMnRwclorWGxtMU9VZ2pjTGJnSUNLaE5cL
1ArSkpPdk1BendRYVlVtTlMyU0QwclN0E2VENmSDEyZTJRPSIsIm1hYyI6ImYyYjc2ZmM30TAwMW
QzNDVlMmViNmM1NTIxNjc0OTc0Y2IwOTA2Yjg4YTMwODNlODUwNzU1NWQ4MmM2ZGRmMjcifQ=="
  }
}
```

Obtener Asset

Se utiliza para solicitar información de un asset almacenado en el orquestador. La búsqueda se realiza filtrando por nombre e instancia a la que pertenece.

Método URL	Formato	Parámetros
POST	https://www.myrb.io/dev/api/assets/get	Authorization: tipo Bearer Token; es el Token de acceso válido para autenticar al usuario (API Key asociada al perfil en el orquestador). name: El nombre del asset. instance: Key de la instancia, tal como se define en el archivo noc.ini.

Avanzado

Ejemplos de request

- cURL

```
curl --location 'https://www.myrb.io/dev/api/assets/get' \
--header 'Authorization: Bearer <token>' \
--data-urlencode 'name=toNotify' \
```

```
--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 : Procesos](#)

¿Qué es un proceso?

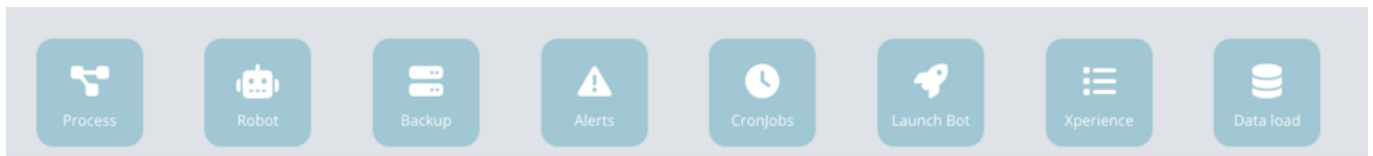
Un proceso es donde se configurará el robot y la automatización.

¿Cómo se crea o se asigna un proceso?

Para crear un proceso sigue el paso a paso del siguiente documento: [Creación de tu primer proceso](#).

Una vez el proceso este creado, al ingresar al mismo encontraremos lo siguiente :

Interfaz



Process

Aquí se puede definir el nombre y la descripción, y cuenta con el *token*, el cual se puede copiar con acceso rápido. En la parte inferior encontramos los botones para borrar proceso, editar y cerrar.

Edit process

Name

Description

Token

Robot

Aquí se debe asignar el nombre para identificar al robot dentro del orquestador. En *Start robot name* se debe asignar el nombre del robot a ejecutar, que debe coincidir exactamente con el de la base de datos; de lo contrario, no se encontrará el robot a automatizar.

También es necesario asignarle la base de datos del robot.

Edit robot

Robot name

prueba orquestador

Robot name must be at least 3 characters long, alphanumeric and whitespaces only.

Start robot name

bot_orquestador

Uplad DB

Choose File

No file chosen



Add a process and load a robot to run in one or more instances.
You can load a robot by exporting to DB production or a DB project.

Close

Save

Backup Instance

Aquí podemos seleccionar una instancia de respaldo, en caso de que la principal se caiga, y los minutos a partir de los cuales deberá activarse si la principal no se reconecta.

Backup instance

Select instance

-- Select --



Minutes to activate

0

Close

Save

Alerts

En esta sección, se puede configurar una notificación ante una desconexión de la instancia principal. Aquí se deben poner los correos electrónicos donde se

enviarán las alertas y los minutos posteriores a la eventualidad (sin que haya reconexión) en los que se disparará la alerta.

Edit process alerts email

Email

Add tag...



