



FULL STACK

**Comenzamos en unos  
minutos**

---

ACADEMY  
by NUMEN

# Javascript: Clase 8

## API REST y CRUD

# ¿Qué es una API?

Una API es un conjunto de definiciones y protocolos que se utiliza para desarrollar e **integrar** el software de las aplicaciones. API significa interfaz de programación de aplicaciones.



# ¿Que es un end point?

Un endpoint es cualquier dispositivo que sea físicamente la parte final de una red. Las computadoras de escritorio, las tablets, los smartphones, los dispositivos de oficina de red, como los routers, las impresoras y las cámaras de seguridad también son considerados endpoints. Los servidores también pueden ser considerados endpoints porque también están conectados a la red.

Básicamente cualquier dispositivo final conectado a la red es un endpoint.

## User Endpoints

GET	/users/self	*** Get information about the owner of the access token.
GET	/users/ user-id	*** Get information about a user.
GET	/users/self/media/recent	*** Get the most recent media of the user.
GET	/users/ user-id /media/recent	*** Get the most recent media of a user.
GET	/users/self/media/liked	*** Get the recent media liked by the user.
GET	/users/search	*** Search for a user by name.

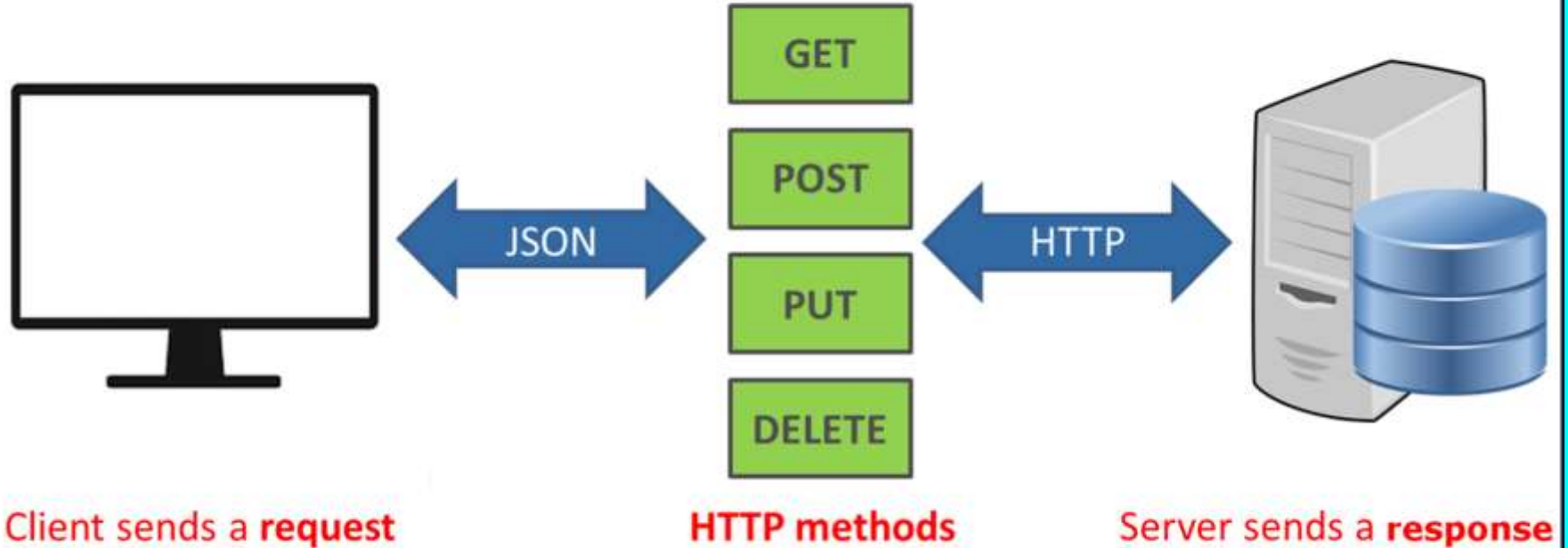
# REST

REST es cualquier interfaz entre sistemas que use HTTP para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles, como XML y JSON. Es una alternativa en auge a otros protocolos estándar de intercambio de datos como SOAP (Simple Object Access Protocol), que disponen de una gran capacidad pero también mucha complejidad.

