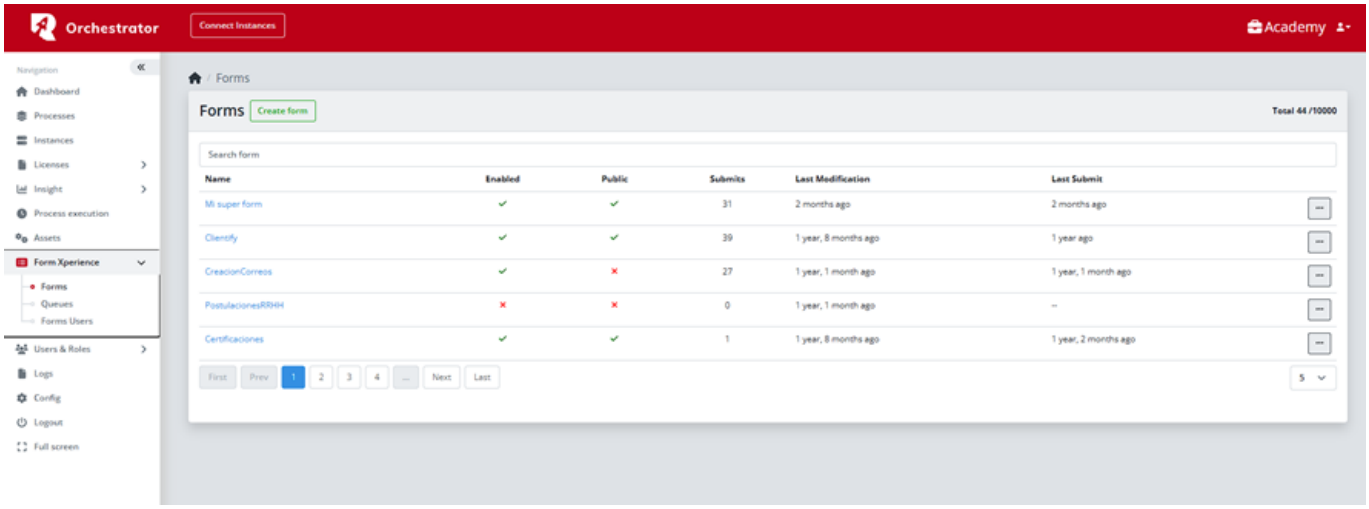
FORMS XPERIENCE

Formularios

Listado

Para ver el listado de formularios nos deberemos dirigir a la pestaña “Forms” en el grupo “forms xperience”.



- “Name”: Nombre del formulario.
- “Enabled”: Información que nos indica si el formulario está habilitado o no.
- “Public”: Información que nos indica si el formulario esta privado o público.
- “Submits”: Cantidad de queues (envíos de información) hechos.
- “Last modification”: Ultima modificación del formulario.
- “Last submit”: Ultima queue (envíos de información) hecho.

Formulario público

Permite que cualquier persona con el link pueda ver e interactuar el formulario.

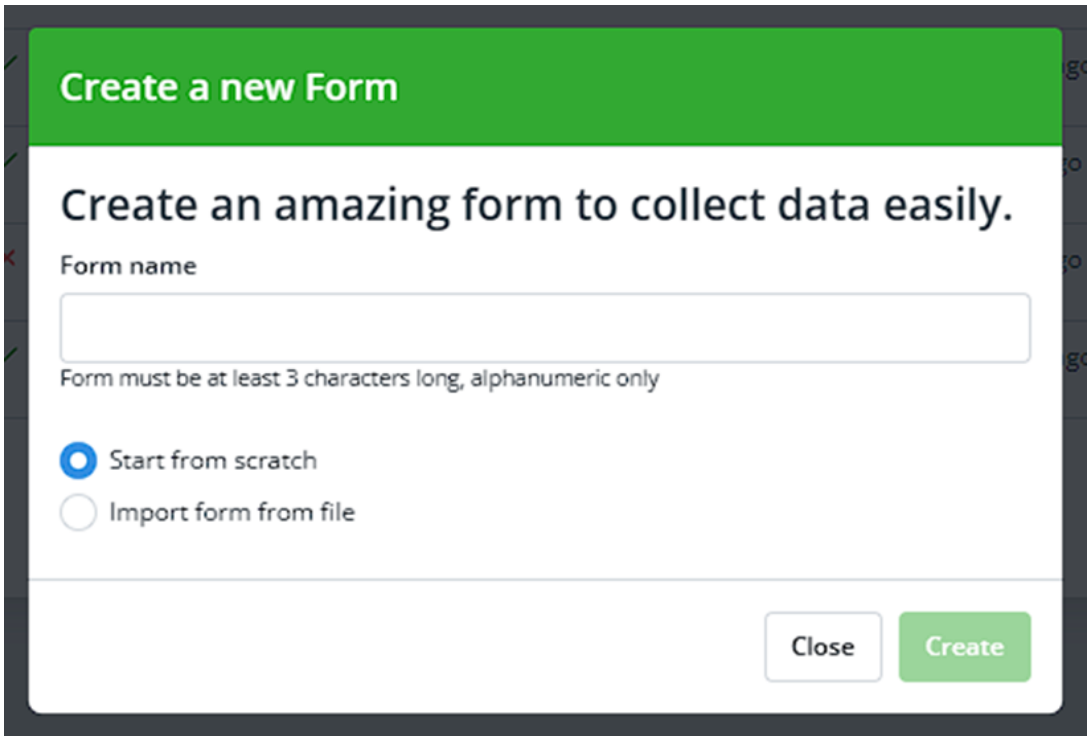
Formulario privado

Para ingresar al formulario se deberá contar con un usuario y contraseña habilitada por el dueño del formulario. (Ver “Forms Users”)

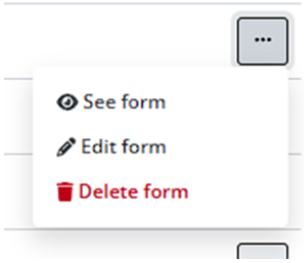
Crear formulario

Para crear un formulario deberemos dirigirnos al botón inferior derecho o el botón superior izquierdo.

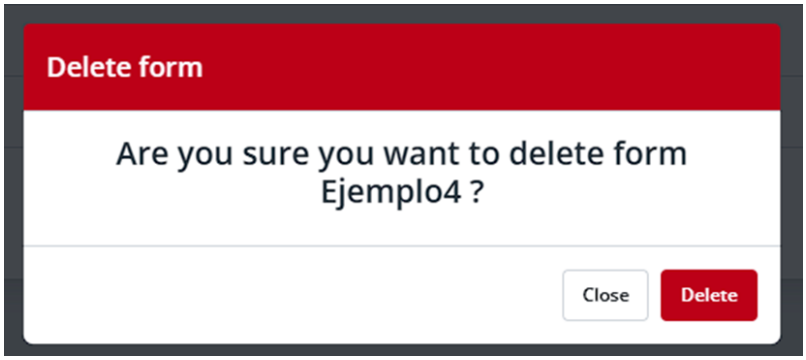
Nos abrirá un modal en donde debemos colocar el nombre del formulario. Y configurar si queremos que sea un formulario desde cero o importando un formulario desde un archivo.



Editar formulario



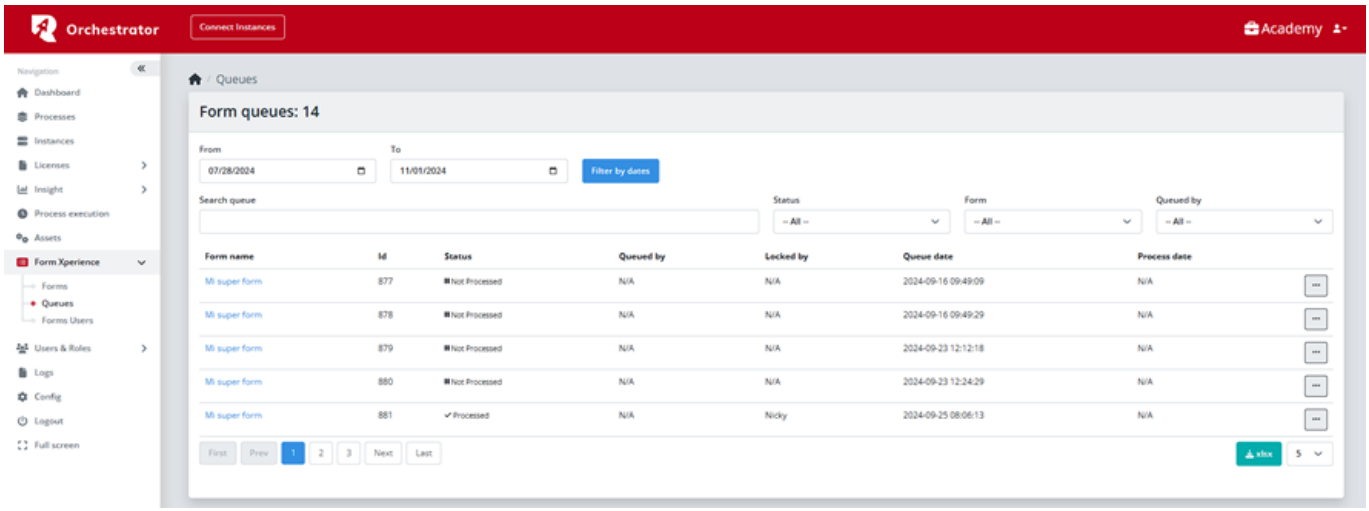
Eliminar formulario



Queues

Listado

Para ver el listado de queues nos deberemos dirigir a la pestaña “Queues” en el grupo “forms xperience”.



- **“Filter by dates”:** Filtro por fecha.
- **“xlsx”:** Exportación de archivo xlsx de lo filtrado.
- **“Status”:** Filtro por estado.
- **“Form”:** Filtro por form .
- **“Queues By”:** Filtro por usuario.
- **“Restart”:** Reinicia la ejecución .
- **“Finish”:** Finaliza la ejecución .
- **“Extra data”:** Muestra data en texto o en tabla

¿Que son las Queues y como se utilizan?

Son envíos de información donde se guardan todos los datos que se envían a través de un formulario.