Email separated by comma: `user1@mail.com,user2@mail.com`

Minutes

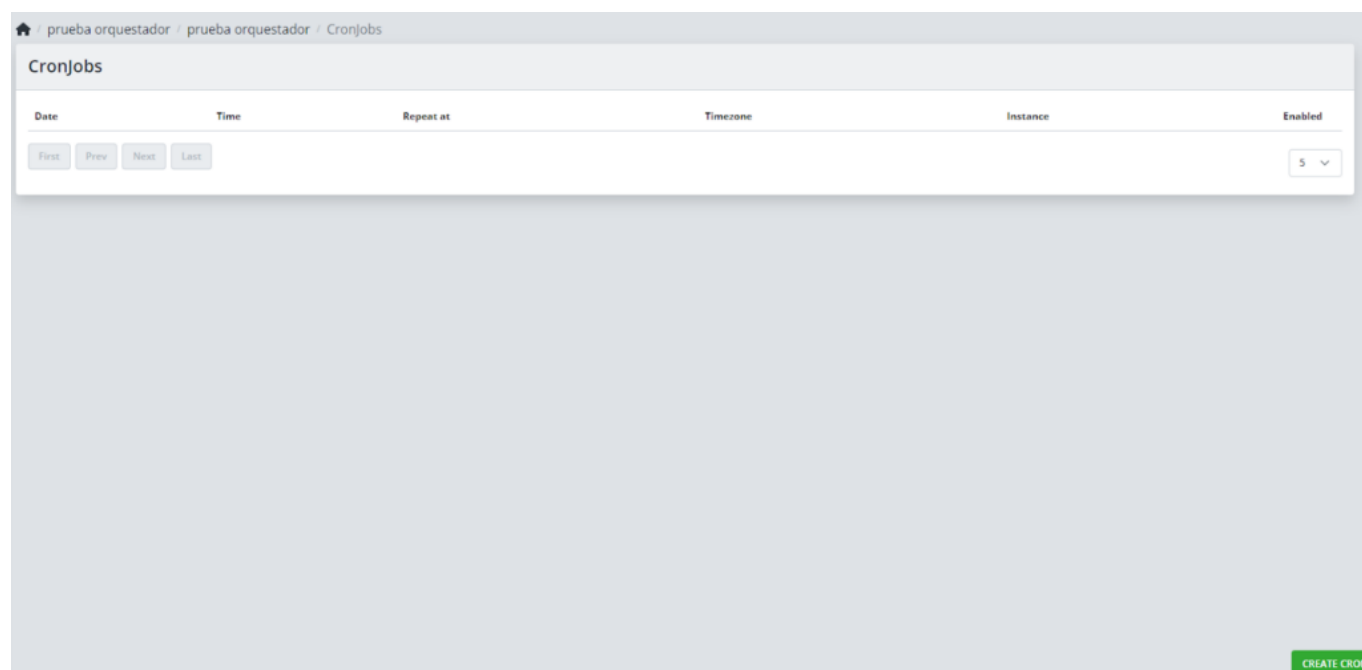
0

Close

Save

Cronjobs

En esta sección se podrán ver las ejecuciones programadas para cada proceso y configurar nuevas realizando click en [\[Create Cron \]](#).



Date	Time	Repeat at	Timezone	Instance	Enabled
------	------	-----------	----------	----------	---------

Para mayor detalle, puede revisar el siguiente documento: [Trabajar con Cronjobs](#)

Launch Bot (*trigger*)

Esta opción permite seleccionar otro proceso para que inicie, o se dispare, cuando finalice el proceso que se esta configurando.

Set trigger bot

Process



Select another process to start when the instances finish.

Close

Save

Xperience

Esta opción permite seleccionar un formulario Xperience para que lance la ejecución del proceso cuando se envíe dicho formulario, ya sea de manera manual o mediante API.

Set Xperience form

Form

No trigger



Select a form to trigger the process when you receive some raw queue.

Close

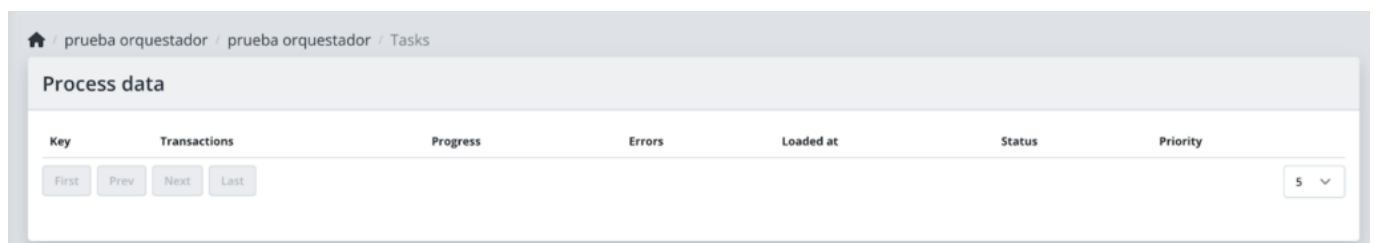
Save

Data Load

Aquí encontraremos las tareas y transacciones asignadas al proceso, utilizando el módulo Orchestrator Framework desde otro robot, y su estado actual.

