

Memoria Práctica 2

Jaime Parra Jiménez

INTEGRACIÓN DE TECNOLOGÍAS Y SERVICIOS INFORMÁTICOS

Universidad de Almería

Correo: jpj451@inlumine.ual.es

5 de noviembre de 2025

Índice general

1. Ejercicio Guiado 1	2
2. Ejercicio 1	5
3. Ejercicio 2	7
4. Ejercicio 3	11

Capítulo 1

Ejercicio Guiado 1

El objetivo del ejercicio guiado fue construir un flujo de trabajo que se ejecuta cada minuto, obtenga una lista de cinco chistes y los clasifique si son seguros o no. Para comenzar el flujo de trabajo es necesario añadir un nodo inicial Schedule Trigger. Este nodo se ha configurado para ejecutar el nodo automáticamente cada minuto, como se muestra a continuación.

The screenshot shows a workflow editor interface with two main sections. On the left, a 'Schedule Trigger' node is being configured. It has tabs for 'Parameters' (selected), 'Settings', and 'Docs'. Under 'Parameters', it says: 'This workflow will run on the schedule you define here once you **activate** it.' and 'For testing, you can also trigger it manually: by going back to the canvas and clicking 'execute workflow''. The 'Trigger Rules' section shows 'Trigger Interval' set to 'Minutes' and 'Minutes Between Triggers' set to '1'. A 'Add Rule' button is available. On the right, the 'OUTPUT' tab is selected, showing a JSON representation of the scheduled execution details:

```
[{"timestamp": "2025-10-29T11:17:13.308-04:00", "Readable date": "October 29th 2025, 11:17:13 am", "Readable time": "11:17:13 am", "Day of week": "Wednesday", "Year": "2025", "Month": "October", "Day of month": "29", "Hour": "11", "Minute": "17", "Second": "13", "Timezone": "America/New_York (UTC-04:00)"}]
```

Una vez inicializado el nodo, hay que añadir un nodo HTTP Request para obtener los chistes desde la app JokeAPI. Para ello utilizaremos la URL dada.

The screenshot shows the MuleSoft Anypoint Studio interface with an **HTTP Request** node selected. The node configuration includes:

- Method:** GET
- URL:** <https://v2.jokeapi.dev/joke/Programming?type=single&amount=5>
- Authentication:** None
- Send Query Parameters:** Off
- Send Headers:** Off
- Send Body:** Off
- Options:** No properties

The output pane displays the JSON response from the API, which contains two jokes:

```
{
  "error": false,
  "amount": 5,
  "jokes": [
    {
      "category": "Programming",
      "type": "single",
      "joke": "Eight bytes walk into a bar.\nThe bartender asks, \"Can I get you anything?\"\n\"Yeah,\" reply the bytes.\n\"Make us a double.\"",
      "flags": {
        "nsfw": false,
        "religious": false,
        "political": false,
        "racist": false,
        "sexist": false,
        "explicit": false
      },
      "id": 34,
      "safe": true,
      "lang": "en"
    },
    {
      "category": "Programming",
      "type": "single",
      "joke": "\nHoney, go to the store and buy some eggs.\n\"OK.\n\"Oh and while you're there, get some milk.\n\"He never returned.",
      "flags": {
        "nsfw": false,
        "religious": false,
        "political": false,
        "racist": false,
        "sexist": false,
        "explicit": false
      }
    }
  ]
}
```

El siguiente paso fue añadir el nodo Split Out para poder dividir el array jokes en elementos individuales, para así poder procesar cada chiste por separado.