Al momento de recibir datos de un envió de información de un formulario, se crea un “queue” el cual se visualizara en el listado.

Podremos obtener información de si no está procesado, se está procesando o ya ha sido procesado.

Si es un formulario privado podremos saber el email asociado a los “Forms Users” (ver) que envió el queue.

También podremos reiniciar el queque por si no se procesó correctamente o perdimos la información. O por el contrario, finalizar un queue el cual consideramos que no hace falta procesar.

USER FORMS

Listado

Para ver el listado de user forms nos deberemos dirigir a la pestaña “User Form” en el grupo “forms xperience”.

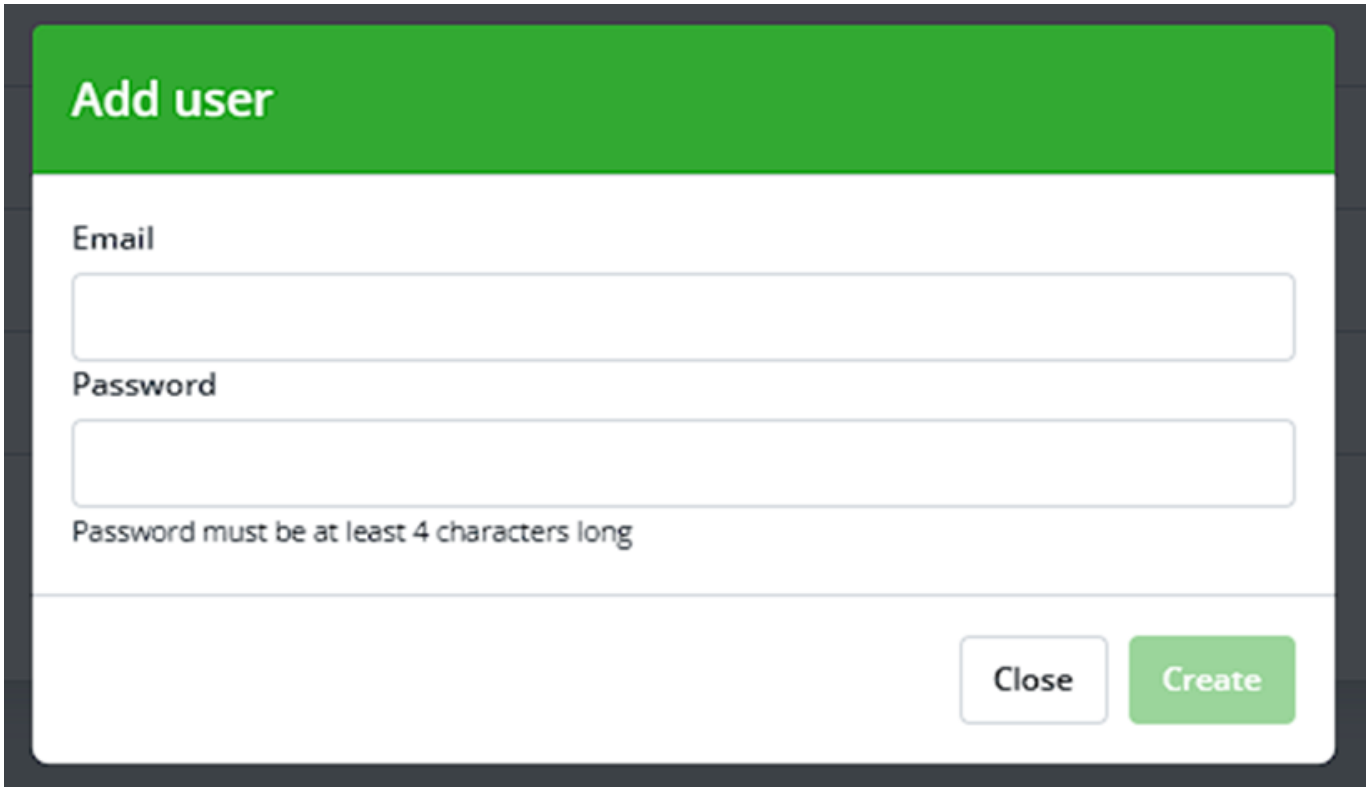


- “Email”: Email con el que vamos a inscribir al usuario.
- “Forms”: Cantidad de formularios en las que está habilitado el usuario.

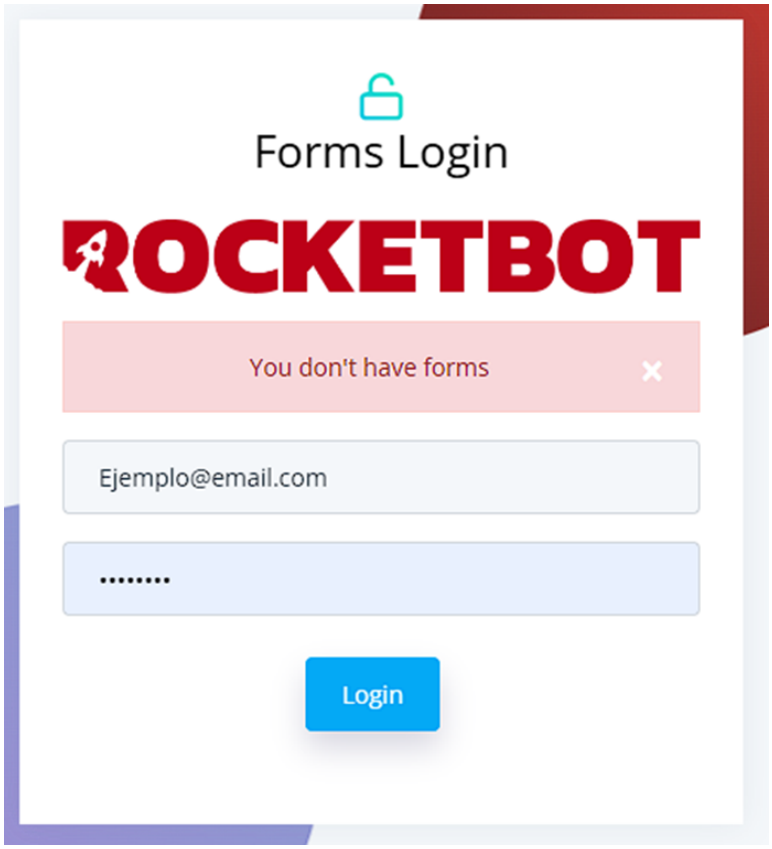
Crear user form

Para crear un form user deberemos dirigirnos al botón inferior derecho  o el botón superior izquierdo. 

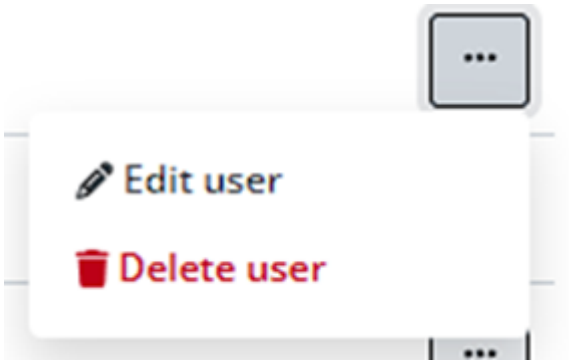
Se nos abrirá un modal donde pondremos el email de la persona que queremos vincular a formularios privados. Y una contraseña para pasarle.



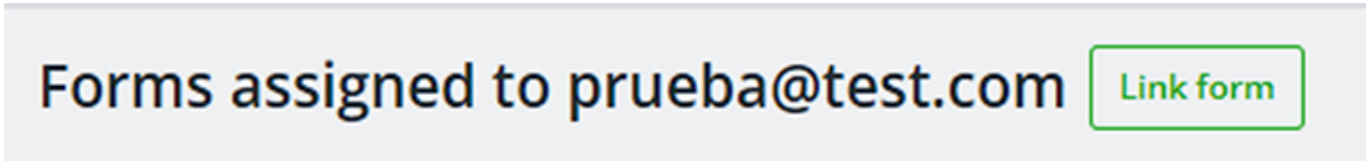
IMPORANTE: Se creará el form user con 0 formularios vinculados. Si no se agrega formularios saldrá el siguiente mensaje al querer ingresar.