A veces es preferible **una solución más sencilla de manipulación de datos como REST.**



# Arquitectura Rest



# ¿Que es CRUD?

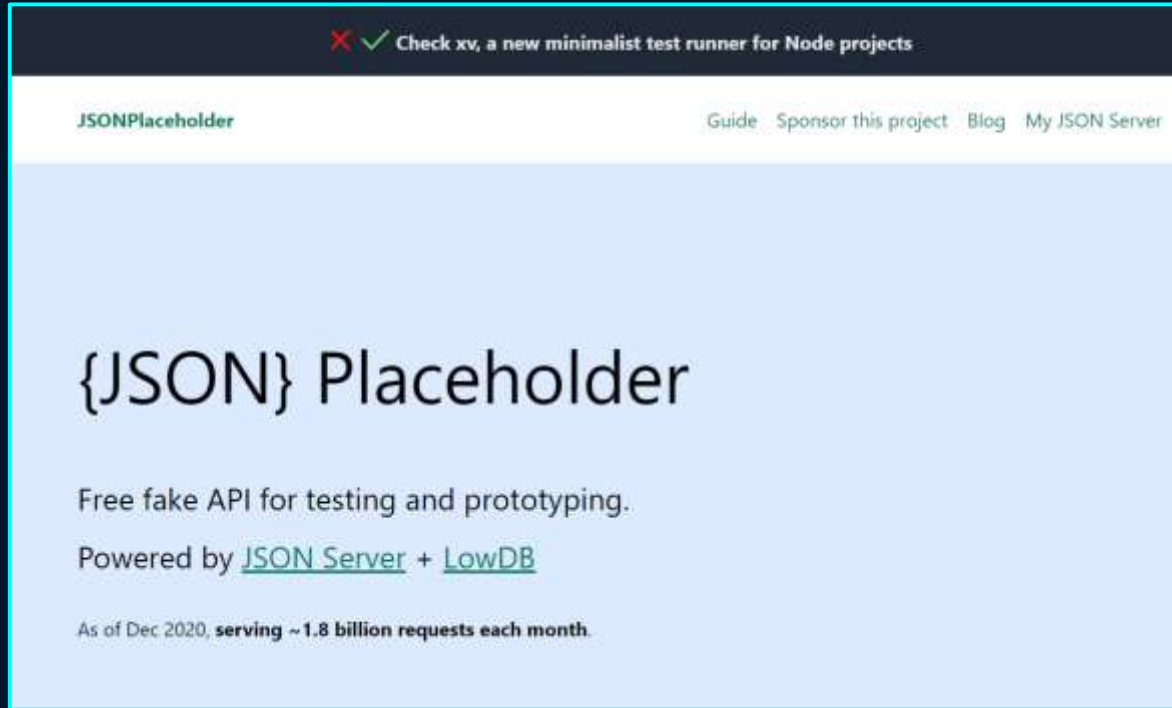


---

**Create   Read   Update   Delete**



# JSON Server



Enlace: <https://github.com/typicode/json-server>

# Instalando JSON Server

Teniendo instalado Node, ejecutamos el siguiente comando para instalar JSON Server en nuestro proyecto.

