BRAÑOS RODRÍGUEZ ADRIÁN

Índice:

API sin Key……..…….. 3

API con Key…….....8

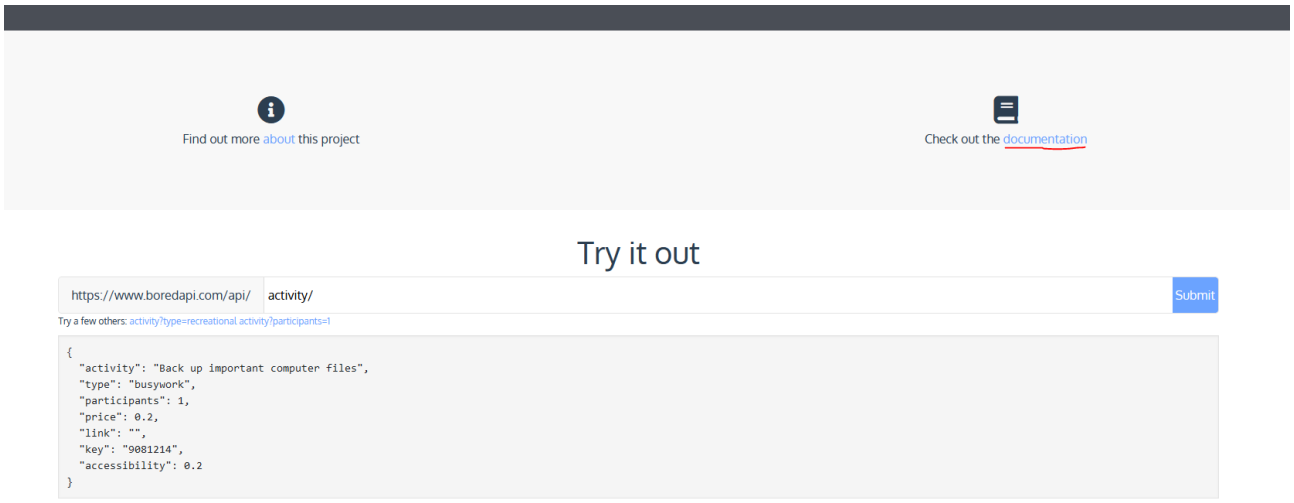
Bibliografía...……..29

API sin clave

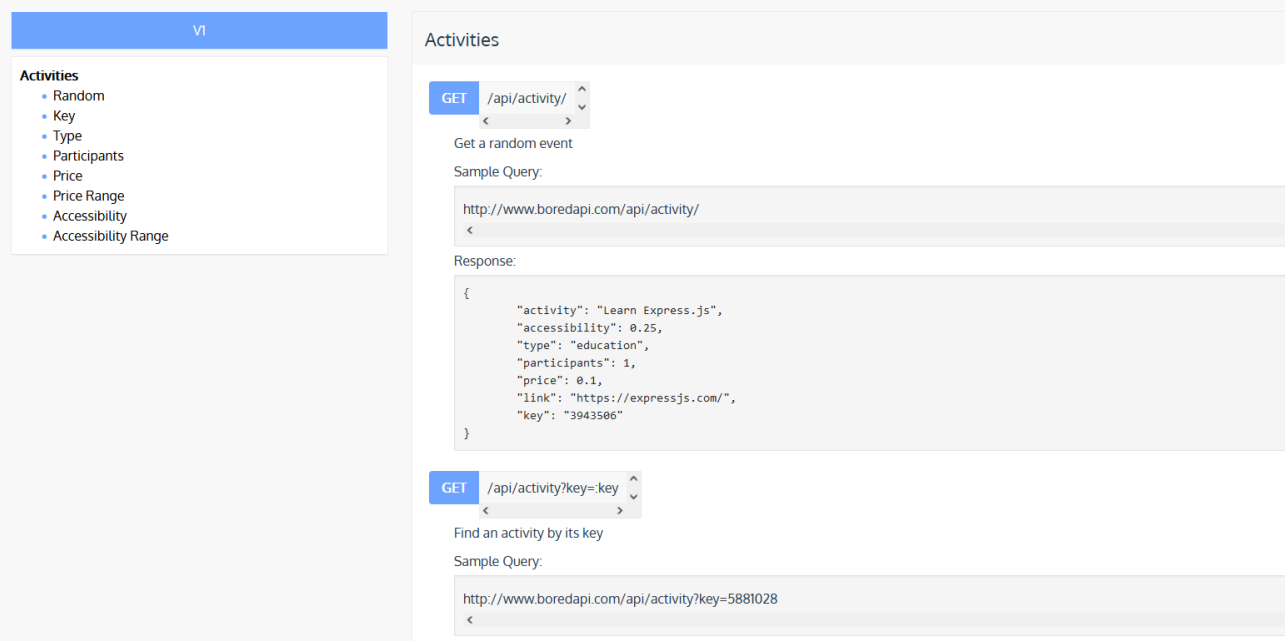
Para la api sin clave escogí la API de Bored