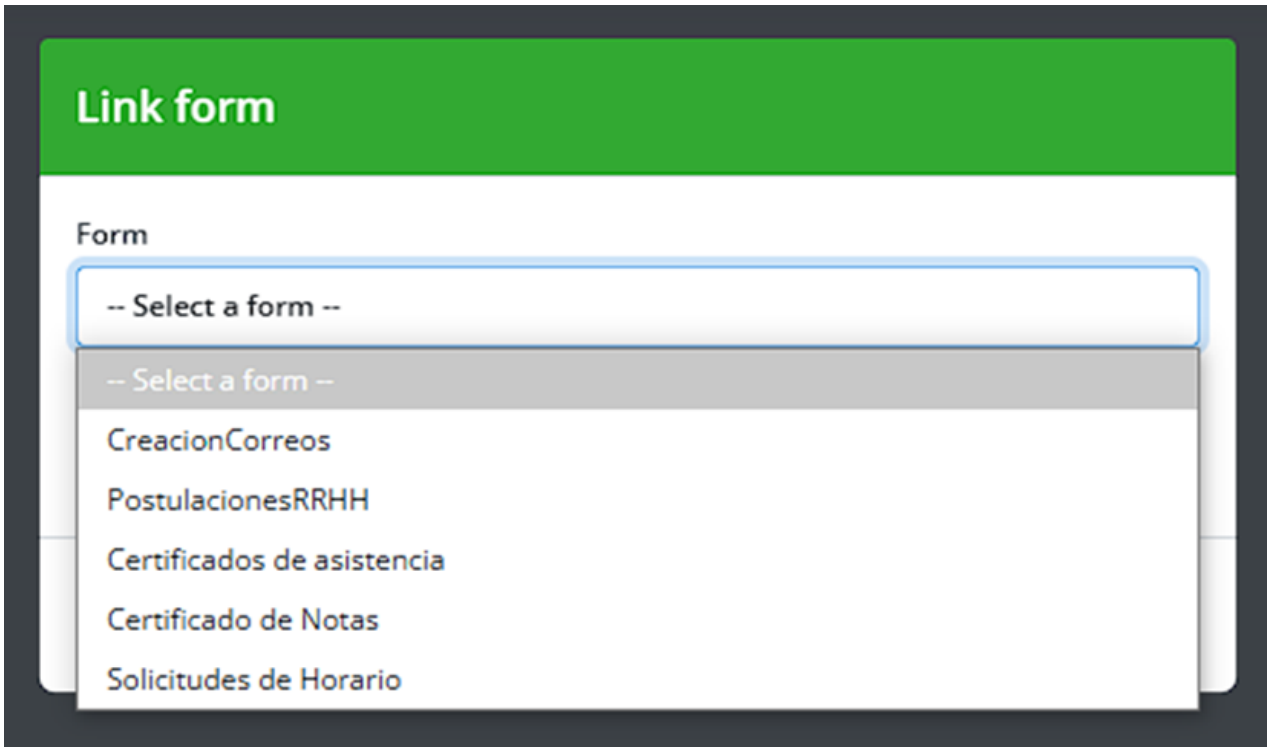
Editar user form



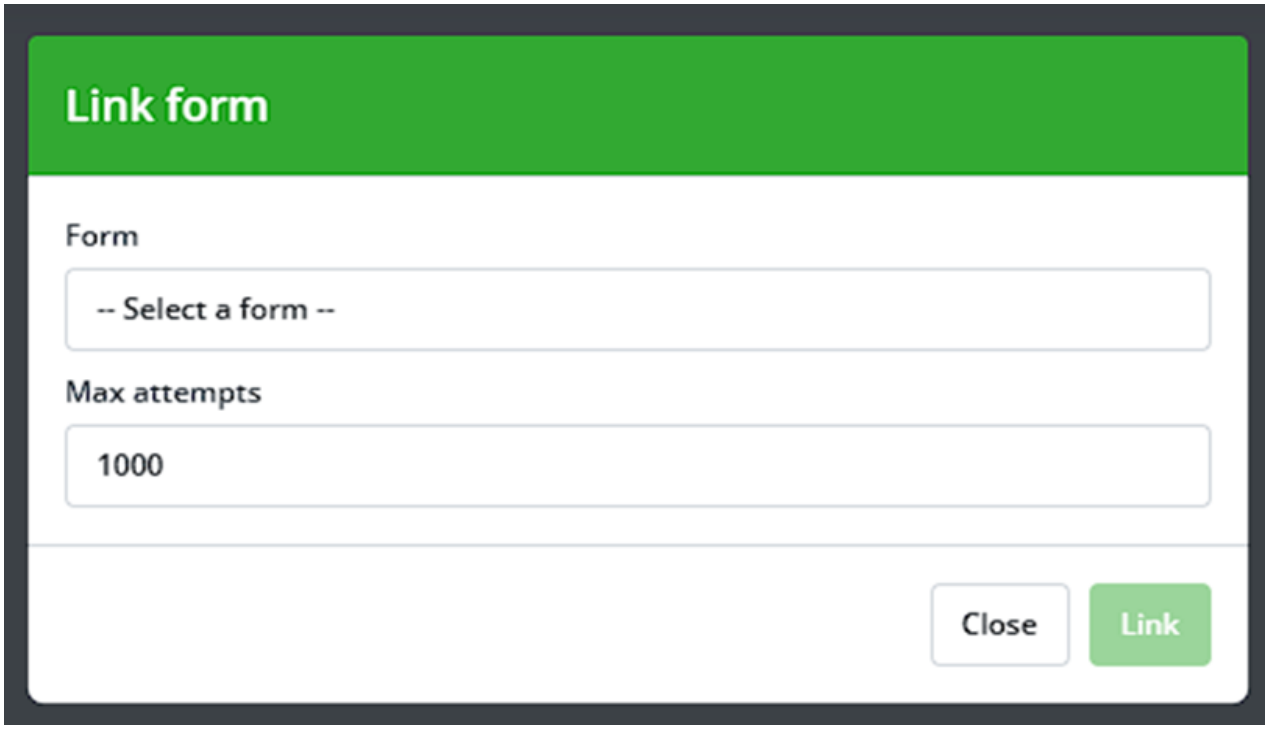
Nos abrirá la configuración del form user donde podremos vincular los formularios yendo a “link form”.



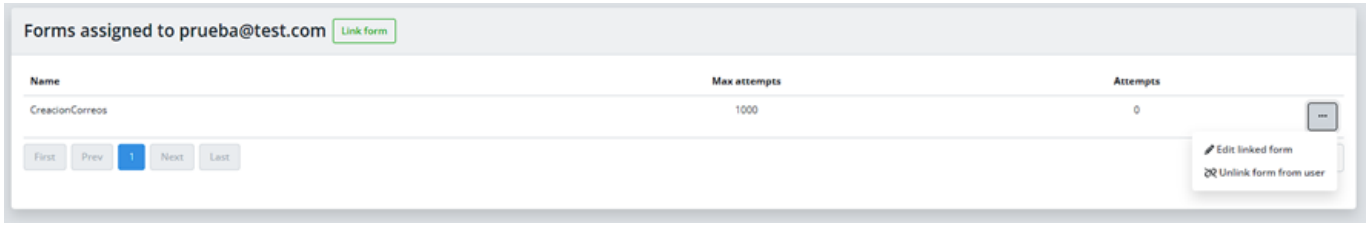
Seleccionaremos el formulario que deseamos vincular.



Max attempt: Debemos seleccionar el máximo de queues (envíos de información) que puede enviar ese user form.



Y se nos creara una vinculación con dicho formulario.



- “Name”: Nombre del formulario vinculado.
- “Max attempts”: Máximo de queues permitidos.
- “Attempts”: Cantidad actual de queues enviados.
- “Edit linked form” : Editar vinculación para cambiar de formulario o agregar queues permitidos.
- “Unlink form”: Desvincular formulario.

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: '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": "JR0EYGNSXEMTQWUP",
            "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": [
                        {
                            "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"
                        }
                    ]
                }
            ]
        }
    ]
}
```


Subir robot a un proceso existente

Permite asociar un robot a un proceso ya existente en el Orquestador.

Método	URL	Formato	Parámetros
POST	**https://www.myrb.io/dev/api/process/robot/update/data**	multipart/form-data	**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

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,
                    "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"
                }
            ]
        }
    }
}
```


Iniciar la ejecución de un robot

Permite iniciar la ejecución de un robot, sin necesidad de un formulario. Es ideal para integrar la ejecución de robots en sistemas externos.

Método	URL	Formato	Parámetros

POST	**https://www.myrb.io/dev/api/process/robots/start**	application/x-www-form-urlencoded	**Authorization**: tipo Bearer Token; es el Token de acceso válido para autenticar al usuario (API Key asociada al perfil en el orquestador). **process_id**: Identificador del proceso. Hace referencia al token del proceso, se puede obtener desde el ROC o utilizando el endpoint **/api/project/list**.

Avanzado

Ejemplos de request

- **cURL**

```
curl --location 'https://www.myrb.io/dev/api/process/robots/start' \
--header 'Authorization: Bearer <token>' \
--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
POST	**https://www.myrb.io/dev/api/formData/all**	application/json	**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)

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": "6HNZNSYVG4W2MCSO",
            "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

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

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":
"eyJpdiI6Inpid0RRVlV0bEczTW9DM1Z5SEFrV1E9PSIsInZhbHVlIjoiOVJCZHhHTDBwQ0p1Y2ps
VU9WZTYxbE5paFwvRTQ4cWlKSFZ3UUxGcVNUQXh4UTBxOFEwc01Ba0pvV1kwQklRdXByTE1JU2tWM
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
POST	**https://www.myrb.io/dev/api/formData/get/:token: :token:** Representa el token de formulario creado en ROC Xperience.	**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 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

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	**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

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
POST	**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.	**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 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":
"eyJpdiI6InNxMXhRekhVUUJrWVptcGpzOUM1aXc9PSIsInZhbHVlIjoiY1NoMGtpcW5udk5vTmZx
VlVYZFg1V29CUGU4SDB1UVUwWXFuYVNMb2FVNkE2OE5MMnRwelorWGxtMU9VZ2pjTGJnSUNLaE5cL
1ArSkpPdk1BendRYVlvTlMyU0QwclNxN0E2VENmSDEyZTJRPSIsIm1hYyI6ImYyYjc2ZmM3OTAwMW
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**	application/x-www-form-urlencoded	**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"
    }
}
```

Rocketbot Xperience – Funciones y eventos

nota: este video será subido a youtube, con una intro y subtítulos

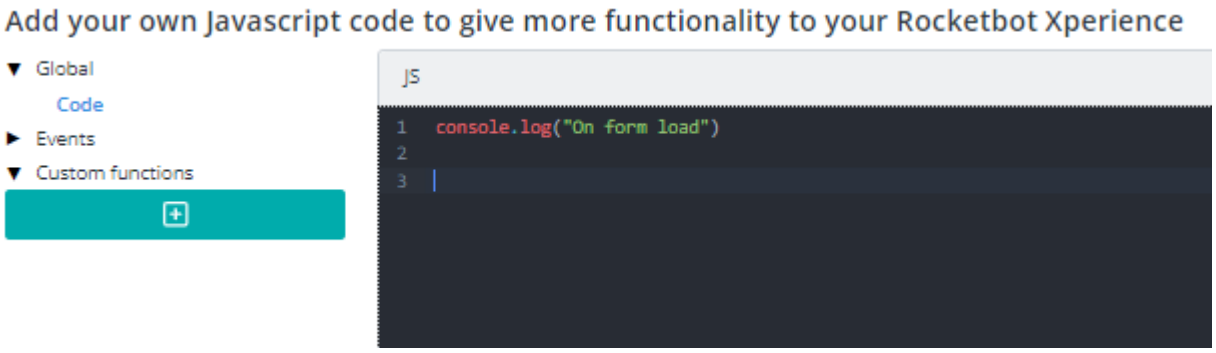
---

En cada form de Xperience, se podrán crear funciones y asignarlas a eventos

de input y de form.

