

TEMA 9: jQuery

e. AJAX

1. ¿Qué es AJAX?

- AJAX significa JavaScript asíncrono y XML, y le permite recuperar contenido del servidor de fondo de forma asíncrona, sin una actualización de página. Por lo tanto, le permite actualizar el contenido de una página web sin volver a cargarla.
- Veamos un ejemplo para comprender cómo podría usar AJAX en el desarrollo diario de su aplicación. Supongamos que desea crear una página que muestre la información de perfil de un usuario, con diferentes secciones como información personal, información social, notificaciones, mensajes, etc.

1. ¿Qué es AJAX?

- El enfoque habitual sería construir diferentes páginas web para cada sección.
- Así, por ejemplo, los usuarios harían clic en el enlace de información social para volver a cargar el navegador y mostrar una página con la información social.
- Sin embargo, esto hace que sea más lento navegar entre secciones, ya que el usuario tiene que esperar a que el navegador se vuelva a cargar y la página se vuelva a mostrar cada vez.

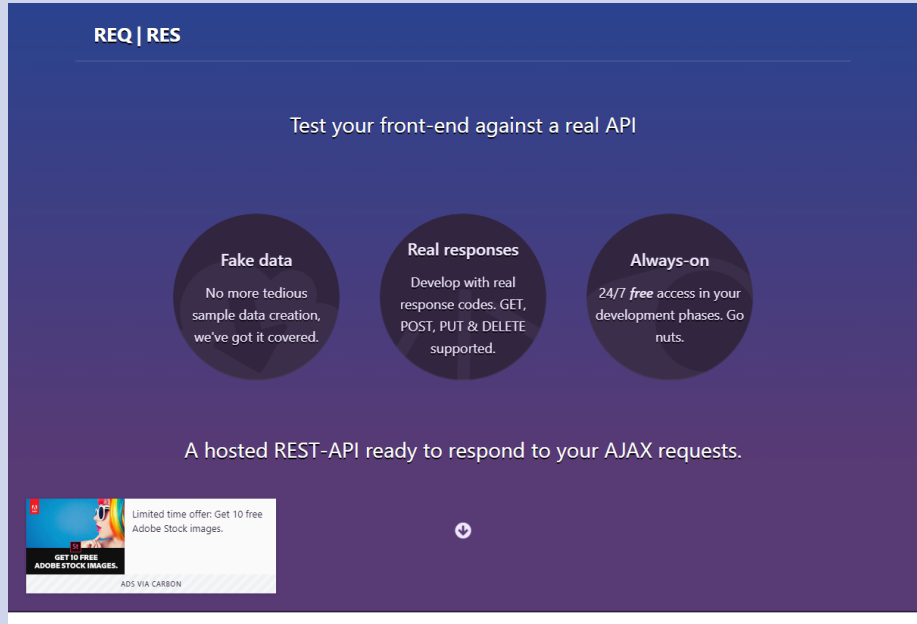
1. ¿Qué es AJAX?

- Por otro lado, también puede usar AJAX para crear una interfaz que cargue toda la información sin actualizar la página.
- En este caso, puede mostrar diferentes pestañas para todas las secciones, y al hacer clic en la pestaña, obtiene el contenido correspondiente del servidor de servicios de fondo y actualiza la página sin actualizar el navegador.
- Esto le ayuda a mejorar la experiencia general del usuario final.

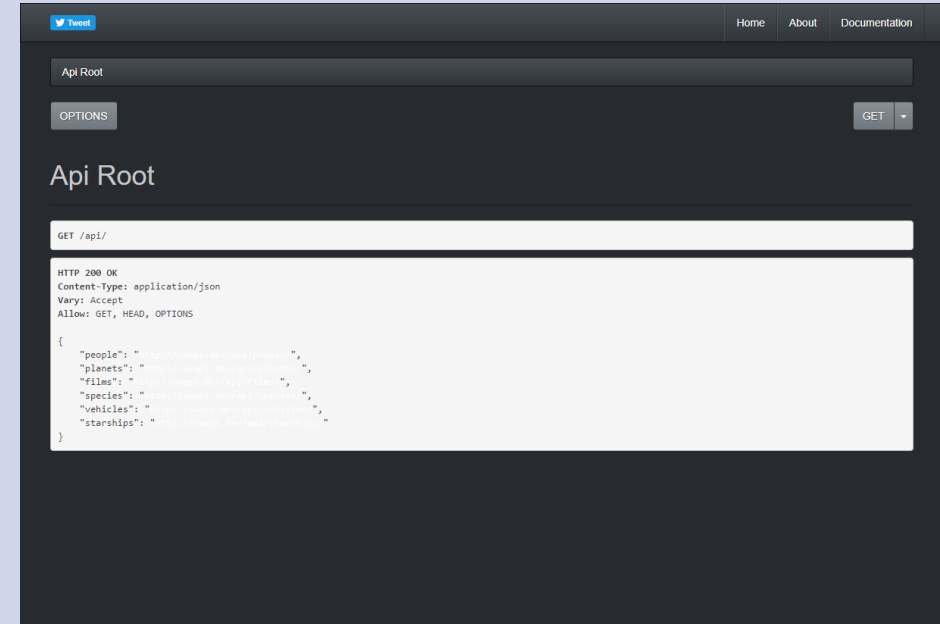
2. Webs de trabajo

- Hay cientos de web que ofrecen servicios REST-API. Vamos a centrarnos en dos de ellas, Reqres (<https://reqres.in/>) y Stars Wars API (<https://swapi.dev/api/>):