Es una Api que te da una actividad aleatoria por si no sabes que hacer o tienes mucho tiempo libre.

Para saber como funciona vamos a la documentación o mediante este enlace



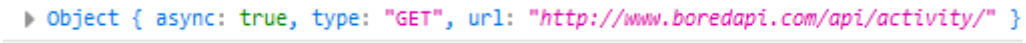
En la documentación nos da alguno valores como la actividad que vamos a desempeñar, de que tipo es la actividad, el numero de participantes, etc...



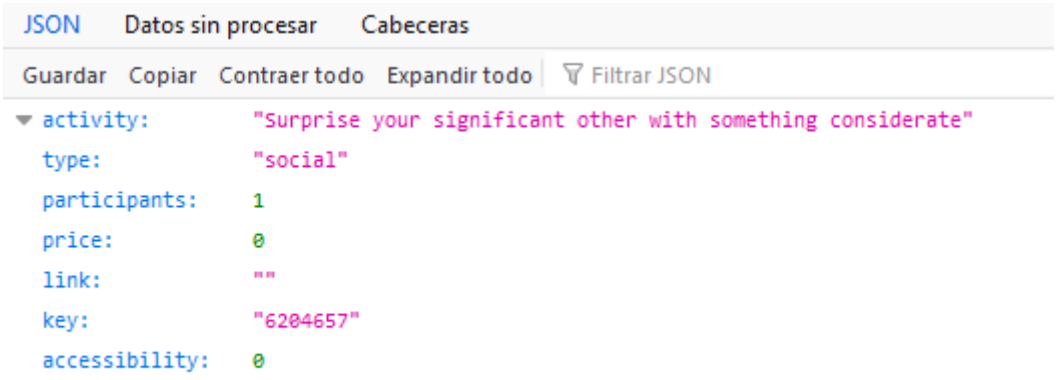
En el main ponemos los ajustes necesarios, me basé en los ajustes de la API de perros aleatorios de los ejemplo de API

```
function main() {
    var settings = {
        "async": true,
        "type": "GET",
        "url": "http://www.boredapi.com/api/activity/",
    };
    console.log(settings)

}
main()
```

Si vamos a comprobar



Si pinchamos en el enlace tendremos los datos que hemos recibido



Si actualizamos nos pondrán otros datos

Vamos a hacer que si está conectado correctamente nos envíe la actividad al html.

Gracias al ejemplo del perro me pude ayudar a hacer este.

```
function main() {
    var settings = {
        "async": true,
        "type": "GET",
        "url": "http://www.boredapi.com/api/activity/",
    };
    console.log(settings)

    $.ajax(settings).done(function (response) {
        if (settings.async == true) {
            document.getElementById("actividad").innerHTML = response.activity;

            console.log(response)
        }
    });
}
main()
```

Es importante que en el html poner este enlace

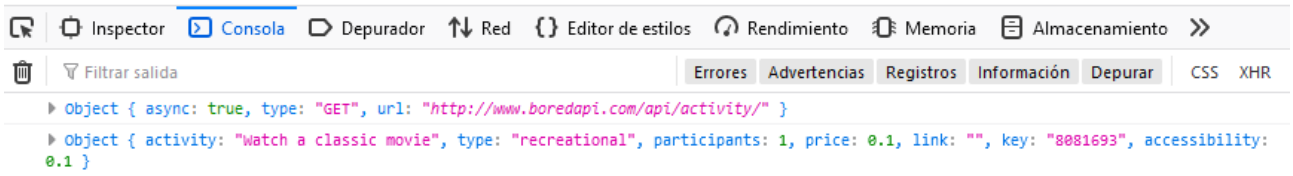
```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
```

Si no funcionará el ajax correctamente.