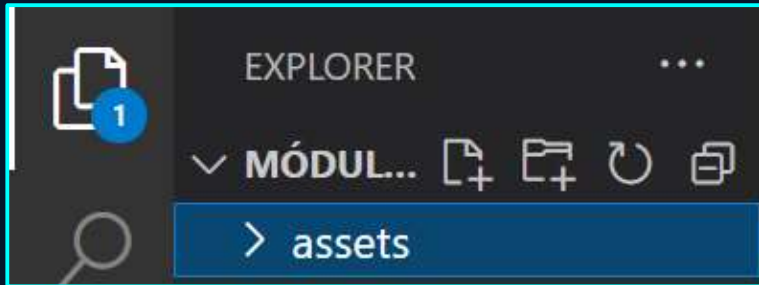
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

powershell + v [ ] [ ] ^ X

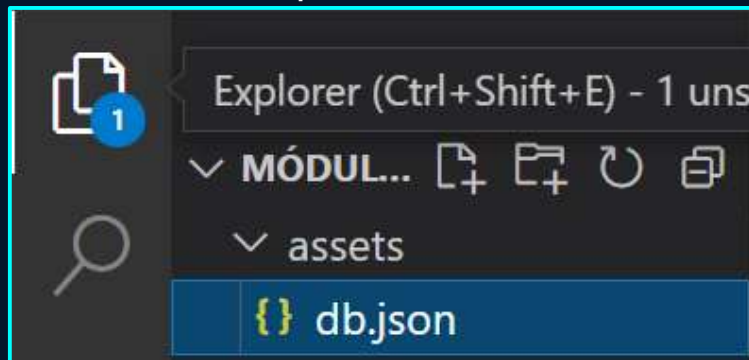
```
PS C:\Users\NicoD\OneDrive\Escritorio\Módulo Javascript> npm install -g json-server
```

# Conectando nuestra base de datos con JSON Server

**Paso 1:** Creamos una carpeta llamada assets



**Paso 2:** Dentro de assets creamos un archivo json llamado db.json con la información que deseamos manipular.



**Paso 3:** Abrimos la terminal y escribimos el siguiente comando

```
PS C:\Users\NicoD\OneDrive\Escritorio\Módulo Javascript> json-server --watch db.json
```

# El Arreglo de objeto que utilizaremos en db.json

```
{
  "pokemons": [
    {
      "id": 1,
      "name": "Bulbasaur",
      "type": ["Grass", "Poison"]
    },
    {
      "id": 2,
      "name": "Ivysaur",
      "type": ["Grass", "Poison"]
    },
    {
      "id": 3,
      "name": "Venusaur",
      "type": ["Grass", "Poison"]
    },
    {
      "id": 4,
      "name": "Charmander",
      "type": ["Fire"]
    },
    {
      "id": 5,
      "name": "Charmeleon",
      "type": ["Fire"]
    },
    {
      "id": 6,
      "name": "Charizard",
      "type": ["Fire", "Flying"]
    },
    {
      "id": 7,
      "name": "Squirtle",
      "type": ["Water"]
    },
    {
      "id": 8,
      "name": "Wartortle",
      "type": ["Water"]
    },
    {
      "id": 9,
      "name": "Blastoise",
      "type": ["Water"]
    }
  ]
}
```

# Descargando Insomnia Rest

The image shows the Insomnia REST client website and a preview of the application interface. The website header includes the Insomnia logo, the text "by Kong", a GitHub star count of 17,971, and navigation links for Products, Docs, Pricing, Plugins, Login, and a "Get Started for Free" button. The main content area features a "Download Insomnia" heading, a descriptive paragraph about building better APIs, and a prominent blue "Download Insomnia for Windows" button. Below the button is a privacy policy agreement and links for downloading on other operating systems. The interface preview on the right shows a list of endpoints under the "pet" and "store" collections, with the "POST /pet" endpoint selected. The right-hand pane displays the JSON body of the selected endpoint, which is a pet object with fields like id, category, name, photoUrls, tags, and status.

Insomnia <sup>by Kong</sup> 17,971 Products Docs Pricing Plugins Login [Get Started for Free →](#)

## Download Insomnia

Start building, designing, testing better APIs through spec-first development driven by an APIOps CI/CD pipelines.

[Download Insomnia for Windows](#)

By downloading and using Insomnia, I agree to the [Privacy Policy](#) and [Terms](#).

[What's New?](#) [Changelog](#)

Not your OS? [Download for MacOS](#) | [Ubuntu](#) or [See all downloads](#).