Si necesita mas informacion sobre Data Load puede revisar el siguiente documento :

[Orquestador Rocketbot : Data Load](#)



Key	Transactions	Progress	Errors	Loaded at	Status	Priority
First	Prev	Next	Last			5

Asignar una instancia

Para agregar una o mas instancias donde ejecutar tu proceso sigue el paso a paso del siguiente documento: [Agregar Instancia](#).

¿Cómo paralelizar?

Cuando hablamos de paralelizar debemos distinguir 2 escenarios:

1. Paralelizar un mismo proceso en diferentes instancias
2. Paralelizar diferentes procesos en una misma instancia

El robot de un proceso se puede ejecutar en múltiples instancias, el orquestador enviará la señal de ejecución a todas las instancias asignadas y conectadas al proceso (Salvo las de backup). Así mismo una misma instancia puede estar asignada a múltiples procesos y en la misma ejecutar el robot de cada uno de ellos, en momentos diferentes o al mismo tiempo, según el proceso lo permita o no (LINK). A su vez, cabe mencionar que no se puede ejecutar un robot varias veces al mismo tiempo en una instancia.

Para saber mas acerca de que procesos pueden paralelizarse:

[¿Qué considerar para ejecución en paralelo?](#)

Queues de un proceso

Las queues que figuran en el dashboard de procesos representan ejecuciones pendientes del mismo que se generan cuando un proceso diferente ha disparado al que contiene estas colas. Las colas o queues del proceso se generan cuando el mismo no tiene una instancia disponible para ejecutar el robot al momento en que el se produjo el “disparo”.

En otras palabras, sólo se producen cuando un proceso tiene configurado un trigger a otro, al finalizar el primero, se ejecuta el trigger y este aguarda a tener una instancia disponible.

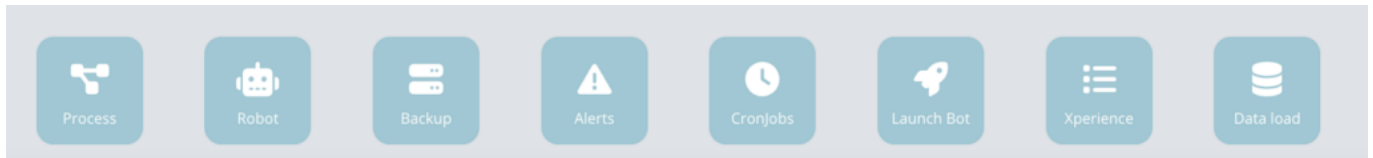
[Orquestador Rocketbot : Trabajar con cronjobs](#)

¿Qué es un cronjob?

El cronjob de un proceso es una o más reglas que determinan la fecha, hora y periodicidad de ejecución del mismo.

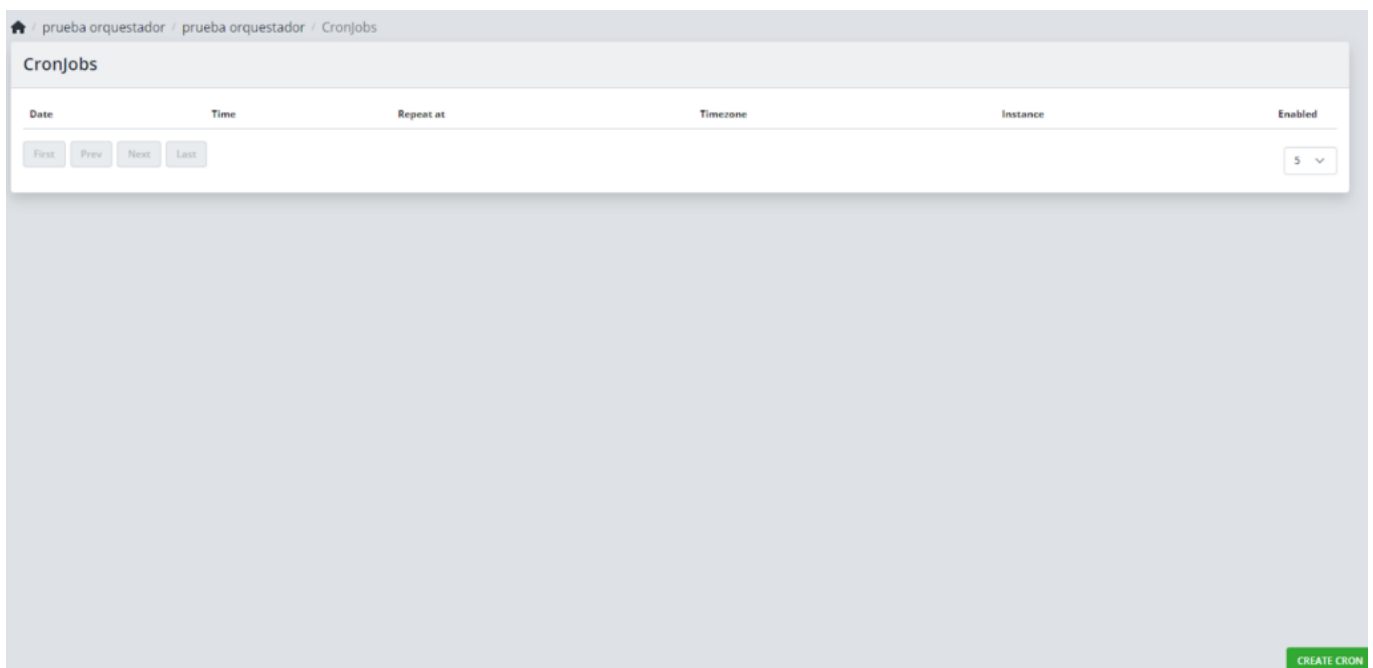
¿Cómo crearlo?