Código global

Se encuentra en la pestaña JAVASCRIPT. Código que afecta a todo el form y se ejecuta cuando se carga. Es retroactivo con el código Javascript de versiones de Orquestador anteriores.



Eventos

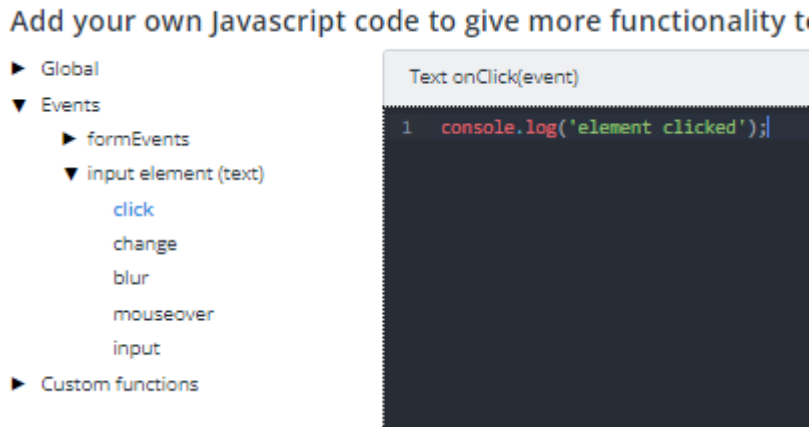
Evento	Descripción	Parámetros
load	Se ejecuta cuando se carga el formulario.	evento
submit	Cuando se completa y se envía la información del formulario.	evento
data_received	Cuando se recibe información extra mediante api.	xperience, data

Eventos de elemento

Sólo los elementos de form del grupo de Inputs y los elementos Select tendrán asociados eventos.

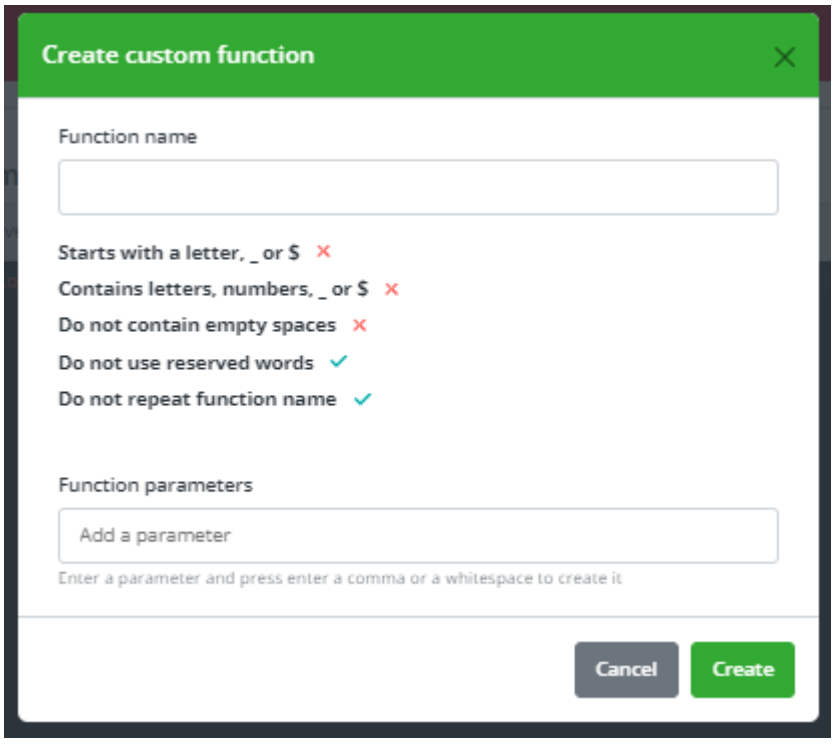
Evento	Descripción	Parámetros
On click	Se ejecuta cuando se clickea el elemento	evento
On change	Se ejecuta cuando el valor del input cambia y pierde el foco	evento
On blur	Se ejecuta cuando un elemento previamente enfocado pierde el foco	evento
On mouse over	Se ejecuta cuando se pasa el mouse por encima del elemento	evento
On input	Se ejecuta inmediatamente después de que el valor del input cambia	evento

En la pestaña JAVASCRIPT aparecerán en la lista todos los elementos creado que posean eventos. Se le podrá asignar código personalizado a cada evento de cada elemento.



Funciones custom

El usuario podrá crear sus propias funciones, código reutilizable. Se encuentra en la pestaña JAVASCRIPT, bajo “custom functions”.



Las funciones requieren de un nombre.

- Tiene que comenzar con una letra, _ o $. Puede contener números, pero no como primer carácter.
- No tiene que tener espacios vacíos entre caracteres.
- No puede tener de nombre palabras reservadas.
- No puede haber dos funciones con el mismo nombre.

No es obligatorio, pero cada función puede tener uno o más parámetros. Cada parámetro debe cumplir las mismas condiciones que el nombre de función (empezar en letra, _, $; sin espacios vacíos, sin palabras reservadas, no puede haber dos parámetros repetidos en la misma función).



Se puede editar el código que pertenecerá exclusivamente a la función.

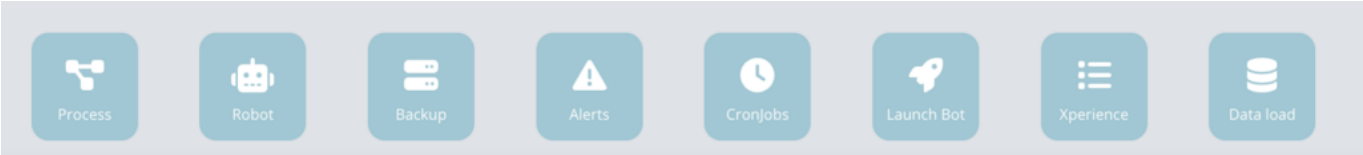
[redacted] Tener cuenta que el código de las funciones no se ejecutará a menos que sean llamadas en el código Global, en los eventos de form, o en los eventos de algún elemento.

Orquestador Rocketbot : Xperience

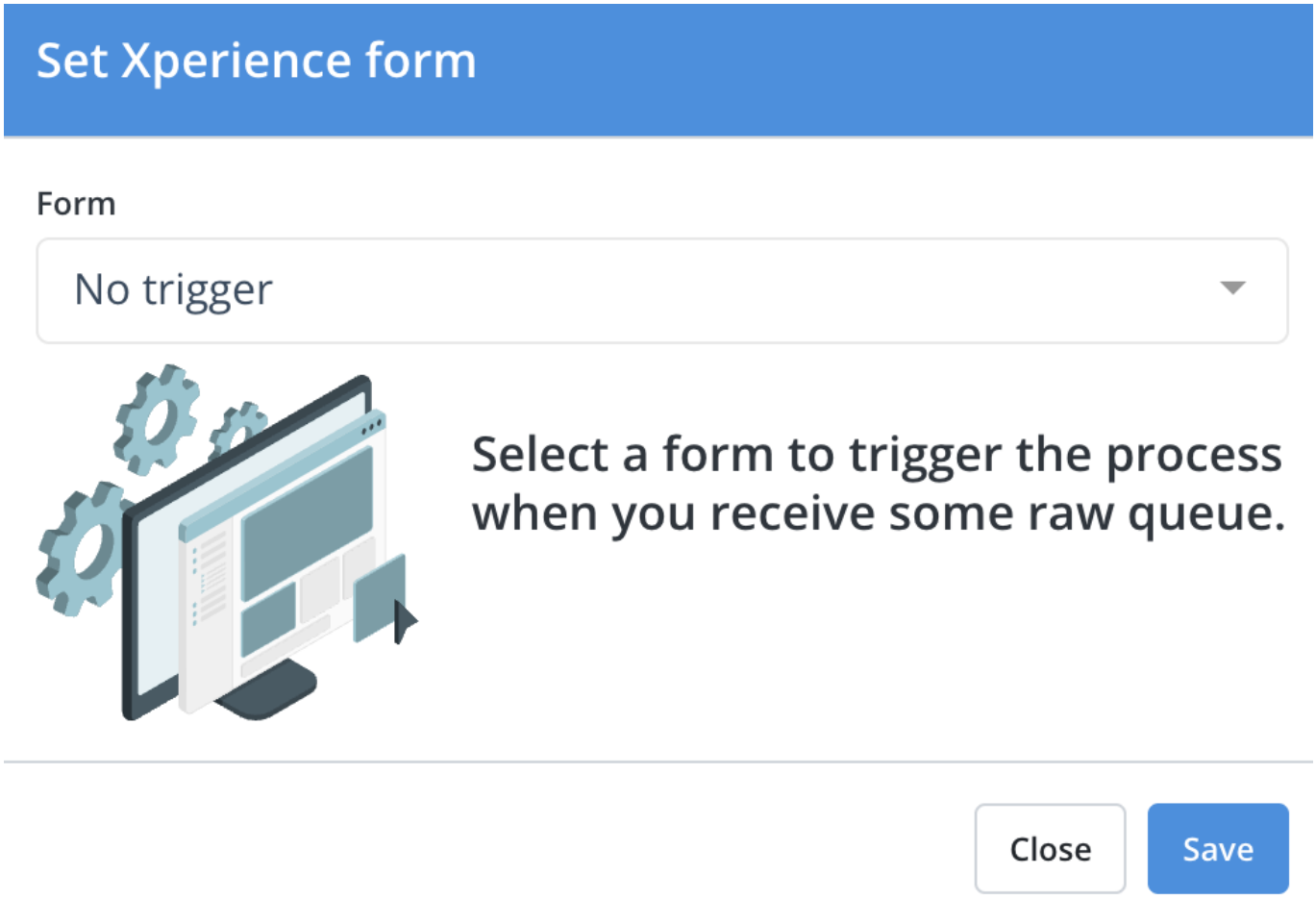
En el orquestador podemos usar los formularios de Xperience, podremos crear formularios, saber de su creador y cuando se creó .