Vamos a comprobar que funciona

En el html nos muestra “Mirar una película clásica” y es el mismo valor que el resultado actividad mostrado en la consola

Actividad: Watch a classic movie



Ahora vamos a añadir los valores restantes

```
function main() {
    var settings = {
        "async": true,
        "type": "GET",
        "url": "http://www.boredapi.com/api/activity/",
    };
    console.log(settings)

    $.ajax(settings).done(function (response) {
        if (settings.async == true) {
            document.getElementById("actividad").innerHTML = response.activity;
            document.getElementById("accesibilidad").innerHTML = response.accessibility;
            document.getElementById("tipo").innerHTML = response.type;
            document.getElementById("participantes").innerHTML = response.participants;
            document.getElementById("precio").innerHTML = response.price;
            document.getElementById("enlace").innerHTML = response.link;
            document.getElementById("llave").innerHTML = response.key;
            console.log(response)
        }
    });
}
main()
```

El html tendría esta forma:

```
<!DOCTYPE html>
<html>
<meta charset="UTF-8">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<body>
    Actividad: <label id="actividad"></label><br>
    Accesibilidad: <label id="accesibilidad"></label><br>
    Tipo: <label id="tipo"></label><br>
    Participantes: <label id="participantes"></label><br>
    Precio <label id="precio"></label><br>
    Enlace <label id="enlace"></label><br>
    Llave <label id="llave"></label><br>
    <div>

    </div>
</body>
<footer>
    <script src="main.js"></script>
</footer>

</html>
```

Como quedará una vez finalizado:

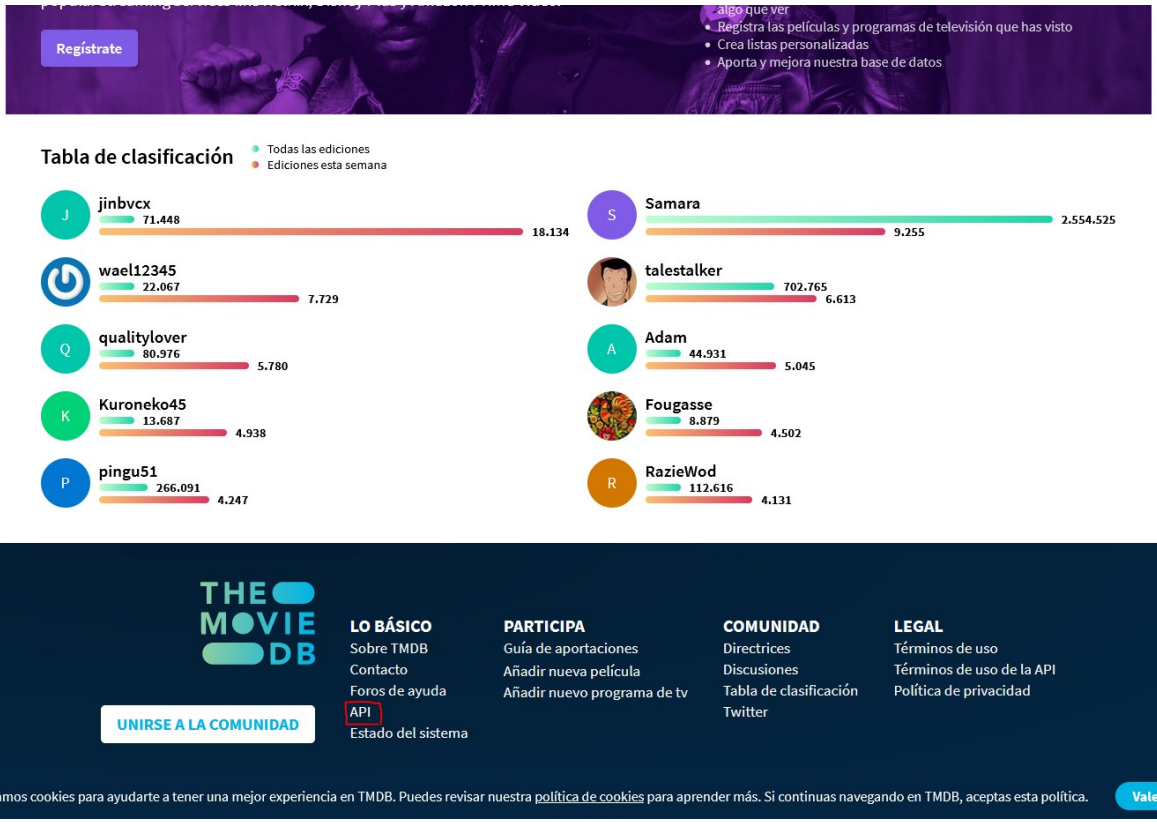
Actividad: Learn and play a new card game
Accesibilidad: 0
Tipo: recreational
Participantes: 1
Precio 0
Enlace https://www.pagat.com
Llave 9660022