Al ingresar al proceso en el orquestador, encontraremos la sección de cronjobs ubicada en el menú superior.



Cronjobs:

En esta sección se encontrará un tablero con el listado de cronjobs definidos para el proceso y, en la esquina inferior derecha, estará el botón para crear nuevos.



Crear Cronjob:

Opciones básicas:

- Date: Fecha de implementación
- Time: Hora de ejecución
- Timezone: Zona horaria del ambiente de ejecución
- Repeat at: Periodicidad de ejecución
- Instance: Instancia asignada de las disponibles en el proceso
- Enabled: Opción que determina si el cronjob esta habilitado o no

Opciones avanzadas:

Estas opciones nos permiten definir más específicamente:

- Cada cuanto repetir el proceso (Una vez, diariamente, semanalmente, mensualmente). A su vez, es se podrá especificar en detalle cuando, por ejemplo, al seleccionar semanalmente se puede seleccionar los días de la semana en los que se deberá ejecutar.
- Durante cuánto tiempo, permite definir una fecha de expiración de la regla.
- Con que frecuencia se ejecutará, por ejemplo, ejecutar cada una hora durante 12 horas.

¿Cómo funciona un Cronjob?

Cuando llega la fecha y hora configurada en un cronjob, el orquestador comprueba si existen instancias disponibles para ejecutar el proceso. En caso afirmativo, ejecutará el robot del proceso en todas las que estén disponibles, de las asignadas a dicho proceso.

IMPORTANTE: Si no hay instancias disponibles al momento de ejecución del Cronjob, el orquestador repetirá la comprobación cuando vuelva a coincidir el día y hora, es decir, se omite.

Opciones Cronologicas de un Cronjob

Frecuencia de ejecución del proceso:

Únicamente: El cronjob se ejecuta una sola vez en la fecha y hora definidas, y no se vuelve a ejecutar.

Diariamente: Se puede configurar para repetirse cada “X” días.

Advanced mode ▲

Will activate every...

- One time
- Daily
- Weekly
- Monthly

Repeat every days

Semanalmente: Se puede configurar para repetirse cada “X” semanas. Además, permite seleccionar los días de la semana en los que se desea que se ejecute el robot.

Advanced mode ▲

Will activate every...

- One time
- Daily
- Weekly
- Monthly

Repeat every weeks on

Sunday Monday Tuesday Wednesday Thursday Friday Saturday

Mensualmente: Se puede configurar para ejecutarse en uno o varios meses específicos, con dos tipos de configuraciones:

- **Días específicos del mes:** Por ejemplo, el 1 de enero, el 3 de marzo, etc.

Advanced mode ▲

Will activate every...

- One time
- Daily
- Weekly
- Monthly

Months

January March May July August September October November December

Days

1 3 5 7 9 12 30

On

—Weeks— —Days—

- **Semanas específicas del mes:** Permite elegir en qué semana o semanas del mes se ejecutará el robot, y además seleccionar los días de la semana para esa ejecución. Por ejemplo, que se ejecute los lunes y miércoles de la primera y tercera semana de enero y marzo.

Advanced mode ▲

Will activate every...