Xperience en procesos

En la creacion de procesos encontraremos ya en su interfaz el siguiente menú:

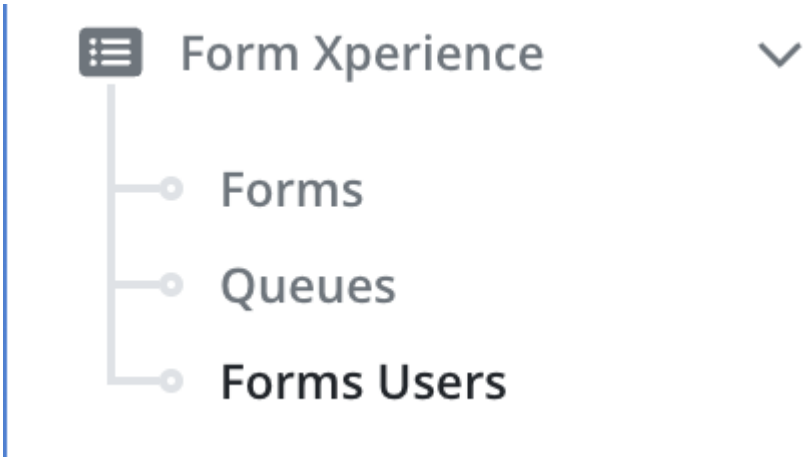


En el apartado Xperience podremos seleccionar el formulario para que lance (trigger) la ejecución del proceso cuando se realice un envió del mismo.



Para revisar los formularios de Xperience, debemos seleccionar en el menú lateral *Form Xperience*.

En la barra lateral nos aparecerán las tres opciones:



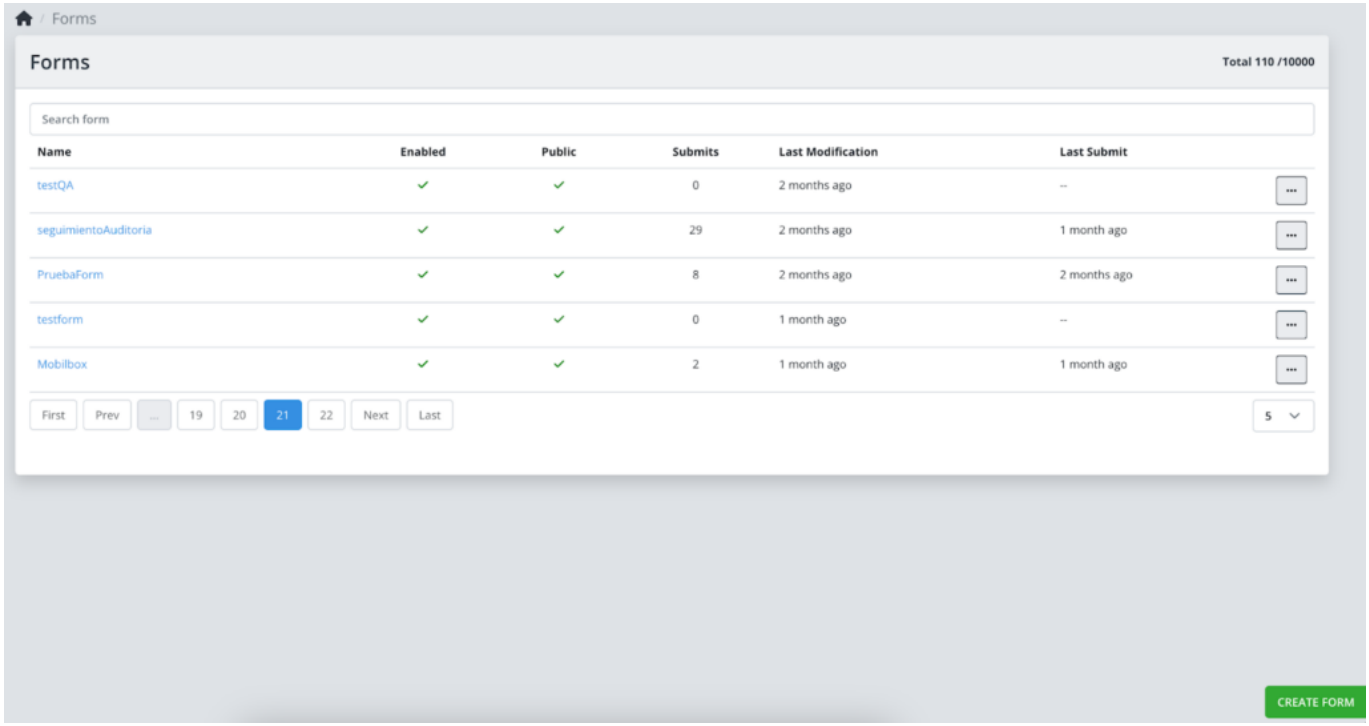
Forms Xperience :

Forms :

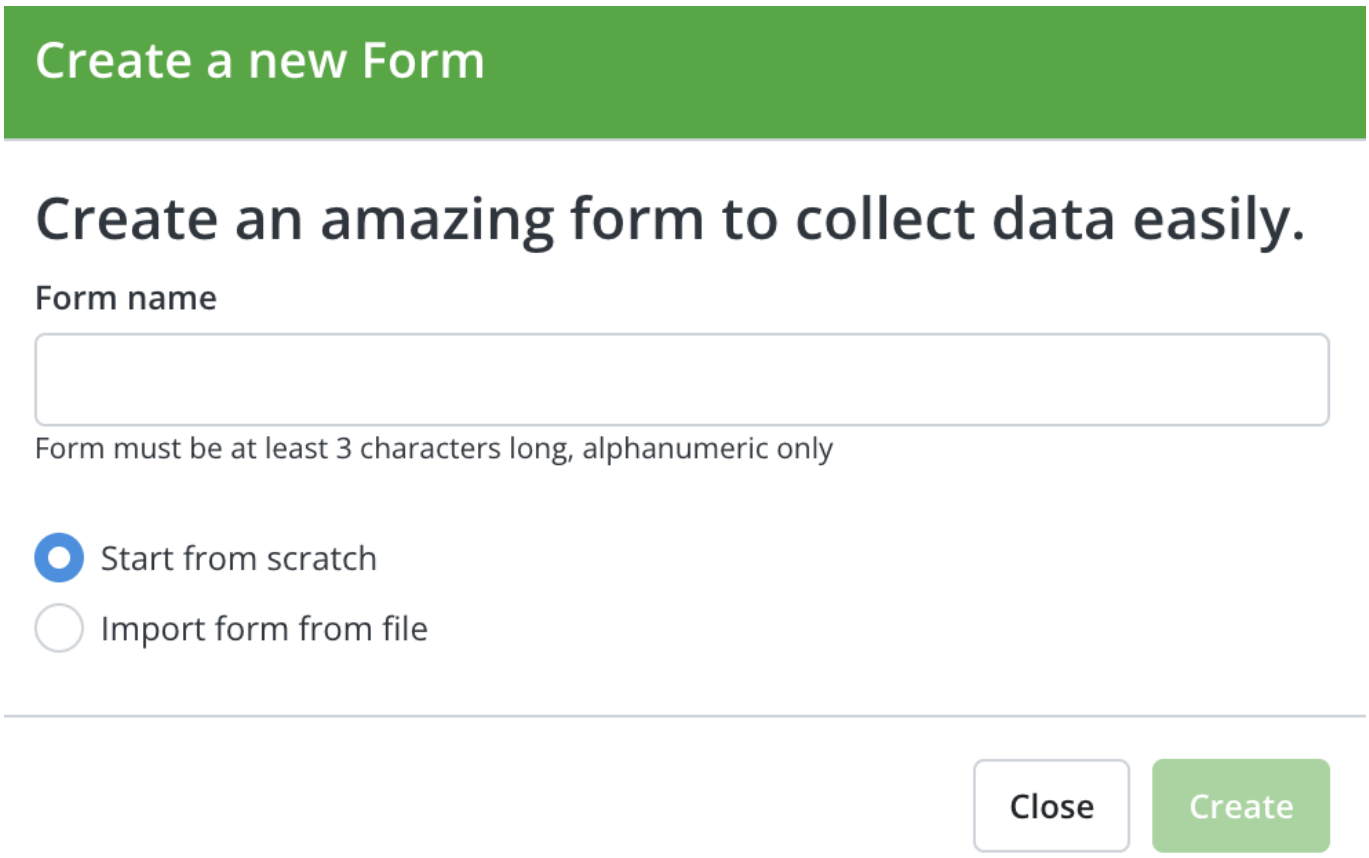
Aquí podemos crear formularios de Xperience y también observar sus detalles en el orquestador, como su nombre, estatus, si es público, los envíos (*submits*), la fecha de la última modificación y el último envío (*last*

submit).

Al lado, en los tres puntos, encontramos el menú donde podemos ver el formulario, editarlo o borrarlo.



Si deseamos crear un formulario, hacemos clic en [Create Form]. Se nos pedirá el nombre del formulario y tendremos dos opciones: comenzar con el formulario desde cero o importarlo desde un archivo.

