Ajax:

Al incorporar JQuery a nuestro proyecto podemos acceder al método $.ajax(), que nos permitirá crear un objeto AJAX para solicitar recursos, en este caso un objeto JSON

```
$.ajax({
        url:'https://jsonplaceholder.typicode.com/comments',
        type:'GET',
        async: true,
        data: $('#myForm').serialize(),
        success: function(result){
                let str_result = "";
                result.forEach((v) => {
                        str_result += `{<br>postId: ${v.postId},<br>id: ${v.id},<br>name: ${v.name},<br>body: ${v.body}<br>}<br><br>`;
                });
                $('#result').html(str_result);
        },
        error: function(err){
                alert("An error occured: " + err.status + " " + err.statusText);
        }
});
```

Indicamos el enlace al recurso, que queremos recuperarlo (Petición GET), la consulta es asíncrona, los parámetros (La función serialize() recupera los valores de los campos del formulario myForm en un formato nombreatrib1=atrib1&nombreatrib2=atrib2…).

success contiene la función que procesa la respuesta devuelta por la página del recurso. Esta función recibe como parámetro el objeto recuperado con la consulta y, usando una cadena auxiliar, va añadiendo los valores de los campos del objeto a la cadena. Tras almacenar toda la información en la cadena, ésta es asignada al contenido del elemento <p> del documento que tiene identificador “id=result”.

error procesa la respuesta en caso de recibir un código de respuesta a la petición que indique un error en la misma. En este caso, usa la función alert() para notificar al usuario del código de error, además de su descripción corta.

Para asegurarnos de que el valor introducido en el formulario myForm es un valor numérico, antes de recurrir al objeto .ajax, usamos la función isNaN para evaluar el tipo de dato de la entrada del usuario. Si no es un número, notifica al usuario mediante la función alert.

```
itton').click(function() {
   if(!isNaN($('#myForm').serializeArray()[0].value)){
           $.ajax({
                   url:'https://jsonplaceholder.typicode
```

```
        });
} else {
        alert("An error occured: You must insert a number");
}
```

Firebase:

Para incluir firebase en nuestro proyecto, hemos creado una nueva página, usando la página principal como referencia. Después de enlazarla con la anterior, se ha añadido el siguiente formulario:

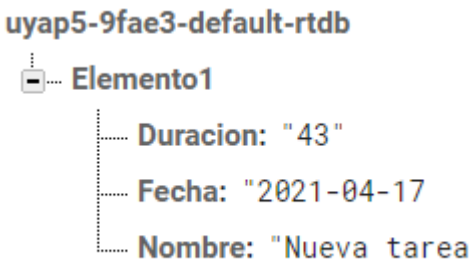
```
<p>//Página en construcción</p>
<p id="databasetask"></p>
<h1> Modificar una tarea </h1>

<form method="post" name="nameform" id="myForm">
        <ul>
                <li>
                        <label for="new_task">Inserte el Nombre de la tarea: </label>
                        <input type="text" name="Atributo1">
                </li>
                <br>
                <li>
                        <label for="new_task">Inserte la duración en horas de la tarea: </label>
                        <input type="number" name="Atributo2">
                </li>
                <br>
                <li>
                        <label for="new_task">Inserte la fecha límite de la tarea: </label>
                        <input type="date" name="Atributo3">
                </li>
                <br>
        </ul>
</form>

<button id="button" form="nameform" value="Submit">Submit</button>
<br>
```

Además, se ha añadido un nuevo párrafo <p> con “id=databasetask”, que se actualizará con el contenido de la base de datos.

La idea detrás de esta práctica es utilizar las funcionalidades de Firebase para configurar un back-end para nuestra página. Al acceder a Firebase creamos un nuevo proyecto, y desde la pestaña de Real Time Database podemos agregar una entrada con varios atributos:



Además, al crear la base de datos, podemos seleccionar un “Modo de pruebas” que nos permite asignar por defecto las siguientes reglas de acceso para todos los usuarios durante 30 días:

```
{
  "rules": {
    ".read": "now < 1620601200000",  // 2021-5-10
    ".write": "now < 1620601200000",  // 2021-5-10
  }
}
```

A continuación, para acceder a la base de datos desde nuestra página, copiamos la información proporcionada por Firebase al final del body de nuestro proyecto:

```
<script type="text/javascript" src="js/materialize.js"></script>
<script src="./jquery-3.6.0.min.js" type="text/javascript"></script>

<!-- The core Firebase JS SDK is always required and must be listed first -->
<script src="https://www.gstatic.com/firebasejs/7.14.0/firebase-app.js"></script>
<script src="https://www.gstatic.com/firebasejs/7.14.0/firebase-database.js"></script>

<!-- TODO: Add SDKs for Firebase products that you want to use
     https://firebase.google.com/docs/web/setup#available-libraries -->

<script>
  // Your web app's Firebase configuration
  var firebaseConfig = {
    apiKey: "AIzaSyB82_uWcKIs4wFA4uVOoybSsjMDlvcxuFo",
    authDomain: "uyap5-9fae3.firebaseapp.com",
    databaseURL: "https://uyap5-9fae3-default-rtdb.firebaseio.com",
    projectId: "uyap5-9fae3",
    storageBucket: "uyap5-9fae3.appspot.com",
    messagingSenderId: "1059776046703",
    appId: "1:1059776046703:web:e92f1b32731a203d9ecf9a"
  };
  // Initialize Firebase
  firebase.initializeApp(firebaseConfig);


      //Referencia a la base de datos
        var database = firebase.database();
```

En la variable database guardamos la referencia a la base de datos de Firebase.

```
var referencia = database.ref('Elemento1');
referencia.once('value', snapshot => {
        let str_aux = "";
        console.log(snapshot.val());
        str_aux += "Nombre: " + snapshot.val().Nombre + "<br>Duración: "
        + snapshot.val().Duracion + " horas <br>Fecha límite: " + snapshot.val().Fecha + "<br>";
        console.log(str_aux);
        $('#databasetask').html(str_aux);
});
```

Para obtener el valor de un objeto de la base de datos debemos recurrir a snapshots, pues Real Time Database opera en tiempo real, y los valores de un elemento pueden haber sido modificados después de haber inicializado Firebase en nuestra página. Una snapshot nos

permite capturar una instantánea de la información de una referencia a la base de datos en el momento de invocación.

Al cargar la página, se carga el contenido del elemento “Elemento1” en el párrafo con id “databasetask”. Para ello, usamos la función once, que nos permitirá trabajar con la información de la snapshot.

Para actualizar la entrada a la base de datos escribimos el siguiente código dentro de la función del evento click del botón del formulario (Para invocar al código únicamente tras interactuar con el botón):

```
$(document).ready(function(){
        $('#button').click(function() {
                let str_aux = "";
                referencia = database.ref('Elemento1');
                //Escribir un documento
                referencia.set({
                        Nombre: String($('#myForm').serializeArray()[0].value),
                        Duracion: String($('#myForm').serializeArray()[1].value),
                        Fecha: String($('#myForm').serializeArray()[2].value)
                });
                referencia.once('value', snapshot => {
                        console.log(snapshot.val());
                        str_aux += "Nombre: " + snapshot.val().Nombre + "<br>Duración: " + sr
                        console.log(str_aux);
                        $('#databasetask').html(str_aux);
                });
        });
});
</script>
```

Con la función set podemos sobreescribir los valores del elemento referenciado.