- One time
- Daily
- Weekly
- Monthly

Months

January March May July August September October November December

Days

—Days—

On

First Third Last

Monday Wednesday Friday

Fecha de expiración del cronjob

Esta opción permite definir una fecha y hora en la que el cronjob dejará de ejecutarse y, por lo tanto, el robot dejará de activarse.

Set date of expiration

Expiration date

06/05/2025

Expiration hour

01:00 PM

Bucle

Permite asignar un bucle para que el cronjob active el robot cada "X" tiempo y lo haga durante un periodo de "Y" tiempo.

Ejemplo: Que el robot se ejecute cada 1 hora durante un día completo.

Set frequency of execution

Repeat each One hour

▼

for One day

▼

Orquestador Rocketbot : Expiración User ApiKey

¿Por qué expira?

La API Key del Orquestador de Rocketbot expira en dos años como una medida de seguridad.

Este límite de tiempo ayuda a proteger la integridad del sistema al asegurarse de que las claves de acceso no se utilicen indefinidamente, lo que podría aumentar el riesgo de exposición o uso indebido.

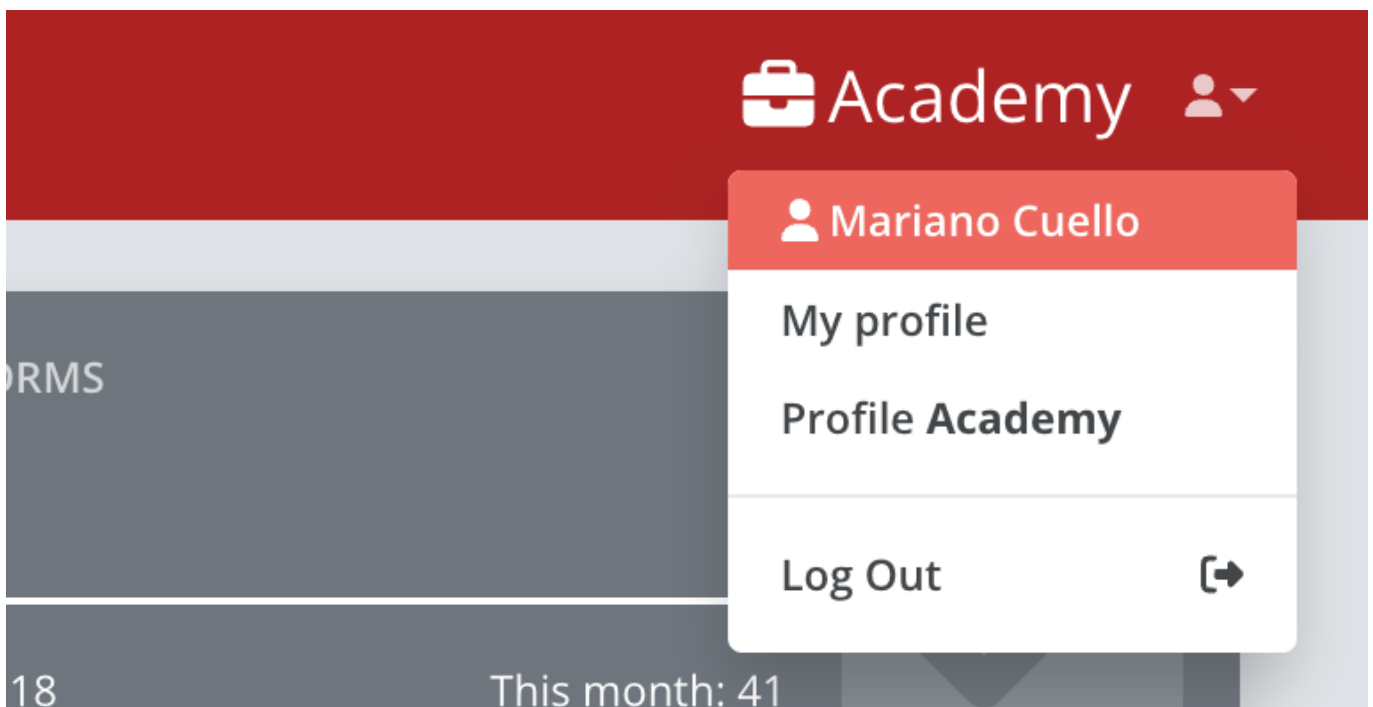
Al forzar la renovación periódica de las API Keys, se obliga a los administradores a revisar y actualizar las claves, asegurando que solo las personas y sistemas autorizados tengan acceso continuo. Además, esto permite implementar mejoras en la seguridad y cambios en las políticas de acceso de manera regular.