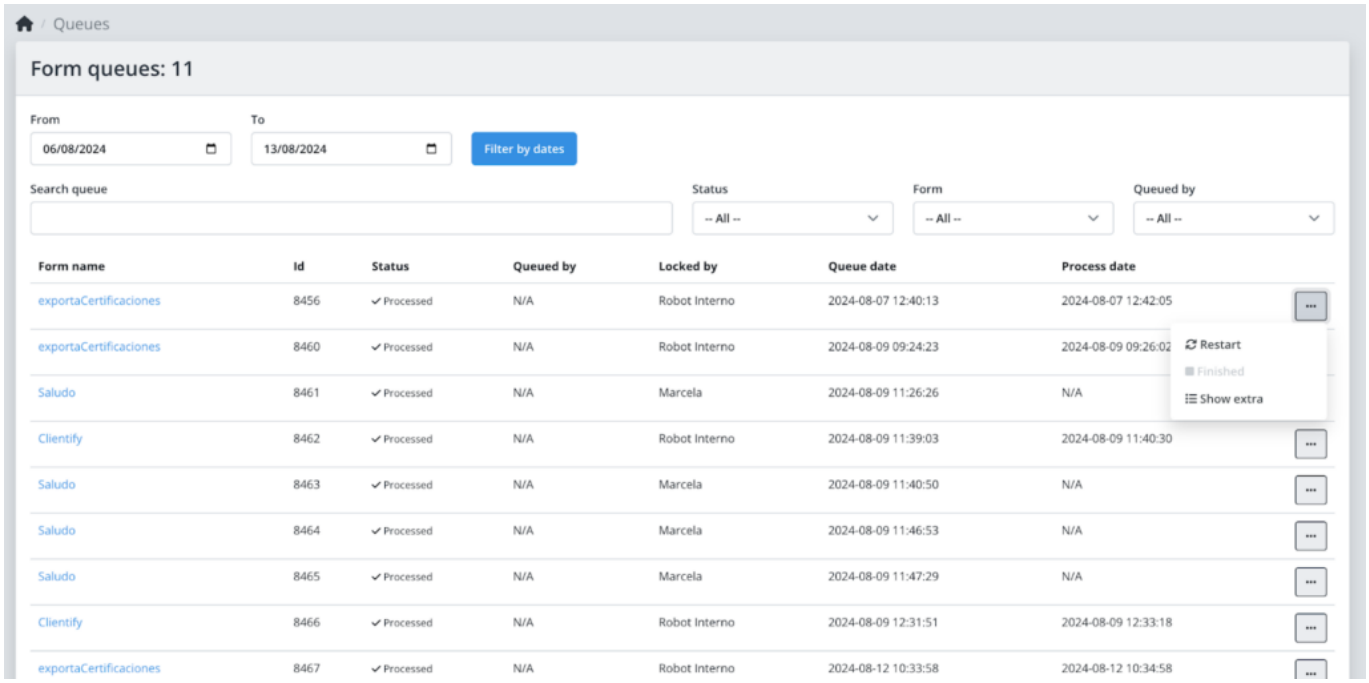


Para la creación de formularios o para importarlos, se recomienda ver el siguiente curso de Xperience: Curso de Xperience.

Form Queues :

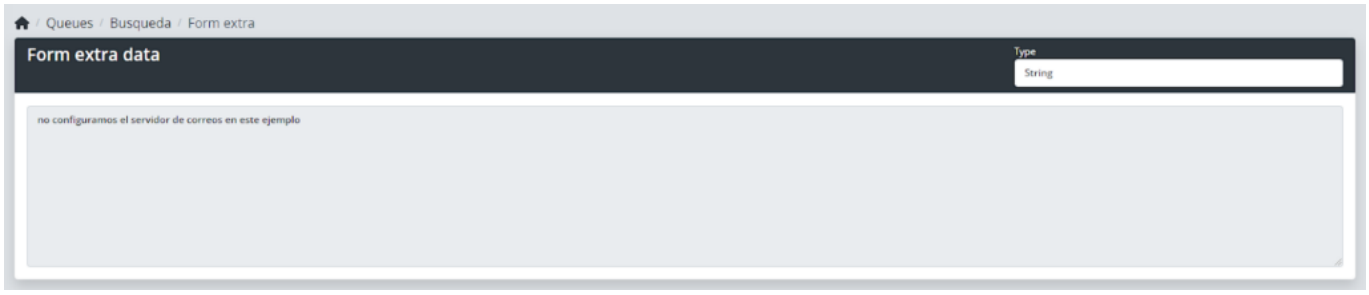
Aquí podemos observar todas las *form queues*, utilizando un filtro para buscarlas. Abajo, veremos el nombre del formulario, su ID, el estatus, *queue by*, *locked by*, la fecha de la *queue* y la fecha del proceso.

En el menú lateral nos da la opción restart , finalizar y mostrar más o extra .



En “show extra” podremos visualizar la información que se envía a través de xperience.

Con el modulo “Rocketbot Xperience” podremos utilizar comandos que interactúen con los formularios xperience, por ejemplo con el comando “Return Message to Xperience” que podremos enviar un mensaje y poder visualizarlo en la sección “show extra”:



Procesamiento de las Queues (Xperience):

Estas son las entradas de datos realizadas desde los formularios. Si un proceso tiene configurado un formulario, el orquestador realiza un chequeo cada un minuto, si hay datos sin procesar de ese formulario; en caso de

encontrar datos (una queue pendiente), si instancias disponibles, se ejecutará el robot de ese proceso en las instancias asignadas disponibles. Si todas están ocupadas, el orquestador repetirá la comprobación al minuto siguiente.

Form Users :

Aquí se encuentran los usuarios y sus correos electrónicos, junto con la cantidad de formularios que poseen. También podemos modificar a los usuarios.

Orquestador Rocketbot – Xperience form editor

Aquí se crean y editan los formularios de Xperience.

Form settings (propiedades del formulario)



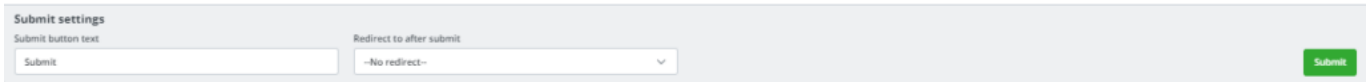
Form name	Permite editar el nombre del formulario. Debe contener más de dos caracteres, solo caracteres alfanuméricos y espacios, y no debe estar compuesto solo por espacios. No pueden existir dos formularios con el mismo nombre.
Public	Si está marcado como público, cualquiera que tenga el enlace del 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



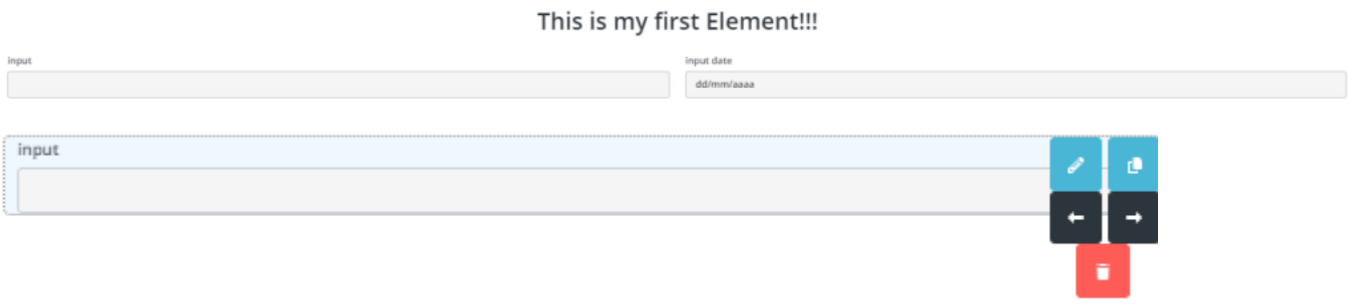
Add element	Abre el modal de creación de elemento (ver sección de crear/editar elemento).
Save	Guarda el formulario. Si surge algún error durante el guardado, se mostrará un mensaje de error y el formulario no podrá ser guardado.

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)*.



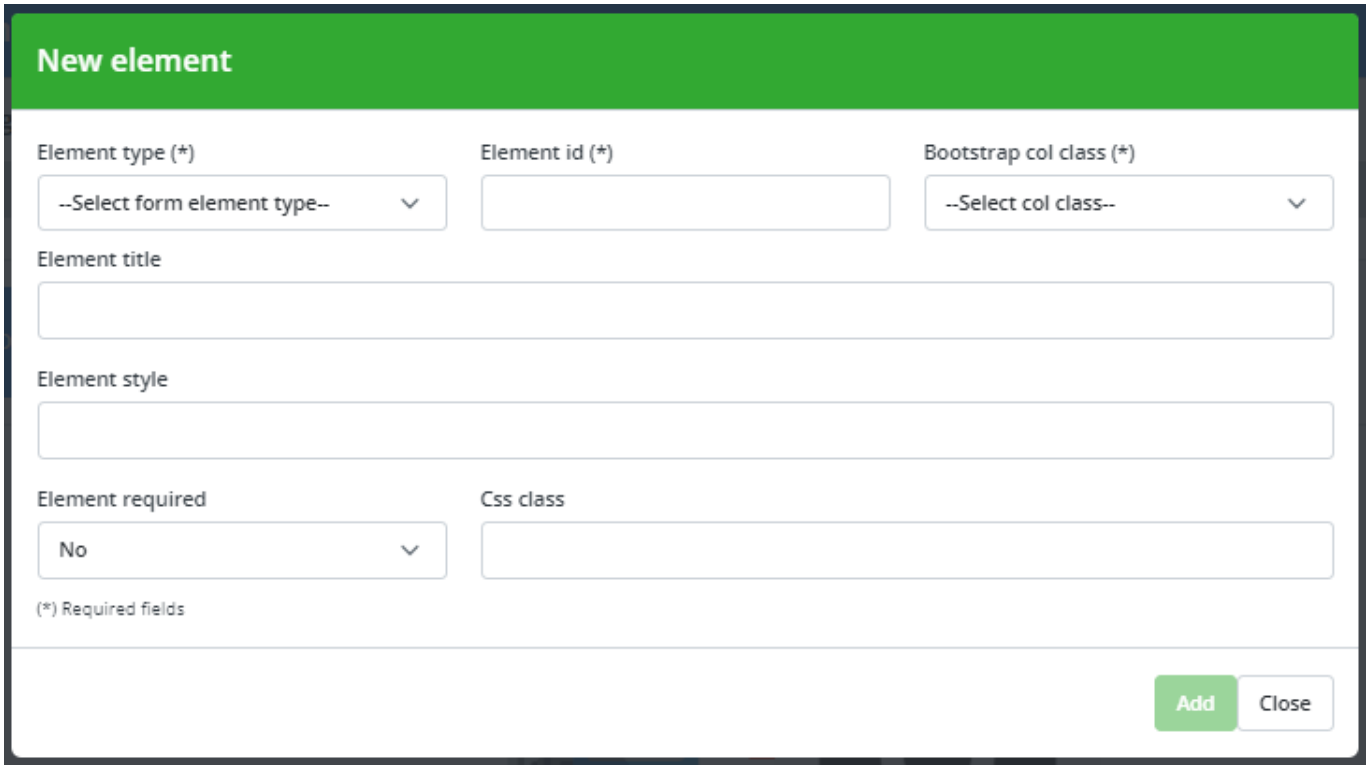
Clonar elemento: Al hacer clic en el ícono de copiar en el menú de opciones, se clonará el elemento *(ver sección de clonar elemento)*.