<https://reqres.in/>



<https://swapi.dev/api/>



3. Métodos Ajax de jQuery

- jQuery posee varios métodos para trabajar con Ajax.
- Sin embargo, todos están basados en el método **\$.ajax**, por lo tanto, su comprensión es obligatoria.
- A continuación se abarcará dicho método y luego se indicará un breve resumen sobre los demás métodos.
- Generalmente, es preferible utilizar el método **\$.ajax** en lugar de los otros, ya que ofrece más características y su configuración es muy comprensible.

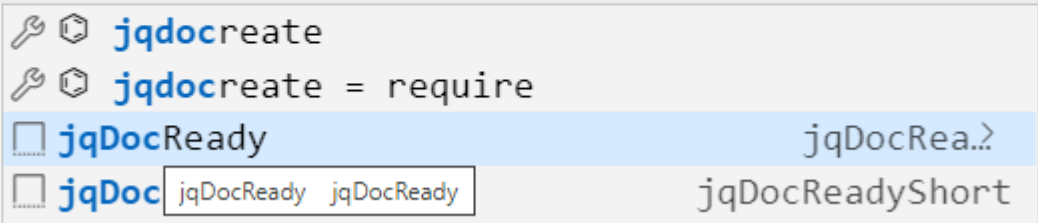
4. \$.ajax

- El método **\$.ajax** se configura mediante un objeto, el cual contiene todas las instrucciones que necesita jQuery para completar la petición.
- Dicho método es particularmente útil debido a que ofrece la posibilidad de especificar acciones en caso que la petición haya fallado o no.
- Además, al estar configurado a través de un objeto, es posible definir sus propiedades de forma separada, haciendo que sea más fácil la reutilización del código.
- En <https://api.jquery.com/jQuery.Ajax/> se puede consultar la documentación completa sobre las opciones disponibles en el método.

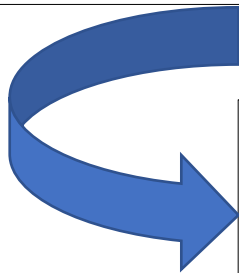
5. Utilizar el método \$.ajax

- En primer lugar generamos el `$(document).ready` (**jqDocReady** con jquery code snippets en VS code).

```
JS index.js
1  'use strict'
2  jqdoc
```



The screenshot shows the VS Code autocomplete menu for the variable 'jqdoc' on line 2. The menu lists several suggestions: 'jqdoccreate', 'jqdoccreate = require', 'jqDocReady' (highlighted in blue), and 'jqDoc' (with a sub-menu showing 'jqDocReady' and 'jqDocReadyShort').



```
1  'use strict'
2  $(document).ready(function () {
3
4  });
```


5. Utilizar el método \$.ajax

- Escribimos ahora jqAjax (de nuevo con jquery snippets instalado) y nos genera un montón de código.

```
$(document).ready(function () {  
    $.ajax({  
        type: "method",  
        url: "url",  
        data: "data",  
        dataType: "dataType",  
        success: function (response) {  
              
        }  
    });  
});
```

6. Utilizar el método \$.Ajax. Recibir datos

Vamos a ver uno a uno lo que nos ha generado el snippet.