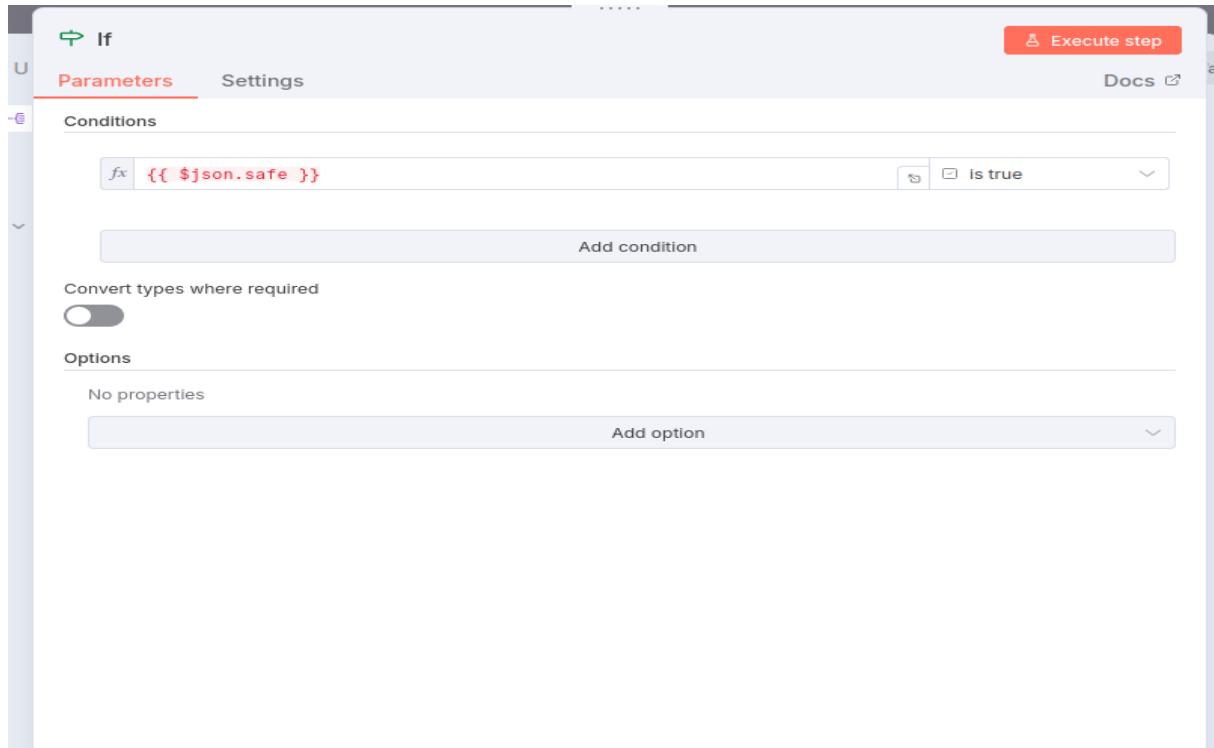
The screenshot shows the MuleSoft Anypoint Studio interface with a **Split Out** node selected. The node configuration includes:

- Fields To Split Out:** jokes
- Include:** All Other Fields
- Options:** No properties

The output pane displays the split JSON response, showing two separate objects corresponding to each joke in the original array:

```
[
  {
    "error": false,
    "amount": 5,
    "jokes": [
      {
        "category": "Programming",
        "type": "single",
        "joke": "Eight bytes walk into a bar.\nThe bartender asks, \"Can I get you anything?\"\n\"Yeah,\" reply the bytes.\n\"Make us a double.\"",
        "flags": {
          "nsfw": false,
          "religious": false,
          "political": false,
          "racist": false,
          "sexist": false,
          "explicit": false
        },
        "id": 34,
        "safe": true,
        "lang": "en"
      }
    ]
  },
  {
    "error": false,
    "amount": 5,
    "jokes": [
      {
        "category": "Programming",
        "type": "single",
        "joke": "\nHoney, go to the store and buy some eggs.\n\"OK.\n\"Oh and while you're there, get some milk.\n\"He never returned.",
        "flags": {
          "nsfw": false,
          "religious": false,
          "political": false,
          "racist": false,
          "sexist": false,
          "explicit": false
        }
      }
    ]
  }
]
```