Aveces ciertas actividades tienen enlace, pero casi siempre suele estar vacio

API con Clave:

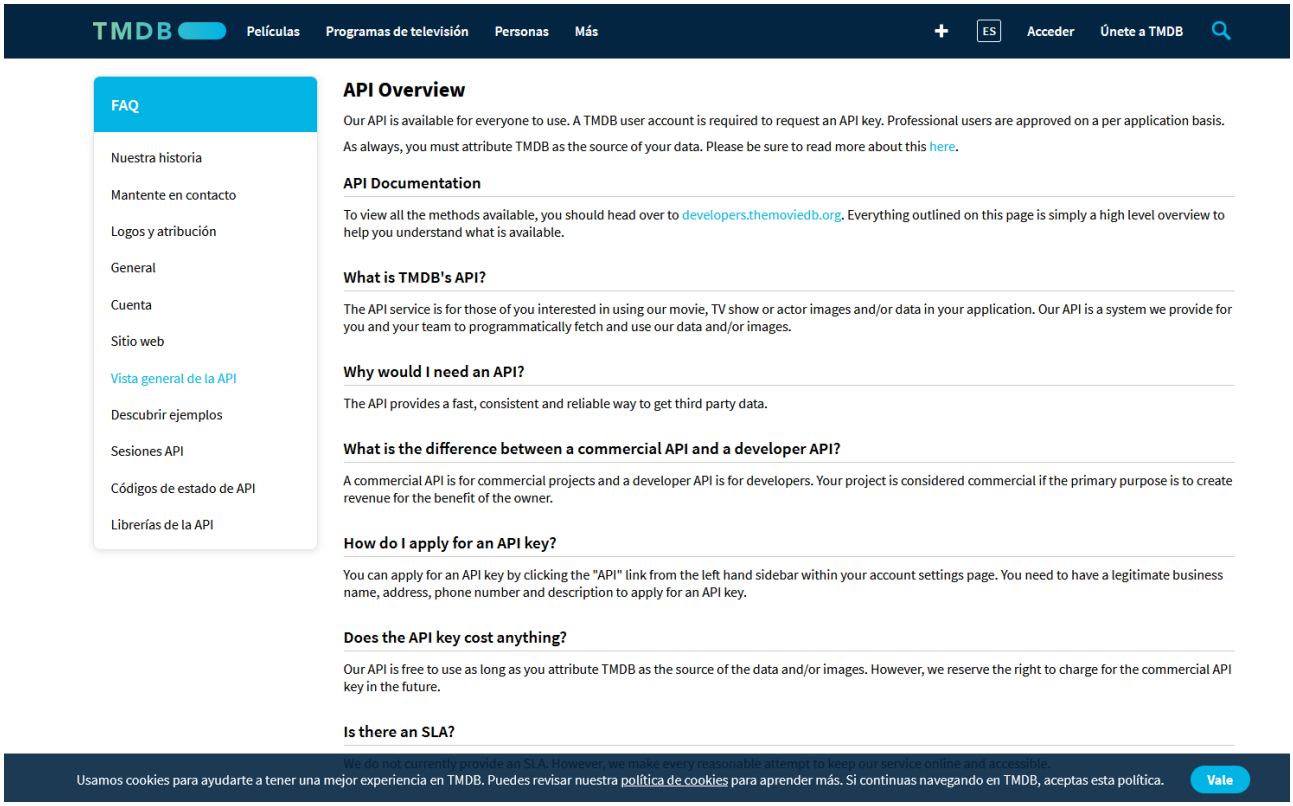
Para la api con clave usaré la API de The Movie Database(TMDB)
Es una plataforma conocida por tener una cartelera de películas y series.
Esta plataforma solo nos permite ver la información de dicha película o serie, como que actores son los protagonistas, un tráiler, en que plataforma se podría llegar a ver dicha serie o película si estuviéramos interesados, aparte de otra información.