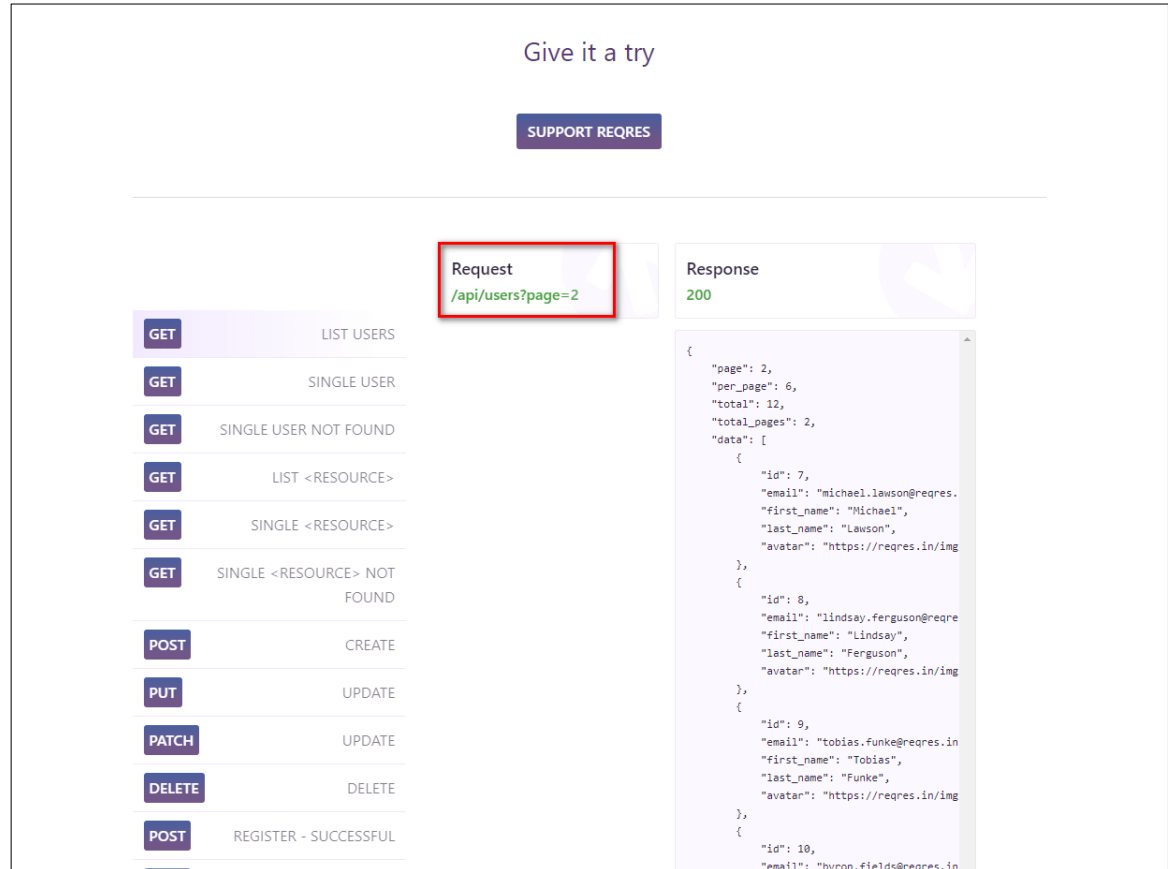
- **Type:** es el método de envío, tenemos varias opciones: **GET, POST, PUT y DELETE.**
- **Url:** es la web a la que vamos a hacer la petición, si está fuera de nuestro dominio habrá que incluir la ruta completa, http:// o https://, este campo es obligatorio.
- **Data:** Establece la información que se enviará al servidor. Esta puede ser tanto un objeto como una cadena de datos (por ejemplo tipo=consola&modelo=play5).

6. Utilizar el método \$.Ajax. Recibir datos

- **dataType**: Establece el tipo de información que se espera recibir como respuesta del servidor. Si no se especifica ningún valor, de forma predeterminada, jQuery revisa el tipo que posee la respuesta.
- **Success**: Establece una función a ejecutar si la petición a sido satisfactoria. Dicha función recibe como argumentos la información de la petición (convertida a objeto JavaScript en el caso que dataType sea JSON).

6. Utilizar el método \$.Ajax. Recibir datos

- Vamos a realizar una petición a las web Reqres y la vamos a mostrar por pantalla. Hacemos clic donde pone **Request**.



The screenshot displays the Reqres API documentation interface. At the top, there is a "Give it a try" button and a "SUPPORT REQRES" link. Below these, a list of API endpoints is shown on the left, each with a method (GET, POST, PUT, PATCH, DELETE) and a description. The "Request" tab is selected, showing the URL `/api/users?page=2`. The "Response" tab shows the JSON response, which includes pagination information and a list of user data.