**DOCUMENT**

- No Environment
- No Cookies (Add)
- No Client Certificates (Add)

**ENDPOINTS**

- pet
  - GET /pet/{petId}
  - POST /pet/{petId}
  - DELETE /pet/{petId}
  - POST /pet/{petId}/uploadImage
  - POST /pet
  - PUT /pet
  - GET /pet/findByStatus
  - GET /pet/findByTags
- store
  - GET /store/inventory
  - GET /store/order/{orderId}
  - DELETE /store/order/{orderId}
  - POST /store/order
- user

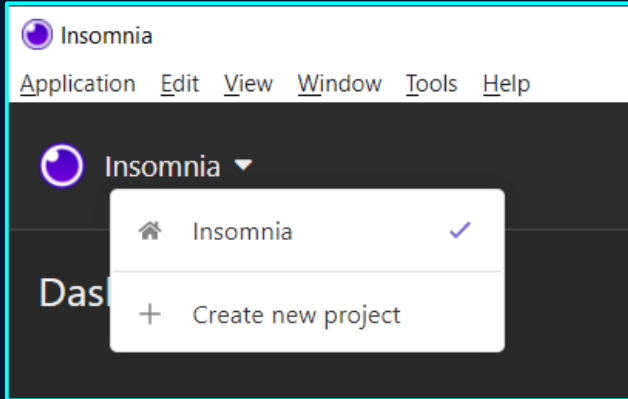
**POST = base\_url/pet** Send

**JSON = Auth Query Header**

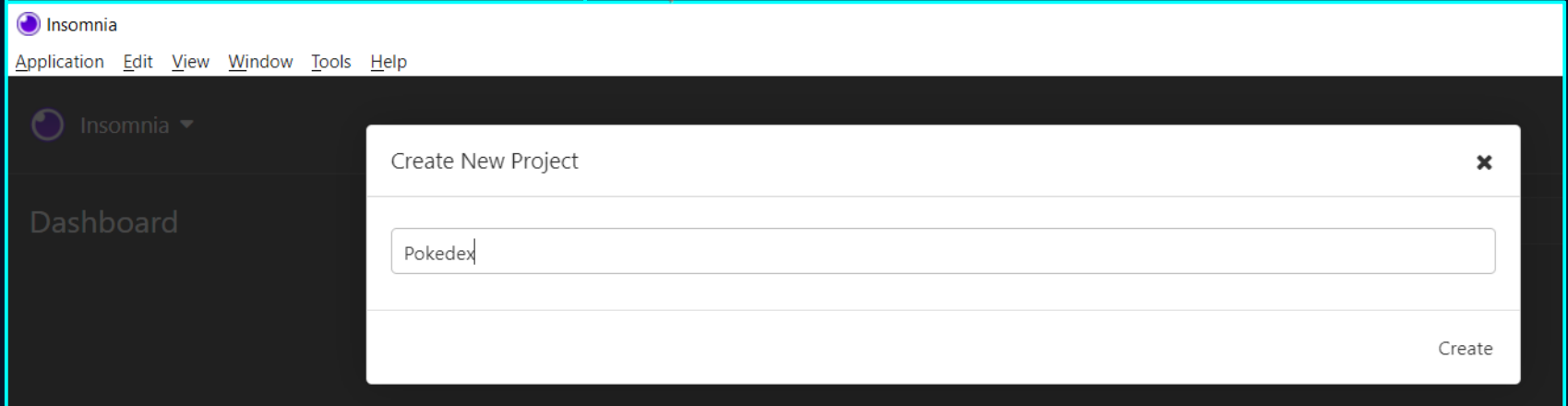
```
{
  "id": 0,
  "category": {
    "id": 0,
    "name": "string"
  },
  "name": "doggie",
  "photoUrls": [
    "string"
  ],
  "tags": [
    {
      "id": 0,
      "name": "string"
    }
  ],
  "status": "string"
}
```