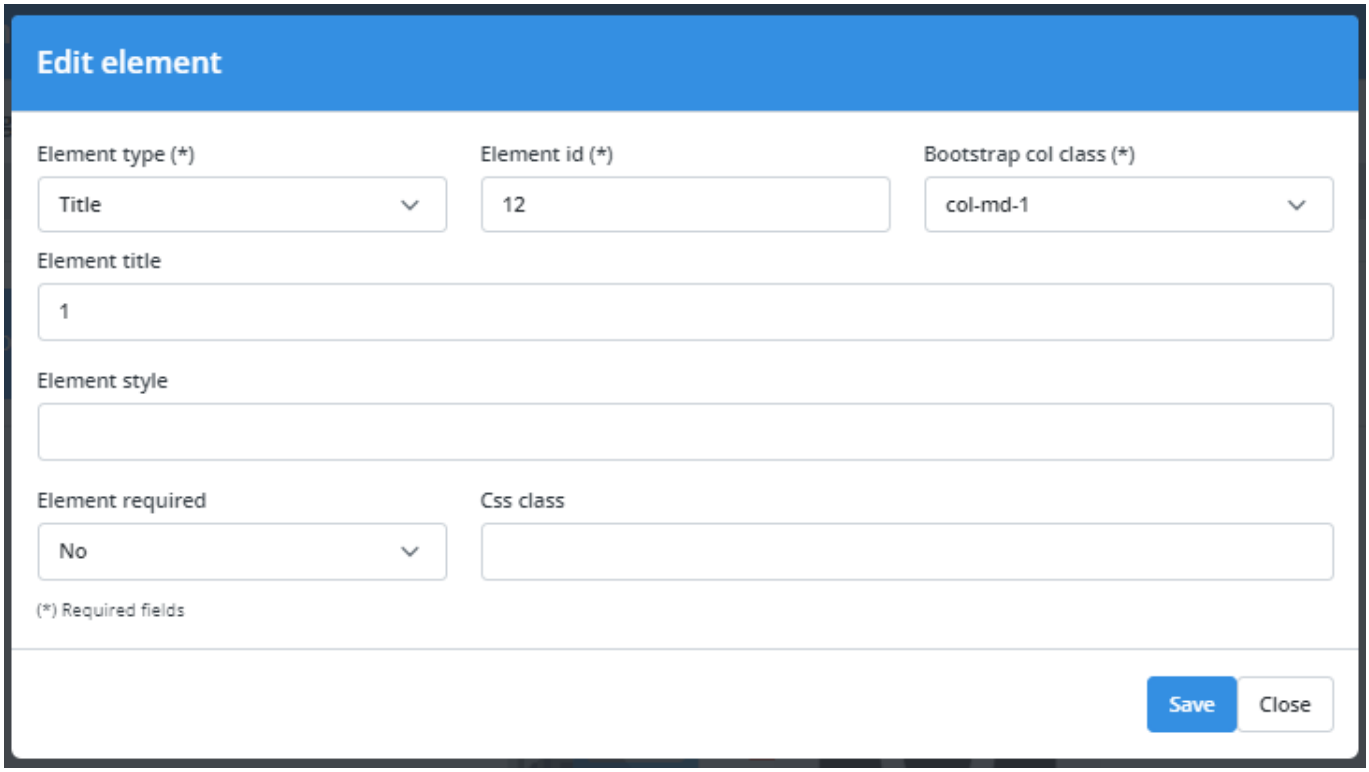
Borrar elemento: Al hacer clic en el botón rojo con el ícono de cesto de basura en el menú de opciones, el elemento será eliminado.

Crear / Editar formulario

Los formularios se crearán/editarán mediante un modal.



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.



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 tenia 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 manualemnte un id en todos los elemento 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

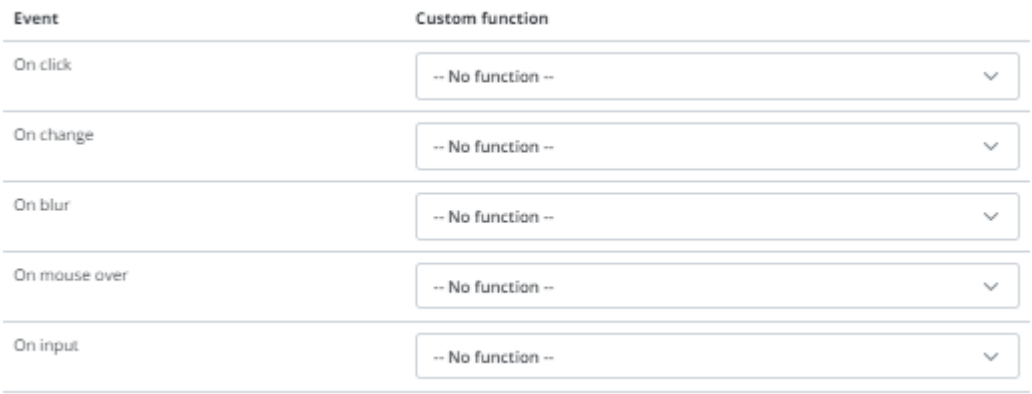
Element title	La etiqueta del elemento. Disponible en todos menos en Paragraph.

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 tendra un botón Show Events. Clickearlo expandirá un grupo de selectores, volverlo a clickear 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.



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



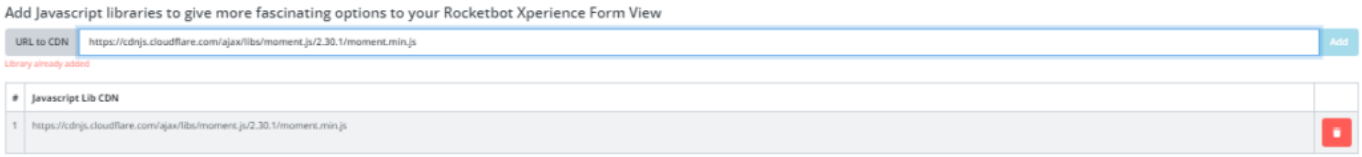
Aquí se puede customizar la experiencia del usuario del formulario agregándole código Javascript.

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.

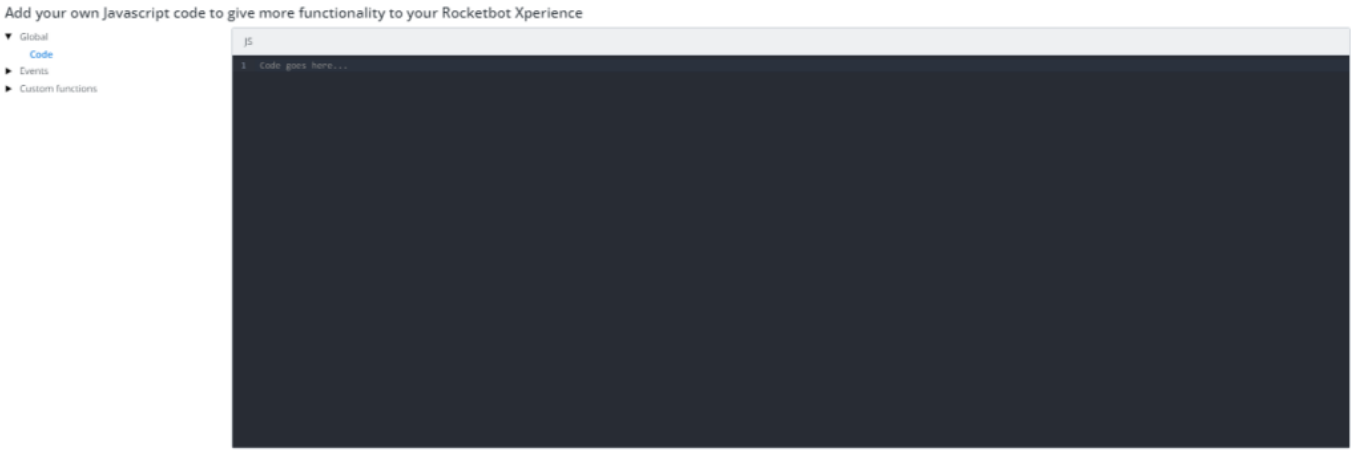
No se puede agregar la misma dirección dos veces. Si se lo intenta aparecerá un mensaje de error y no se permitirá agregar la librería.



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.



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 clickear *Global* para expandirlo y luego clickear 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.

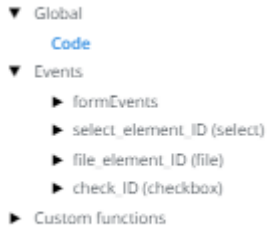
Form events (Eventos del formulario):

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 clickear 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)

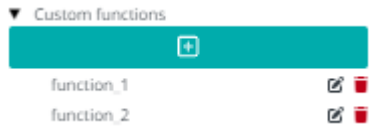


En este menú se listan todos los elementos creados del form cuyo tipo tenga eventos asociados (Input, Date, Number, Password, Textarea, Checkbox, File, Range). En el menú se listarán por su ID, con el tipo de elemento entre paréntesis.



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)



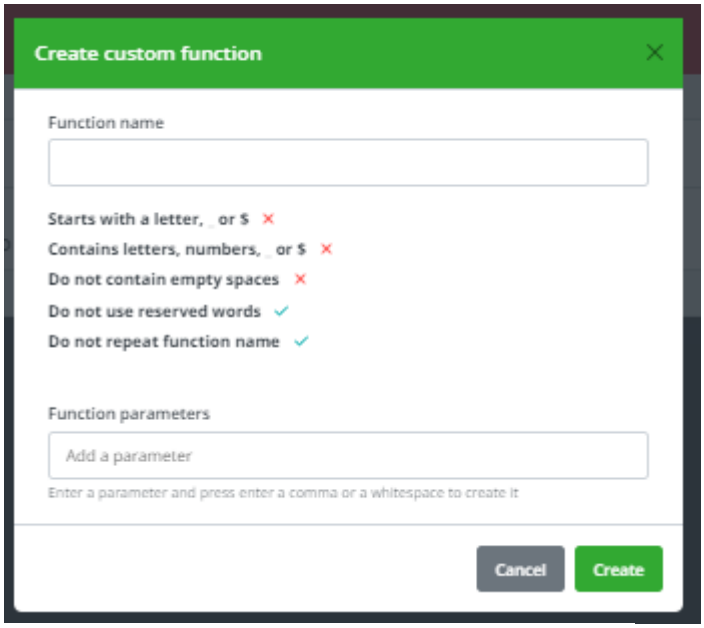
Aquí se listarán por nombre todas las funciones creadas. Clickeando el botón de “+” al principio de la lista abrirá el modal de creación de función.