Give it a try

SUPPORT REQRES

Request

`/api/users?page=2`

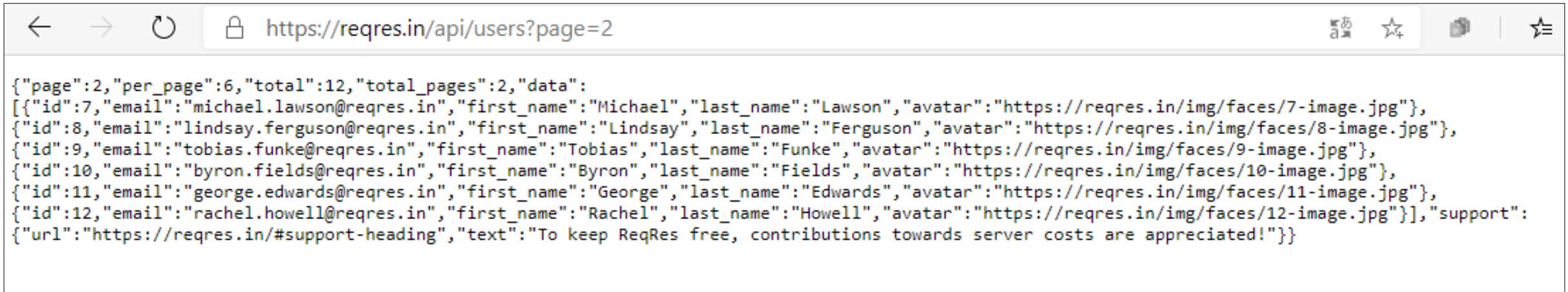
Response

200

```
{
  "page": 2,
  "per_page": 6,
  "total": 12,
  "total_pages": 2,
  "data": [
    {
      "id": 7,
      "email": "michael.lawson@reqres.",
      "first_name": "Michael",
      "last_name": "Lawson",
      "avatar": "https://reqres.in/img"
    },
    {
      "id": 8,
      "email": "lindsay.ferguson@reqre",
      "first_name": "Lindsay",
      "last_name": "Ferguson",
      "avatar": "https://reqres.in/img"
    },
    {
      "id": 9,
      "email": "tobias.funke@reqres.in",
      "first_name": "Tobias",
      "last_name": "Funke",
      "avatar": "https://reqres.in/img"
    },
    {
      "id": 10,
      "email": "byron.fields@reqres.in"
    }
  ]
}
```

6. Utilizar el método \$.Ajax. Recibir datos

- Nos llevará a la pagina que genera un fichero JSON. Esa será la URL



A screenshot of a web browser window. The address bar shows the URL `https://reqres.in/api/users?page=2`. The main content area displays a JSON response. The JSON object has a `page` property set to 2, a `per_page` property set to 6, a `total` property set to 12, and a `total_pages` property set to 2. The `data` property is an array of 6 user objects, each containing `id`, `email`, `first_name`, `last_name`, and `avatar` fields. The `support` property is an object with `url` and `text` fields.

```
{
  "page": 2,
  "per_page": 6,
  "total": 12,
  "total_pages": 2,
  "data": [
    {
      "id": 7,
      "email": "michael.lawson@reqres.in",
      "first_name": "Michael",
      "last_name": "Lawson",
      "avatar": "https://reqres.in/img/faces/7-image.jpg"
    },
    {
      "id": 8,
      "email": "lindsay.ferguson@reqres.in",
      "first_name": "Lindsay",
      "last_name": "Ferguson",
      "avatar": "https://reqres.in/img/faces/8-image.jpg"
    },
    {
      "id": 9,
      "email": "tobias.funke@reqres.in",
      "first_name": "Tobias",
      "last_name": "Funke",
      "avatar": "https://reqres.in/img/faces/9-image.jpg"
    },
    {
      "id": 10,
      "email": "byron.fields@reqres.in",
      "first_name": "Byron",
      "last_name": "Fields",
      "avatar": "https://reqres.in/img/faces/10-image.jpg"
    },
    {
      "id": 11,
      "email": "george.edwards@reqres.in",
      "first_name": "George",
      "last_name": "Edwards",
      "avatar": "https://reqres.in/img/faces/11-image.jpg"
    },
    {
      "id": 12,
      "email": "rachel.howell@reqres.in",
      "first_name": "Rachel",
      "last_name": "Howell",
      "avatar": "https://reqres.in/img/faces/12-image.jpg"
    }
  ],
  "support": {
    "url": "https://reqres.in/#support-heading",
    "text": "To keep ReqRes free, contributions towards server costs are appreciated!"
  }
}
```

6. Utilizar el método \$.Ajax. Recibir datos

- Como puede apreciarse en la imagen el método que debemos utilizar es GET (por la forma en la que se aprecia la URL).
- Ya tenemos dos parámetros configurados.

```
$.ajax({  
    type: "GET",  
    url: "https://reqres.in/api/users?page=2",
```

6. Utilizar el método \$.Ajax. Recibir datos

- El campo **data**, como no vamos a enviar información (vamos a recibirla), no hay que tocarlo, por tanto, lo dejamos tal cual está.
- Como hemos observado, la web devuelve como resultado un fichero en formato JSON, por tanto en **dataType** le indicaremos tal hecho, mediante la etiqueta "JSON".

```
$.ajax({  
    type: "GET",  
    url: "https://reqres.in/api/users?page=2",  
    data: "data",  
    dataType: "JSON",  
})
```

6. Utilizar el método \$.Ajax. Recibir datos

- Nos queda la ultima parte, que es el parámetro **success** , que corresponde con una función callback que se ejecutará si la operación ha tenido un resultado exitoso.
- La función recibe un parámetro donde se almacenará el resultado obtenido por la consulta a la web, en nuestro caso recibirá un objeto JSON.
- Nos quedará ya completo de la siguiente forma:

6. Utilizar el método \$.Ajax. Recibir datos

```
$.ajax({  
    type: "GET",  
    url: "https://reqres.in/api/users?page=2",  
    data: "data",  
    dataType: "JSON",  
    success: function (response) {  
        // response es un objeto.  
        console.log(response); // objeto completo  
        console.log(response.total_pages); //  
    }  
});
```

6. Utilizar el método \$.Ajax. Recibir datos

- El resultado obtenido sería el siguiente:

```
index.js:10
▼ Object i
  ▶ data: (6) [{...}, {...}, {...}]
  ▶ page: 2
  ▶ per_page: 6
  ▶ support: {url: "https:..."}
  ▶ total: 12
  ▶ total_pages: 2
  ▶ __proto__: Object
2 index.js:11
> |
```

6. Utilizar el método \$.Ajax. Recibir datos

- En data es donde esta el contenido. Si hago desde el código lo siguiente:

```
console.log(json.data);
```

- Obtendré como resultado:

```
▼ Array(6) ⓘ
  ► 0: {id: 1, name: "cerulean", year: 2000, color: "#98B2D1", pantone_value: "15-4020"}
  ► 1: {id: 2, name: "fuchsia rose", year: 2001, color: "#C74375", pantone_value: "17-2031"}
  ► 2: {id: 3, name: "true red", year: 2002, color: "#BF1932", pantone_value: "19-1664"}
  ► 3: {id: 4, name: "aqua sky", year: 2003, color: "#7BC4C4", pantone_value: "14-4811"}
  ► 4: {id: 5, name: "tigerlily", year: 2004, color: "#E2583E", pantone_value: "17-1456"}
  ► 5: {id: 6, name: "blue turquoise", year: 2005, color: "#53B0AE", pantone_value: "15-5217"}
  length: 6
```

6. Utilizar el método \$.Ajax. Recibir datos

- Refinando un poco el ejercicio anterior, añadimos una hoja de estilo, y un <div> con varios <div> en su interior. Tenemos el HTML:

```
<script src="https://code.jquery.com/jquery-3.5.1.min.js" integrity="sha256-  
<script type="text/javascript" src="index.js"></script>  
<link rel="stylesheet" href="style.css">  
</head>  
<body>  
  <div id="contenedor" class="contenedor">  
    <div><h3>ID</h3></div>  
    <div><h3>Nombre</h3></div>  
    <div><h3>Año</h3></div>  
    <div><h3>color</h3></div>  
    <div><h3>Pantone</h3></div>  
  </div>
```

6. Utilizar el método \$.Ajax. Recibir datos

- La hoja de estilo incorporará un grid CSS y 5 columnas de igual tamaño:

```
.contenedor {  
    display: grid;  
    grid-template-columns: 1fr 1fr 1fr 1fr 1fr;  
}
```

6. Utilizar el método \$.Ajax. Recibir datos

- Y ahora hacemos la magia con jQuery:

```
$(document).ready(function () {  
    $.ajax({  
        type: 'GET',  
        dataType: 'json',  
        url: 'https://reqres.in/api/unknown',  
  
        success : function(json) {  
            console.log(json.data);  
            var contenedor=$("#contenedor");  
  
            for(let element of json.data) {  
                let hijo1=$("<div />");  
                $(hijo1).html(element.id);  
                $(contenedor).append(hijo1);  
  
                let hijo2=$("<div />");  
                $(hijo2).html(element.name);  
                $(contenedor).append(hijo2);  
            }  
        }  
    });  
});
```

6. Utilizar el método \$.Ajax. Recibir datos

```
let hijo3=$("<div />");
$(hijo3).html(element.year);
$(contenedor).append(hijo3);

let hijo4=$("<div />");
$(hijo4).html(element.color);
$(contenedor).append(hijo4);

let hijo5=$("<div />");
$(hijo5).html(element.pantone_value);
$(contenedor).append(hijo5);
}
},
});
});
```