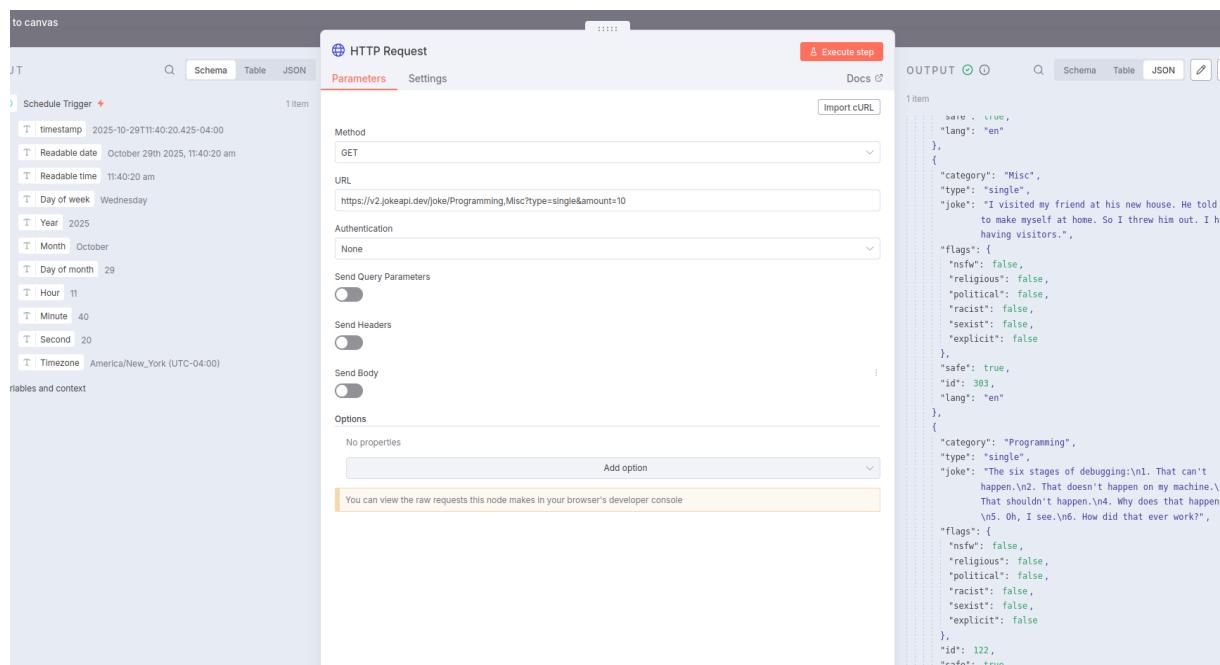
Para clasificar los chistes por su nivel de seguridad se añadio un nodo IF, donde se evaluó si el campo booleano safe es verdadero o no. En caso de ser verdadero se distribuira a través de la rama true a un nodo Edit Fields (Set) donde se pondrá un mensaje de Chiste seguro, en el caso de que no sea verdadero se mandará a través de la rama false a un nodo Edit Fields (Set) donde se mostrará el mensaje Chiste NO seguro.