Headfly REST

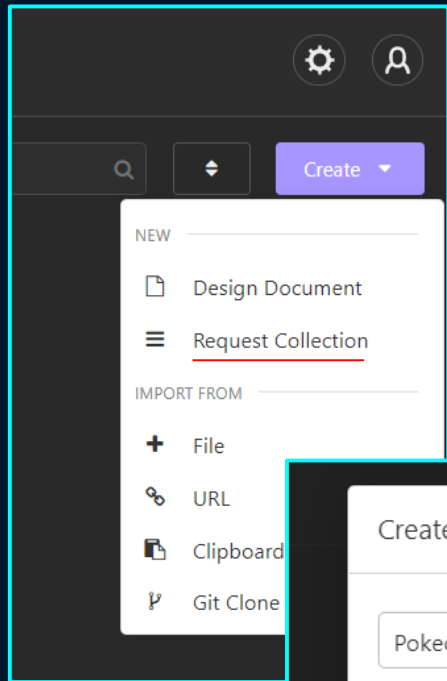
# Creando nuevo proyecto



Iniciamos creando un nuevo proyecto donde tendremos nuestras colecciones pedidos de consumición a APIs.



# Creando una nueva colección



Ahora crearemos nuestro primer pedido de colección.

A screenshot of a dialog box titled 'Create New Request Collection' with a close button (X) in the top right corner. Inside the dialog, there is a text input field containing the word 'Pokedex'. At the bottom right of the dialog is a button labeled 'Create'.

# Creando una nueva colección

Ahora crearemos lo que será nuestro primer pedido de consumición a través del método GET. Una vez puesto el link de nuestro JSON Server estamos listos para consumir los datos del cliente.

The image shows a two-part interface for a REST client. The top part is a 'New Request' dialog box with a title bar 'New Request' and a close button 'x'. It contains a label 'Name (defaults to your request URL if left empty)' above a text input field containing 'READ Pokedex'. To the right of the input is a dropdown menu showing 'GET'. Below the input is a tip: '\* Tip: paste Curl command into URL afterwards to import it'. A 'Create' button is at the bottom right. A red curved arrow points from the 'Create' button to the bottom part of the interface. The bottom part is the main dashboard, titled 'Dashboard / Pokedex' with a purple circular icon. It has a 'No Environment' dropdown and a 'Cookies' section. The main area shows a list of requests with columns for 'Method', 'URL', and 'Body'. The first request is 'GET http://localhost:3000/pokedex' with a 'Send' button. Below the list, there are tabs for 'Body', 'Auth', 'Query', 'Header', and 'Docs'. The 'Body' tab is selected, showing an empty text area.

New Request

Name (defaults to your request URL if left empty)

READ Pokedex

GET

\* Tip: paste Curl command into URL afterwards to import it

Create

Dashboard / Pokedex

No Environment

Cookies

GET http://localhost:3000/pokedex

Send

Filter

Body

Auth

Query

Header

Docs

GET READ Pokedex



# HTML de nuestro CRUD (Parte 1)

```
<h1>POKE CRUD</h1>
  <section class="crud">
    <article>
      <h2 class="crud-title">Agregar Pokemon</h2>
      <form class="crud-form">
        <input type="text" name="name" placeholder="name"
required>
        <br>
        <input type="text" name="type" placeholder="type"
required>
        <br>
        <input type="submit" value="Enviar">
        <input type="hidden" name="id">
      </form>
    </article>
```



# HTML de nuestro CRUD (Parte 2 y 3)

```
<article>
  <h2>Pokelista</h2>
  <table class="crud-table">
    <thead>
      <tr>
        <th>Nombre</th>
        <th>Tipo</th>
        <th>Acciones</th>
      </tr>
    </thead>
    <tbody>

    </tbody>
  </table>
</article>
</section>
```



```
<template id="crud-template">
  <tr>
    <td class="name"></td>
    <td class="type"></td>
    <td>
      <button class="edit">Editar</button>
      <button class="delete">Eliminar</button>
    </td>
  </tr>
</template>
```