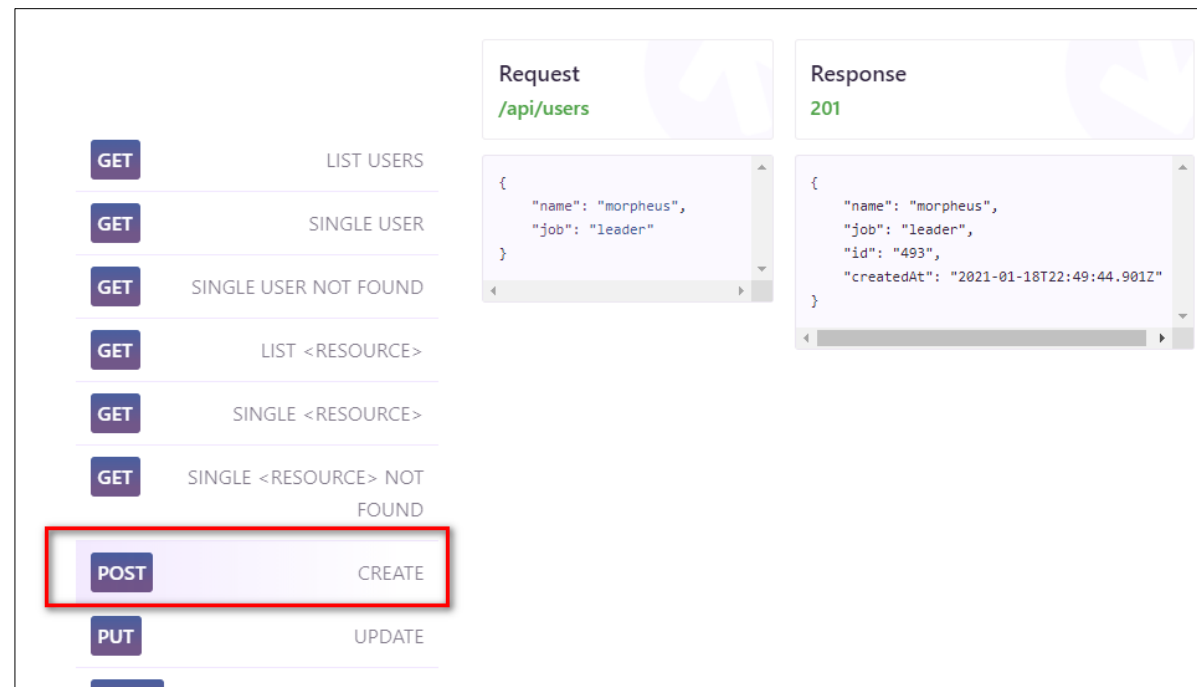
6. Utilizar el método \$.Ajax. Recibir datos

- Y obtenemos como resultado:

ID	Nombre	Año	color	Pantone
1	cerulean	2000	#98B2D1	15-4020
2	fuchsia rose	2001	#C74375	17-2031
3	true red	2002	#BF1932	19-1664
4	aqua sky	2003	#7BC4C4	14-4811
5	tigerlily	2004	#E2583E	17-1456
6	blue turquoise	2005	#53B0AE	15-5217

7. Utilizar el método \$.Ajax. Enviar datos

- En este caso vamos a la inversa del anterior, en lugar de recibir datos, vamos a enviarlos al servidor para que los procese como considere oportuno.
- Volvemos a nuestra página de cabecera ReqRes y vamos a seleccionar el método POST.



7. Utilizar el método \$.Ajax. Enviar datos

- En este caso, vemos que enviamos unos valores con formato JSON y se recibe una respuesta por parte del servidor, también en formato JSON.



Request	Response
<code>/api/users</code>	<code>201</code>
<pre>{ "name": "morpheus", "job": "leader"} </pre>	<pre>{ "name": "morpheus", "job": "leader", "id": "411", "createdAt": "2021-01-18T23:07:56.556Z"} </pre>

7. Utilizar el método \$.Ajax. Enviar datos

- Comenzamos la creación de la aplicación. Como siempre generaremos `$(document).ready. (jqdocready)`

```
$(document).ready(function () {  
    |  
});
```

7. Utilizar el método \$.Ajax. Enviar datos

- Generamos también \$.ajax (jqajax).

```
$(document).ready(function () {  
    $.ajax({  
        type: "method",  
        url: "url",  
        data: "data",  
        dataType: "dataType",  
        success: function (response) {  
  
        }  
    });  
});
```

7. Utilizar el método \$.Ajax. Enviar datos

- Como nos indica la web ReqRes, debemos introducir los siguientes valores:
- **Type:** POST
- **url:** <https://reqres.in/api/users>
- **dataType:** JSON
- **data:** en este caso este valor es muy importante, pues es la información que vamos a introducir en el servidor. En nuestro caso la vamos a introducir un **objeto** con un campo **name** y un campo **job**.
- **Success:** Podemos visualizar lo devuelto por el servidor.

7. Utilizar el método \$.Ajax. Enviar datos

```
$(document).ready(function () {  
    let datos={  
        name: 'Juan', job: 'Leader'  
    }  
    $.ajax({  
        type: "POST",  
        url: "https://reqres.in/api/users",  
        data: datos,  
        dataType: "JSON",  
        success: function (response) {  
            console.log(response);  
            console.log(response.id);  
            console.log(response.name)  
        }  
    });  
});
```

7. Utilizar el método \$.Ajax. Enviar datos



- El resultado obtenido es el siguiente:


```
index3.js:11
▼ {name: "Juan", job: "Leader", id: "460", createdAt: "2021-01-18T23:44:33.828Z"} ⓘ
  createdAt: "2021-01-18T23:44:33.828Z"
  id: "460"
  job: "Leader"
  name: "Juan"
  ► __proto__: Object
460 index3.js:12
Juan index3.js:13
>
```

8. Mezclando las Churras con las Merinas. Creando un API Rest en PHP

- En primer lugar vamos a crear una Base de datos nueva con PHPMyAdmin, le llamaremos API (por ejemplo).