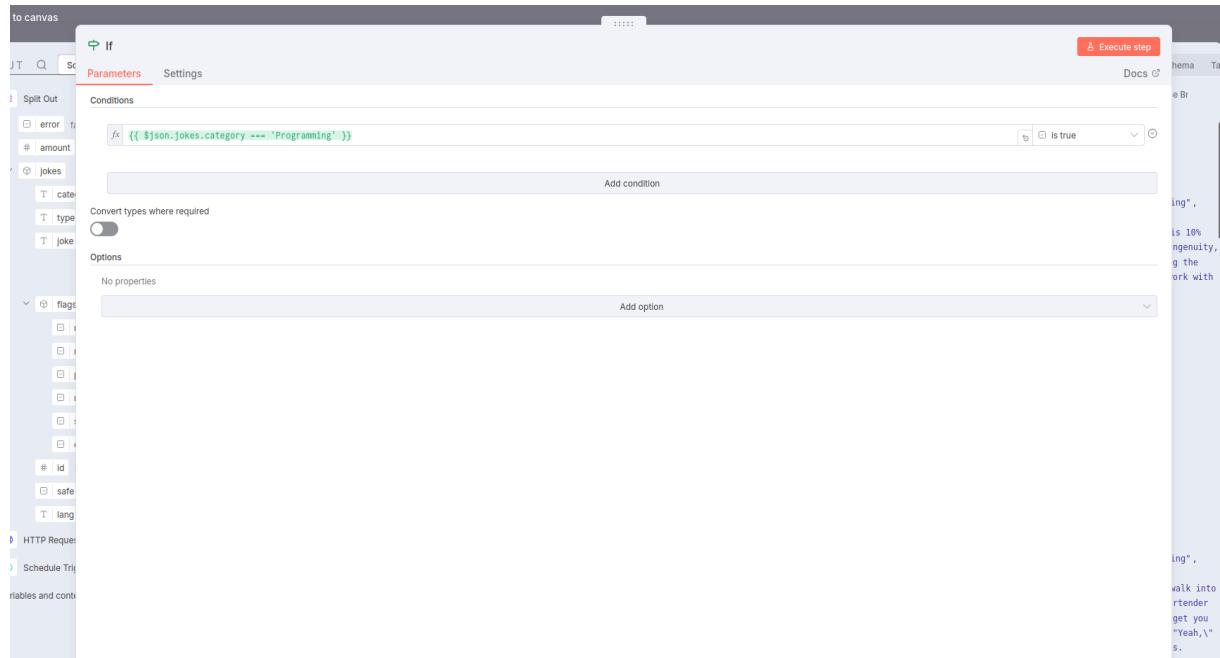
Capítulo 2

Ejercicio 1

El objetivo del ejercicio 1 es clasificar los chistes por su categoría, diferenciando entre Programming y Misc. Para comenzar el flujo de trabajo es necesario añadir un nodo inicial Schedule Trigger. Este nodo se ha configurado para ejecutar el nodo automáticamente cada minuto. A continuación, añadimos un nodo HTTP Request para solicitar chistes de las dos categorías mediante la URL.



El siguiente paso es añadir un nodo IF para diferenciar los chistes por categorías. Por lo que deberemos configurar el nodo para que si la categoria del chiste es programming se vaya a la rama true y si es misc se vaya a la rama false.



A continuación del nodo IF se añadirán un nodo Edit Fields (set) al final de cada rama para mostrar la categoría de cada chiste. En la siguiente foto se podrá ver los chistes de categoría Misc.

Capítulo 3

Ejercicio 2

El objetivo del ejercicio 2 es crear un flujo de trabajo que analice los productos de un comercio. Por lo que comenzaremos poniendo un nodo Schedule Trigger que se ejecute una vez al dia.

The screenshot shows a workflow editor interface with a 'Schedule Trigger' node selected. The left panel displays configuration options for the trigger, including:

- Rules**:
 - Trigger Interval**: Days
 - Days Between Triggers**: 1
 - Trigger at Hour**: Midnight
 - Trigger at Minute**: 0

An orange box highlights the 'Execute step' button above the configuration area. The right panel shows the **INPUT** tab with a JSON representation of the trigger's execution context:

```
{
  "timestamp": "2025-10-29T11:54:56.736-04:00",
  "Readable date": "October 29th 2025, 11:54:56 am",
  "Readable time": "11:54:56 am",
  "Day of week": "Wednesday",
  "Year": "2025",
  "Month": "October",
  "Day of month": "29",
  "Hour": "11",
  "Minute": "54",
  "Second": "56",
  "Timezone": "America/New_York (UTC-04:00)"
}
```

A continuacion, se añadio un nodo HTTP Request para realizar la solicitud a la API para obtener los datos mediante la URL.

The screenshot shows a Node-RED interface with the following components:

- Left Panel (Variables and context):** Displays a "Schedule Trigger1" node with the following properties:
 - timestamp: 2025-10-29T11:53:24.001-04:00
 - Readable date: October 29th 2025, 11:53:24 am
 - Readable time: 11:53:24 am
 - Day of week: Wednesday
 - Year: 2025
 - Month: October
 - Day of month: 29
 - Hour: 11
 - Minute: 53
 - Second: 24
 - Timezone: America/New_York (UTC-04:00)
- Middle Panel (HTTP Request Node):** A "HTTP Request1" node with the following configuration:
 - Method: GET
 - URL: <https://fakestoreapi.com/products>
 - Authentication: None
 - Send Query Parameters: Enabled
 - Send Headers: Enabled
 - Send Body: Enabled
 - Options: No properties
- Right Panel (Output):** Shows the response from the HTTP request, displayed as JSON. The output is a list of products, with the first two items shown below:

```
[{"id": 1, "title": "Fjallraven - Foldsack No. 1 Backpack, Fits 15 Laptops", "price": 109.95, "description": "Your perfect pack for everyday use and walking in the forest. Stash your laptop (up to 15 inches) in the padded sleeve, your everyday", "category": "men's clothing", "image": "https://fakestoreapi.com/img/81fPKd-2AYL._AC_SL1500_t.png", "rating": {"rate": 3.9, "count": 128}, "id": 2, "title": "Mens Casual Premium Slim Fit T-Shirts ", "price": 22.3, "description": "Slim-fitting style, contrast raglan long sleeve, three-button henley placket, light weight & soft fabric for breathable and comfortable wearing. And Solid stitched shirt with round neck made for durability and a great fit for casual fashion wear and diehard baseball fans. The Henley style round neck includes a three-button placket.", "category": "men's clothing", "image": "https://fakestoreapi.com/img/71-3HjGNDUL._AC_SX879_.SX_.UX_.SY_.t.png", "rating": {"rate": 4.1, "count": 259}}
```