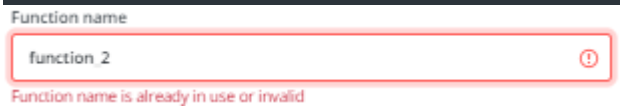
Cada función posee un nombre, ningún o varios parámetros y su código. Las funciones no pueden tener nombres repetidos y éstas y los nombres de parámetros deben seguir las reglas de nomenclatura de Javascript.

- 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.



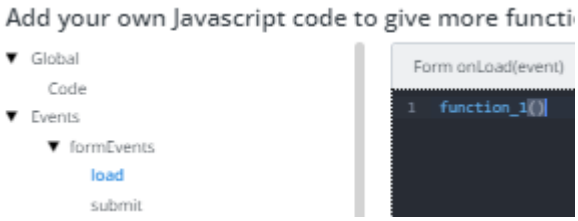
Asignación y uso de custom functions

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").



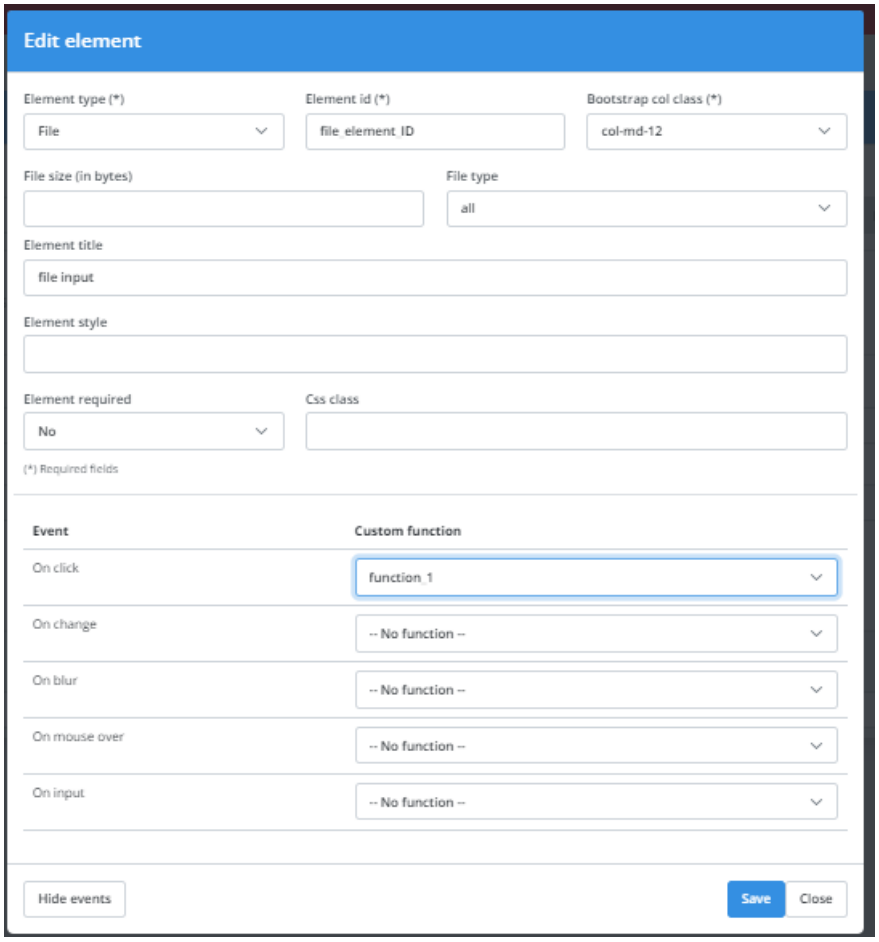
Se va hasta Events -> formEvents -> load y en su editor se ingresa function_1().



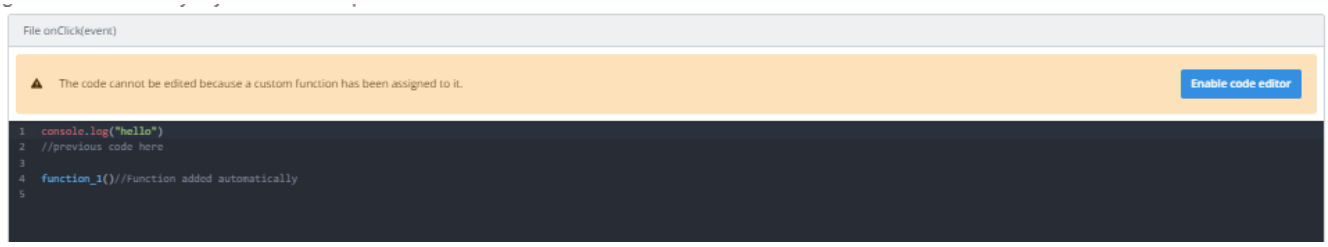
Cuando un usuario abra un formulario, cada vez que este cargue, se llamará a la función function_1 y se ejecutará el código asociado a esta, en este caso un console.log.

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.



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.



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)



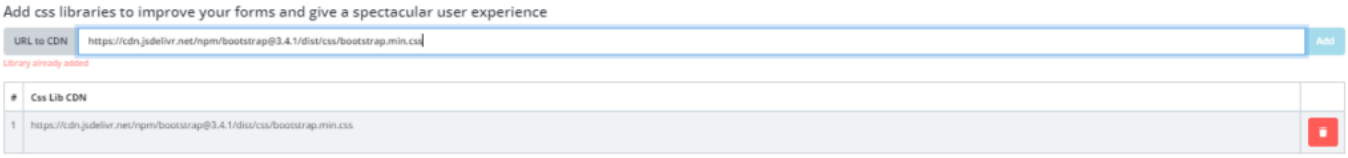
Aquí se puede customizar el estilo y la experiencia de usuario del formulario agregándole estilos de CSS.

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.

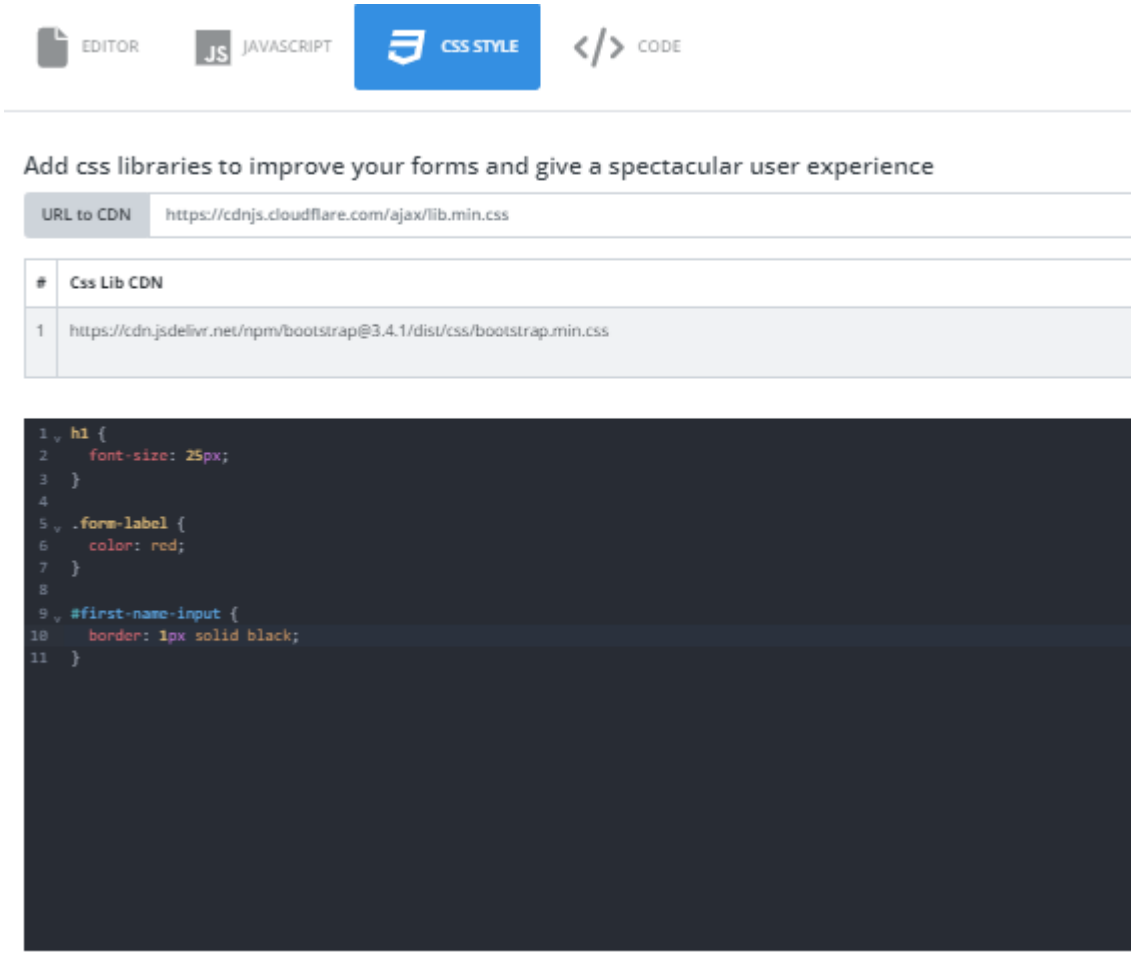
No se puede agregar la misma dirección dos veces. Si se lo intenta aparecerá un mensaje de error y no se permitirá agregar la librería.



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.



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 clickear 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.

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'
      });
 }

});
```