Bases de datos

 Crear base de datos 



8. Creando un API Rest en PHP

- Una vez dentro de la base de datos, debemos crear una tabla con el código postal y el nombre de todas las provincias de España.
- Ese trabajo ya está hecho, me voy a la web: https://github.com/alombarte/utilities/blob/master/sql/spain_provincias.sql que incluye un fichero .sql con la sentencia para la creación de la tabla y la inserción de contenido

```
CREATE TABLE `provincias` (  
  `id_provincia` smallint(6) DEFAULT NULL,  
  `provincia` varchar(30) DEFAULT NULL  
) ENGINE=InnoDB;  
  
INSERT INTO `provincias` (`id_provincia`, `provincia`)  
VALUES  
  (2, 'Albacete'),  
  (3, 'Alicante/Alacant'),  
  (4, 'Almería'),  
  (1, 'Araba/Álava'),  
  (33, 'Asturias'),  
  (5, 'Ávila'),
```

8. Creando un API Rest en PHP

- Selecciono toda esta parte la copio y la pego en la ventana SQL de PHPMyAdmin.



The screenshot shows the PHPMyAdmin interface with the 'SQL' tab selected. The title bar indicates 'Servidor: 127.0.0.1 » Base de datos: api'. Below the title bar are buttons for 'Estructura', 'SQL', 'Buscar', and 'Generar una consulta'. A message box says 'Ejecutar la(s) consulta(s) SQL en la base de datos api:'. The SQL editor contains the following code:

```
1 CREATE TABLE `provincias` (  
2   `id_provincia` smallint(6) DEFAULT NULL,  
3   `provincia` varchar(30) DEFAULT NULL  
4 ) ENGINE=InnoDB;  
5  
6 INSERT INTO `provincias` (`id_provincia`, `provincia`)  
7 VALUES  
8   (2, 'Albacete'),  
9   (3, 'Alicante/Alacant'),  
10  (4, 'Almería'),  
11  (1, 'Araba/Álava'),  
12  (33, 'Asturias'),
```

8. Creando un API Rest en PHP

- Creamos un fichero .php dentro de nuestro servidor (por ejemplo, XAMPP) llamado provincia.php. En este fichero vamos a incluir una consulta tabla provincias de la base de datos.

```
$conexion=new PDO("mysql:host=localhost;dbname=api", "root", "");
$consulta="SELECT * FROM provincias";
$resultado=$conexion->query($consulta)->fetchAll();
$contador=0;
foreach($resultado as $fila) {
    $dato[$contador]['id']=$fila['id_provincia'];
    $dato[$contador]['provincia']=$fila['provincia'];
    $contador++;
}
```

- En la variable \$dato, tenemos un array bidimensional con el id y el nombre de dicha provincia.

8. Creando un API Rest en PHP

- Las APIS no pueden devolver los datos en forma de array, deben devolver los datos como un objeto JSON.
- En PHP tenemos funciones para trabajar con Arrays y convertirlos a JSON y viceversa.
- Para convertir un array a JSON usaremos **json_encode**, como nuestro resultado incluye acentos, letras ñ y demás debemos añadir un segundo parámetro: **JSON_UNESCAPED_UNICODE**

```
$datos=json_encode($dato,JSON_UNESCAPED_UNICODE);
```

8. Creando un API Rest en PHP

- Como hemos observado trabajando con otras APIs realmente no devuelven nada, simplemente muestran el contenido del JSON por pantalla, pues nada, lo hacemos así también:

```
echo $datos;
```

- Si unimos todas las piezas tenemos que la pagina PHP quedaría así:

8. Creando un API Rest en PHP

```
$conexion=new PDO("mysql:host=localhost;dbname=api", "root", "");  
$consulta="SELECT * FROM provincias";  
$resultado=$conexion->query($consulta)->fetchAll();  
$contador=0;  
foreach($resultado as $fila) {  
    $dato[$contador]['id']=$fila['id_provincia'];  
    $dato[$contador]['provincia']=$fila['provincia'];  
    $contador++;  
}  
$datos=json_encode($dato,JSON_UNESCAPED_UNICODE);  
echo $datos;
```

8. Creando un API Rest en PHP

- Si el fichero .js que va a acceder a esta página estuviese en nuestro servidor, no habría que hacer nada más.
- Para asegurarnos que vamos a poder acceder también desde fuera del servidor, debemos añadir esta línea al principio de la pagina, y, ahora si, nuestra parte PHP estaría concluida.

```
header('Access-Control-Allow-Origin: *');
```