El siguiente paso consistió en añadir un nodo IF para realizar la comparación de cada producto. El objetivo es separar los productos en los que tienen un precio superior a 100 y los que tienen un precio igual o inferior.

The screenshot shows a data processing interface with a central configuration panel and a left sidebar.

Central Panel:

- Title:** If1
- Buttons:** Execute step, Parameters, Settings, Docs
- Conditions:** A condition is defined: `fn: {{ $json.price }}` is greater than 100.
- Options:** No properties are defined.

Left Sidebar (J.T. View):

- Split Out1 (Info)
- HTTP Request (20 items)
- Schedule Trigger1 (1 item)
- Variables and context

Right Sidebar (OUTPUT):

Execute this node to view data

Una vez separados los productos por precio se añadirá dos nodos Edit Fields(Set) al final de cada rama, donde en la rama true tendremos los productos de más de 100 de valor y el la rama false los de menos de 100.

```

[{"Etiqueta": "Producto Premium"}, {"Etiqueta": "Producto Premium"}]
  
```

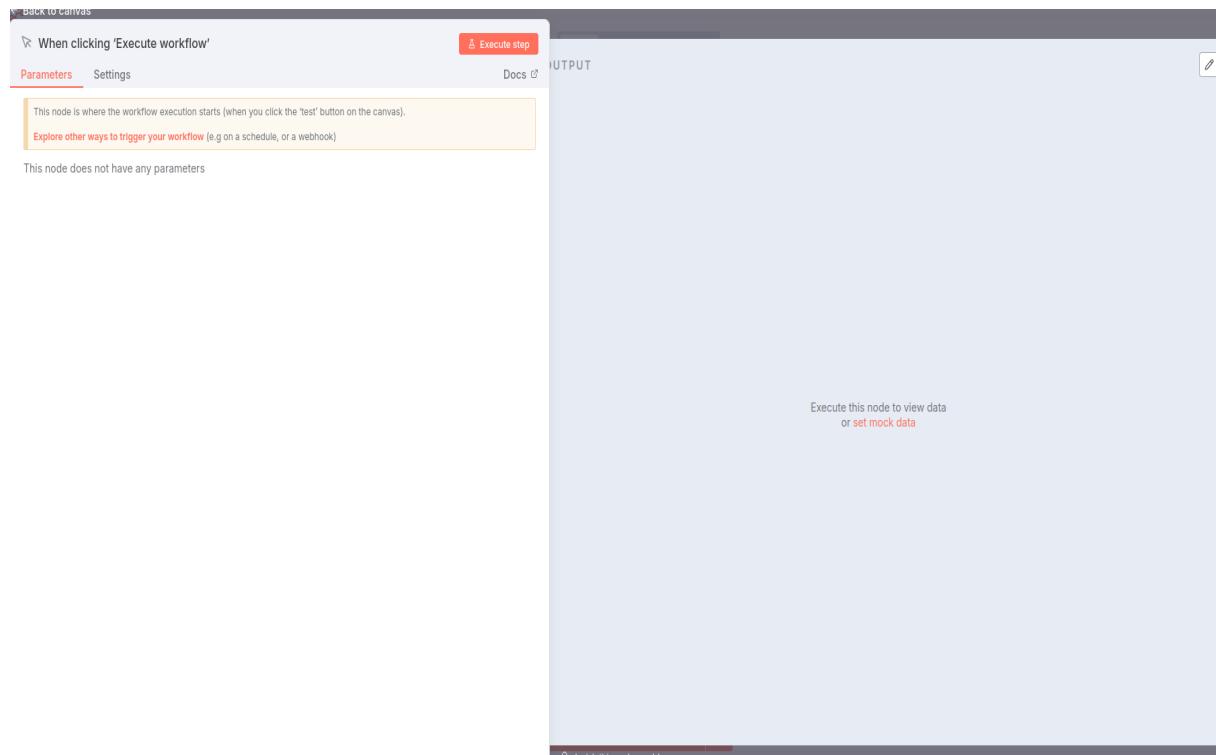
```

[{"etiqueta": "Producto Est\'andar"}, {"etiqueta": "Producto Est\'andar"}]
  