¿Dónde la encuentro?

Para saber esta información debemos acceder al perfil del usuario.



Hacer clic en *My Profile*.



En la última casilla se encontrará la fecha de expiración de su API Key.

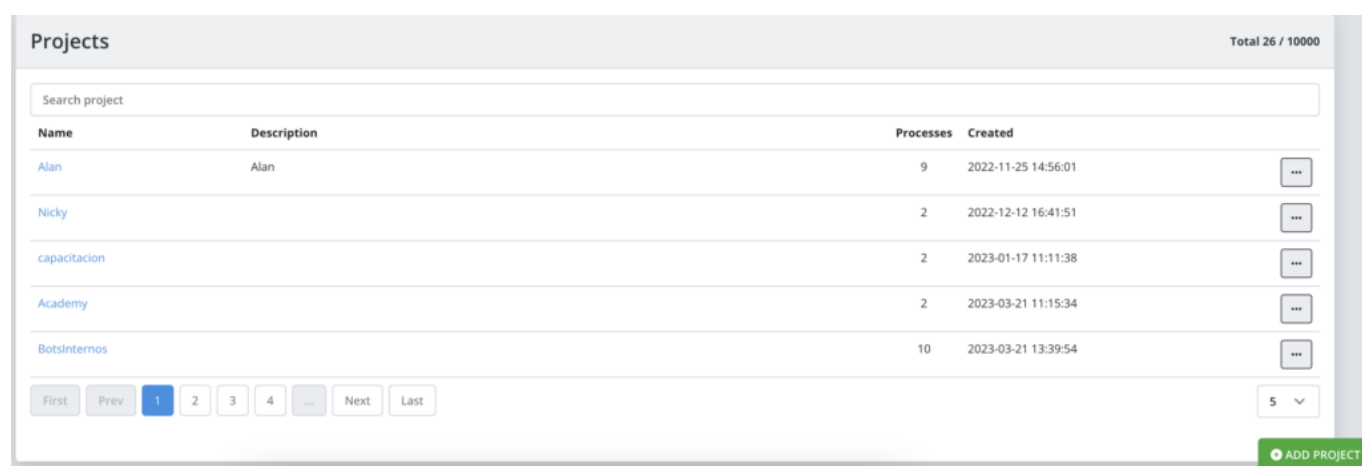
API key expiration: 8/7/2026, 10:44:06 AM

Si está interesado en la generación de API Keys, se recomienda leer el siguiente documento: [Generación de API Key en Rocketbot](#).

Orquestador Rocketbot : Como crear mi primer proceso

Paso 1: Proyecto

Un proyecto es una carpeta donde estarán los procesos automatizados. Su función es ordenar y agrupar los procesos.



The screenshot shows the 'Projects' dashboard with a search bar and a table of projects. The table has columns for Name, Description, Processes, and Created. There are five rows of data. At the bottom right, there is a green 'ADD PROJECT' button.

Name	Description	Processes	Created	
Alan	Alan	9	2022-11-25 14:56:01	...
Nicky		2	2022-12-12 16:41:51	...
capacitacion		2	2023-01-17 11:11:38	...
Academy		2	2023-03-21 11:15:34	...
BotsInternos		10	2023-03-21 13:39:54	...

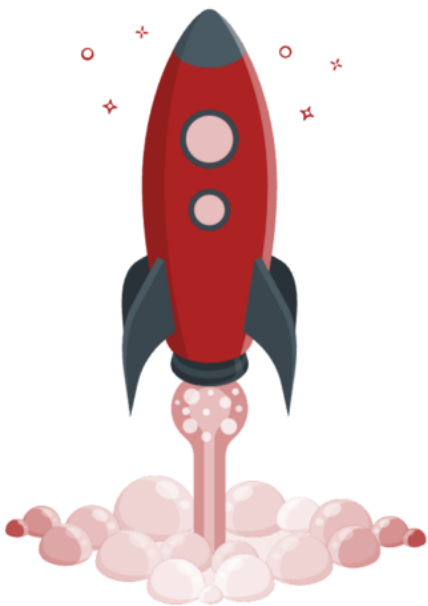
Navigation: First, Prev, 1, 2, 3, 4, Next, Last. Page 5 of 5. ADD PROJECT

Crear proyecto

El primer paso que se debe realizar es crear un proyecto o seleccionar uno existente, al cual se le asignara el nuevo proceso. En el dashboard de inicio podrá ver los proyectos existentes para seleccionar.