# Conectado JS con nuestro documento HTML

```
// Referenciando selectores
const d = document,
      $table = d.querySelector(".crud-table"),
      $form = d.querySelector(".crud-form"),
      $title = d.querySelector(".crud-title"),
      $template = d.getElementById("crud-template").content,
      $fragment = d.createDocumentFragment();
```

# Programando la consumición de datos

```
const ajax = (options) => {  
  let { url, method, success, error, data } = options;  
  const xhr = new XMLHttpRequest();  
  
  xhr.addEventListener("readystatechange", e => {  
    if (xhr.readyState !== 4) return;  
  
    if (xhr.status >= 200 && xhr.status < 300) {  
      let json = JSON.parse(xhr.responseText);  
      success(json);  
    } else {  
      let message = xhr.statusText || "Ocurrió un error";  
      error(`Error ${xhr.status}: ${message}`);  
    }  
  });  
  
  xhr.open(method || "GET", url);  
  xhr.setRequestHeader("Content-type", "application/json; charset=utf-8");  
  xhr.send(JSON.stringify(data));  
}
```

# Programando la inserción en el DOM

```
const getAll = () => {  
  ajax({  
    url: "http://localhost:3000/pokemons",  
    success: (res) => {  
      console.log(res);  
  
      res.forEach(el => {  
        $template.querySelector(".name").textContent = el.name;  
        $template.querySelector(".type").textContent = el.type;  
        $template.querySelector(".edit").dataset.id = el.id;  
        $template.querySelector(".edit").dataset.name = el.name;  
        $template.querySelector(".edit").dataset.type = el.type;  
        $template.querySelector(".delete").dataset.id = el.id;  
  
        let $clone = d.importNode($template, true);  
        $fragment.appendChild($clone);  
      });  
  
      $table.querySelector("tbody").appendChild($fragment);  
    },  
    error: (err) => {  
      console.log(err);  
      $table.insertAdjacentHTML("afterend", `<p><b>${err}</b></p>`);  
    }  
  })  
}  
  
d.addEventListener("DOMContentLoaded", getAll);
```

# Programando el evento "submit" (parte 1 - POST)

```
d.addEventListener("submit", e => {  
  if (e.target === $form) {  
    e.preventDefault();  
  
    if (!e.target.id.value) {  
      //Create - POST  
      ajax({  
        url: "http://localhost:3000/pokemons",  
        method: "POST",  
        success: (res) => location.reload(),  
        error: (err) => $form.insertAdjacentHTML("afterend", `

<b>${err}</b></p>`),  
        data: {  
          name: e.target.name.value,  
          type: e.target.type.value  
        }  
      });  
    }  
  }  
});


```



# Programando el evento “submit” (parte 2 - PUT)

```
} else {  
    //Update - PUT  
    ajax({  
        url: `http://localhost:3000/pokemons/${e.target.id.value}`,  
        method: "PUT",  
        success: (res) => location.reload(),  
        error: (err) => $form.insertAdjacentHTML("afterend", `

<b>${err}</b></p>`),  
        data: {  
            name: e.target.name.value,  
            type: e.target.type.value  
        }  
    });  
}  
}


```

# Programando botones de UPDATE y DELETE (parte 1)

```
d.addEventListener("click", e => {  
  if (e.target.matches(".edit")) {  
    $title.textContent = "Editar Pokemon";  
    $form.name.value = e.target.dataset.name;  
    $form.type.value = e.target.dataset.type;  
    $form.id.value = e.target.dataset.id;  
  }  
  
  if (e.target.matches(".delete")) {  
    let isDelete = confirm(`¿Estás seguro de eliminar el id ${e.target.dataset.id}?`);  
  }  
});
```





# Programando botones de UPDATE y DELETE (parte 2)

```
if (isDelete) {  
    //Delete - DELETE  
    ajax({  
        url: `http://localhost:3000/pokemons/${e.target.dataset.id}`,  
        method: "DELETE",  
        success: (res) => location.reload(),  
        error: (err) => alert(err)  
    });  
}  
  
})
```

ACADEMY  
by NUMEN