```

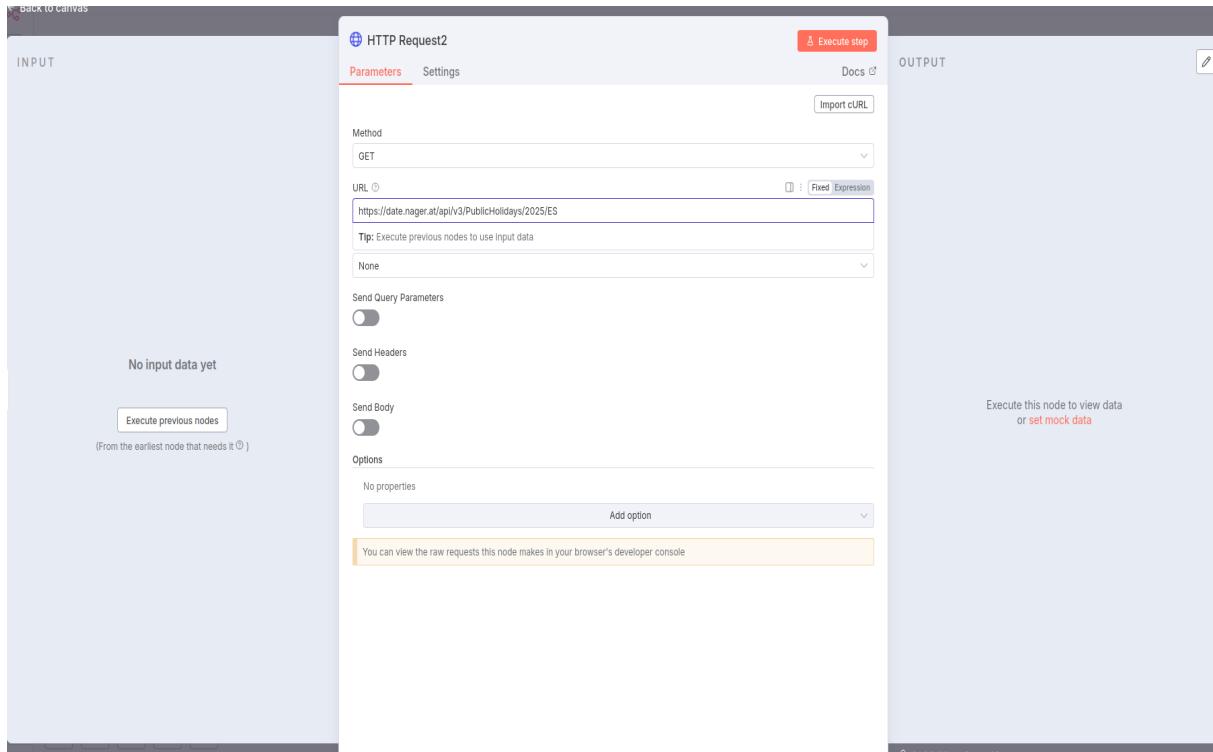
Capítulo 4

Ejercicio 3

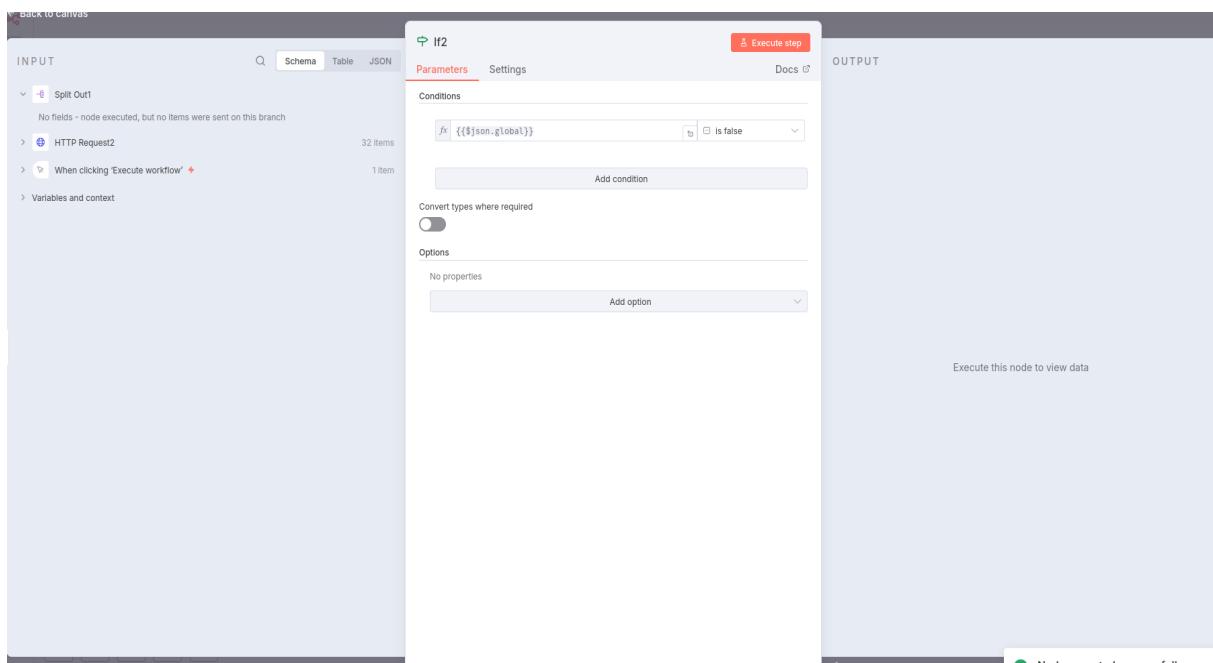
El objetivo del ejercicio 3 ha sido crear un flujo de trabajo que obtuviera los festivos nacionales y, únicamente a esos días, consultara un API para sugerir una actividad. Para ello se añadió un nodo Manual Trigger, que permite al usuario iniciar la ejecución del flujo manualmente.



El siguiente paso fue agregar un nodo HTTP Request para obtener los festivos oficiales en España.



Una vez obtenidos los festivos se añadirá un nodo IF para comprobar si el festivo es nacional o no. Para ello, se comprobará si el booleano global es false. Esto quiere decir que el festivo es local.



En la rama true se añadió un nodo HTTP Request para obtener las actividades aleatorias. Además, ha sido necesario usar un nodo Merge para poder conectar el HTTP Request con los festivos locales y el HTTP Request con las actividades aleatorias.

```

{
    "activity": "Organize your movie collection",
    "availability": 0,
    "type": "busywork",
    "participants": 1,
    "price": 0,
    "accessibility": "Few no challenges",
    "duration": "hours",
    "kidFriendly": true,
    "link": null,
    "key": 6378359