Si desea crear uno nuevo haga click en el botón **[Add Project]** ubicado abajo a la derecha.

New project



A project will allow you to have control of one or more robots and add instances of various types of systems so that your robots run in different architectures.

To create a project, enter the name and press create.

Name

Only alphanumeric characters and at least 3 characters long, project names cannot be repeated.

Close

Create

Asigne un nombre al proyecto que contendrá sus procesos. por ej: "QA", "Produccion", "DEV", etc.

Una vez creado, ábralo desde el la lista de proyectos para poder agregar procesos al mismo.

Paso 2: Proceso y Robot

Un proceso es donde se subirá y configurará el robot, triggers y demás puntos correspondientes a la automatización.

A screenshot of a web interface for managing processes. The header shows 'ProyectoEjemplo processes' with an 'Add process' button and a 'Total 7 / 10000' indicator. Below the header is a search bar and a table with columns: Name, Description, Has robot, Cron task, Instances, Queues, and Status. The table contains two rows: 'Ejemplo2' and 'Ejemplo3', both with 'Has robot' marked with a red 'x'. At the bottom, there are pagination controls (First, Prev, 1, Next, Last) and a dropdown menu showing '5'. A green 'ADD PROCESS' button is visible in the bottom right corner.

Name	Description	Has robot	Cron task	Instances	Queues	Status
Ejemplo2		x	0	1	0	Public
Ejemplo3		x	0	1	0	Public

Para subir un robot, se debe crear o seleccionar un proceso existente.

Crear proceso


Para crear un proceso debe hacer click en el botón **[Add Process]** ubicado abajo a la derecha.

Al nuevo proceso, se le debe ingresar:

- Nombre de proceso
- Nombre del robot principal o robot Padre que inicia el proceso
- Ruta del archivo exportado a producción desde Rocketbot Studio (extensión .db)

Add process

Process name <input type="text" value="orquestador"/> <small>Process name must be at least 3 characters long, no special characters. Only one whitespace between characters.</small>	Start robot name <input type="text" value="bot_orquestador"/>	Uplad DB <input type="button" value="Choose File"/> <input type="text" value="bot_orquestador.db"/>
---	---	---



Add a process and load a robot to run in one or more instances You can load a robot by exporting to DB production or a DB project

Una vez creado, se puede seleccionar desde la lista de procesos para realizar las configuraciones siguientes.

Paso 3: Instancia donde corre el robot


En caso de querer asignar una nueva instancia al proceso, primero debemos agregarla y luego vincularla, por defecto no estará conectada.

ProyectoEjemplo / Ejemplo3

Process Robot Backup Alerts Cronjobs Launch Bot Xperience Data load

All instances [Add or link instance](#) Total 0 / 10000

Available instances:
85 / 100
 You need to connect at least one instance to run this process [CONNECT INSTANCE](#)




[ADD OR LINK INSTANCE](#)

[¿Cómo vincular/crear instancia y conectarla?](#)

Orquestador Rocketbot – Xperience form editor

Aquí se crean y editan los formularios de Xperience.

Form settings (propiedades del formulario)

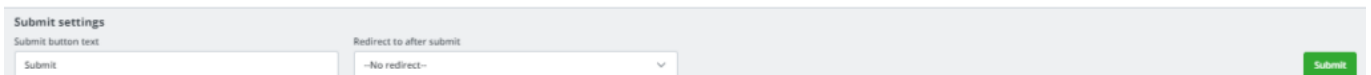
Form settings	
Form name: formExample	Public <input type="checkbox"/> Enabled <input checked="" type="checkbox"/> reCaptcha <input type="checkbox"/> send API <input type="checkbox"/> Form token: JCQF5GGF14FB85IV 
Form name	Permite editar el nombre del formulario. Debe contener más de dos caracteres, 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