8. Creando un API Rest en PHP

- Podemos comprobar el resultado de llamar a nuestra pagina desde el servidor:

```
[{"id": "2", "provincia": "Albacete"}, {"id": "3", "provincia": "Alicante\\Alacant"}, {"id": "4", "provincia": "Almería"}, {"id": "1", "provincia": "Araba\\Álava"}, {"id": "33", "provincia": "Asturias"}, {"id": "5", "provincia": "Ávila"}, {"id": "6", "provincia": "Badajoz"}, {"id": "7", "provincia": "Balears, Illes"}, {"id": "8", "provincia": "Barcelona"}, {"id": "48", "provincia": "Bizkaia"}, {"id": "9", "provincia": "Burgos"}, {"id": "10", "provincia": "Cáceres"}, {"id": "11", "provincia": "Cádiz"}, {"id": "39", "provincia": "Cantabria"}, {"id": "12", "provincia": "Castellón\\Castelló"}, {"id": "51", "provincia": "Ceuta"}, {"id": "13", "provincia": "Ciudad Real"}, {"id": "14", "provincia": "Córdoba"}, {"id": "15", "provincia": "Coruña, A"}, {"id": "16", "provincia": "Cuenca"}, {"id": "20", "provincia": "Gipuzkoa"}, {"id": "17", "provincia": "Girona"}, {"id": "18", "provincia": "Granada"}, {"id": "19", "provincia": "Guadalajara"}, {"id": "21", "provincia": "Huelva"}, {"id": "22", "provincia": "Huesca"}, {"id": "23", "provincia": "Jaén"}, {"id": "24", "provincia": "León"}, {"id": "27", "provincia": "Lugo"}, {"id": "25", "provincia": "Lleida"}, {"id": "28", "provincia": "Madrid"}, {"id": "29", "provincia": "Málaga"}, {"id": "52", "provincia": "Melilla"}, {"id": "30", "provincia": "Murcia"}, {"id": "31", "provincia": "Navarra"}, {"id": "32", "provincia": "Ourense"}, {"id": "34", "provincia": "Palencia"}, {"id": "35", "provincia": "Palmas, Las"}, {"id": "36", "provincia": "Pontevedra"}, {"id": "26", "provincia": "Rioja, La"}, {"id": "37", "provincia": "Salamanca"}, {"id": "38", "provincia": "Santa Cruz de Tenerife"}, {"id": "40", "provincia": "Segovia"}, {"id": "41", "provincia": "Sevilla"}, {"id": "42", "provincia": "Soria"}, {"id": "43", "provincia": "Tarragona"}, {"id": "44", "provincia": "Teruel"}, {"id": "45", "provincia": "Toledo"}, {"id": "46", "provincia": "Valencia\\València"}, {"id": "47", "provincia": "Valladolid"}, {"id": "49", "provincia": "Zamora"}, {"id": "50", "provincia": "Zaragoza"}]
```


8. Creando un API Rest en PHP

- Si instalamos el complemento “json viewer” a nuestro navegador, nos muestra la página de una forma “mas ordenada”.

```
// 20210128235951
// http://localhost/provincia.php

[
  {
    "id": "2",
    "provincia": "Albacete"
  },
  {
    "id": "3",
    "provincia": "Alicante/Alacant"
  },
  {
    "id": "4",
    "provincia": "Almería"
  },
  {
    "id": "1",
    "provincia": "Araba/Álava"
  },
  {
    "id": "33",
    "provincia": "Asturias"
  },
  {
    "id": "5",
    "provincia": "Ávila"
  },
]
```

9. Acceder a nuestra API desde jQuery

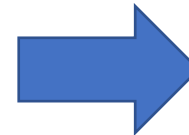
- Tenemos el siguiente código HTML :

```
<h2>Mostrar CP de provincias</h2>  
<select name="provincia" id="provincia"> </select>  
<span id="salida"></span>
```

9. Acceder a nuestra API desde jQuery

- El código jQuery asociado es:

```
$(document).ready(function () {  
    let provincia=$("#provincia");  
    var salida="<option value=''>Selecciona una provincia ... </option>";  
    $.ajax({  
        type: "GET",  
        url: "http://localhost/provincia.php",  
        data: "data",  
        dataType: "JSON",  
        success: function (response) {  
            for(let elemento of $(response)) {  
                salida+="<option value='"+elemento.id+"'>"+elemento.provincia+"</option>";  
            }  
            $(provincia).html(salida);  
        }  
    });  
  
    $(provincia).change(function (e) {  
        $("#salida").html($(provincia).val());  
    });  
});
```



Mostrar CP de provincias

Ciudad Real ▼ 13