}
{
    "activity": "Have a football scrimmage with some friends",
    "availability": 0.2,
    "type": "social",
    "participants": 8,
    "price": 0,
    "accessibility": "Major challenges",
    "duration": "minutes",
    "kidfriendly": true,
    "link": null,
    "key": null
}
  
```

A continuación se creó un nodo Edit Fields(Set) para estructurar la salida del flujo donde se podrá ver el nombre del festivo y la actividad asignada a él.

```

{
    "name": "Día de Andalucía",
    "activity": "Organize your movie collection"
},
{
    "name": "Día de les Illes Balears",
    "activity": "Have a football scrimmage with some friends"
},
{
    "name": "Jueves Santo",
    "activity": "Find a DIY to do"
},
{
    "name": "Lunes de Pascua",
    "activity": "Mow your neighbor's lawn"
},
{
    "name": "Día de Castilla y León",
    "activity": "Mow your lawn"
},
{
    "name": "San Jorge (Día de Aragón)",
    "activity": "Go to the gym"
},
{
    "name": "Fiesta de la Comunidad de Madrid",
    "activity": "Do a jigsaw puzzle"
},
{
    "name": "Día das Letras Galegas",
    "activity": "Fix something that's broken in your house"
},
{
    "name": "Día de Canarias",
    "activity": "Do a jigsaw puzzle"
}
  
```