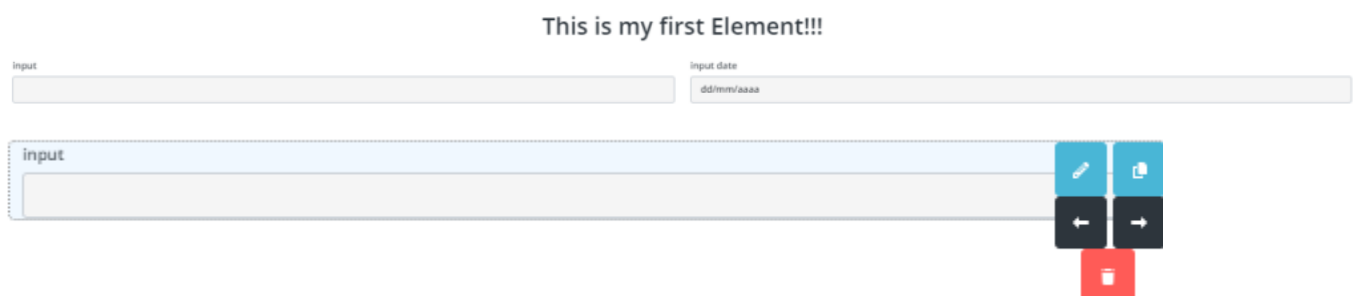
- Add element** Abre el modal de creación de elemento (ver sección de crear/editar elemento).
- Save** Guarda el formulario. Si surge algún error durante el guardado, se mostrará un mensaje de error y el formulario no podrá ser guardado.

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--	<input type="text"/>	--Select col class--
Element title		
<input type="text"/>		
Element style		
<input type="text"/>		
Element required	Css class	
No	<input type="text"/>	

(*) Required fields

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 (*)
<input style="width: 90%;" type="text" value="Title"/>	<input style="width: 90%;" type="text" value="12"/>	<input style="width: 90%;" type="text" value="col-md-1"/>
Element title		
<input style="width: 95%;" type="text" value="1"/>		
Element style		
<input style="width: 95%;" type="text"/>		
Element required	Css class	
<input style="width: 90%;" type="text" value="No"/>	<input style="width: 95%;" type="text"/>	

(*) Required fields

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)*
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.
- Id**
- 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)	<p>Disponible en Select. Hay dos opciones, agregar las opciones manualmente o traerlas desde una API.</p> <p>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.</p> <p>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.</p> <p>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".</p>

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	-- No function --
On change	-- No function --
On blur	-- No function --
On mouse over	-- No function --
On input	-- 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		Add
https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.30.1/moment.min.js		
<small>Library already added</small>		
#	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

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.

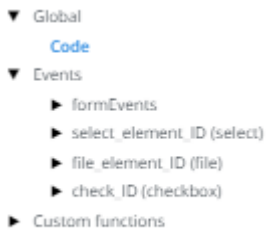
load Cuando un formulario se termina de cargar en el navegador.

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)

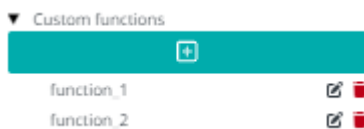


En este menú se listan todos los elementos creados del form cuyo tipo tenga eventos asociados (Input, Date, Number, Password, Textarea, Checkbox, File, Range). En el menú se listarán por su ID, con el tipo de elemento entre paréntesis.



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)



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 (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.

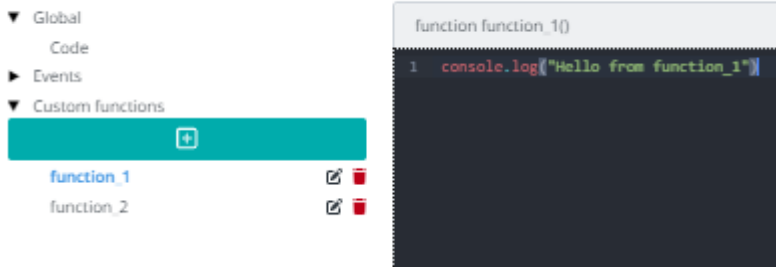
Si se intenta ingresar un parámetro inválido o repetido, aparecerá un mensaje de error y no se podrá agregar.

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")`.

Add your own Javascript code to give more functionality to your Ro



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`.

Edit element

Element type (*)

Element id (*)

Bootstrap col class (*)

File size (in bytes)

File type

Element title

Element style

Element required

Css class

(*) Required fields

Event	Custom function
On click	<input type="text" value="function_1"/>
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 --"/>

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.

▲ The code cannot be edited because a custom function has been assigned to it.

```

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.

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

URL to CDN	<input type="text" value="https://cdn.jsdelivr.net/npm/bootstrap@3.4.1/dist/css/bootstrap.min.css"/>	Add
<small>Library already added</small>		
#	Css Lib CDN	
1	https://cdn.jsdelivr.net/npm/bootstrap@3.4.1/dist/css/bootstrap.min.css	

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	<input type="text" value="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.



Presionar el botón verde a la derecha habilitará el modo de edición. En el 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.

Orquestador Rocketbot – R.O.C. Infraestructura

Administre y controle sus robots en cualquier momento con R.O.C.