En está página para encontrar la api podremos bajar hasta el final de la página o mediante este [enlace]



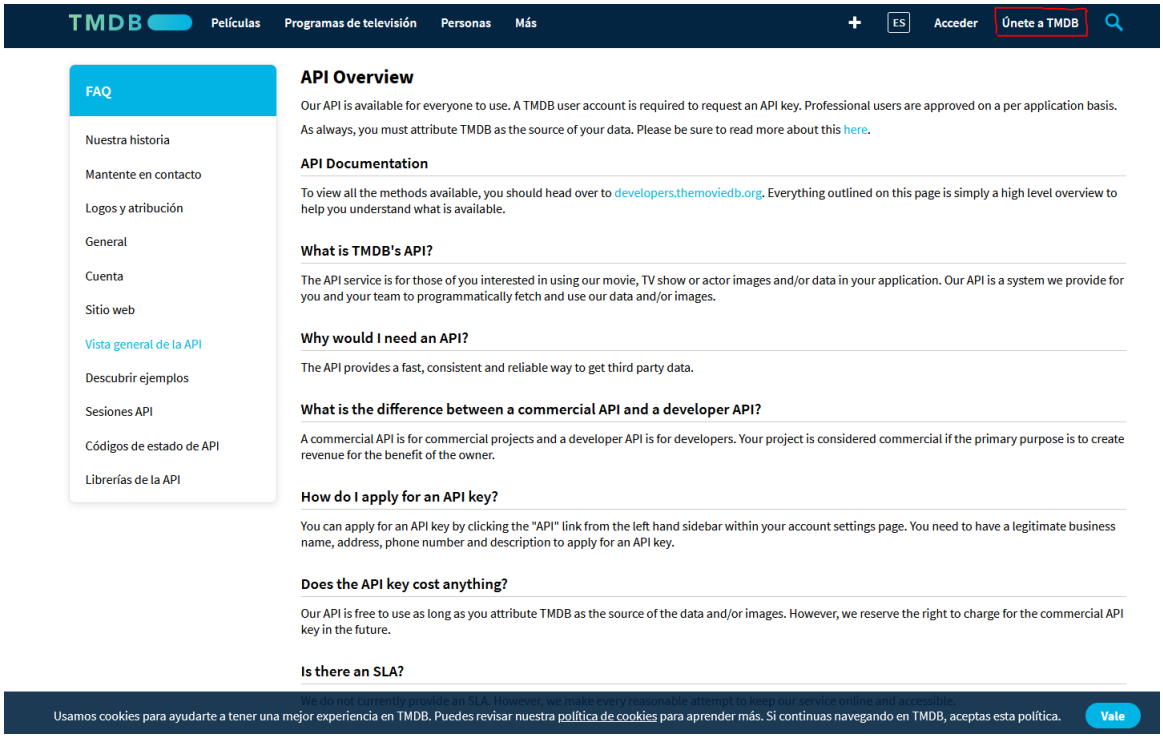
Mediante este enlace podremos saber la información de la API que vamos a usar, como que vamos a necesitar, si dicha API tenemos que hacer algún aporte económico o no etc.

Información de la API

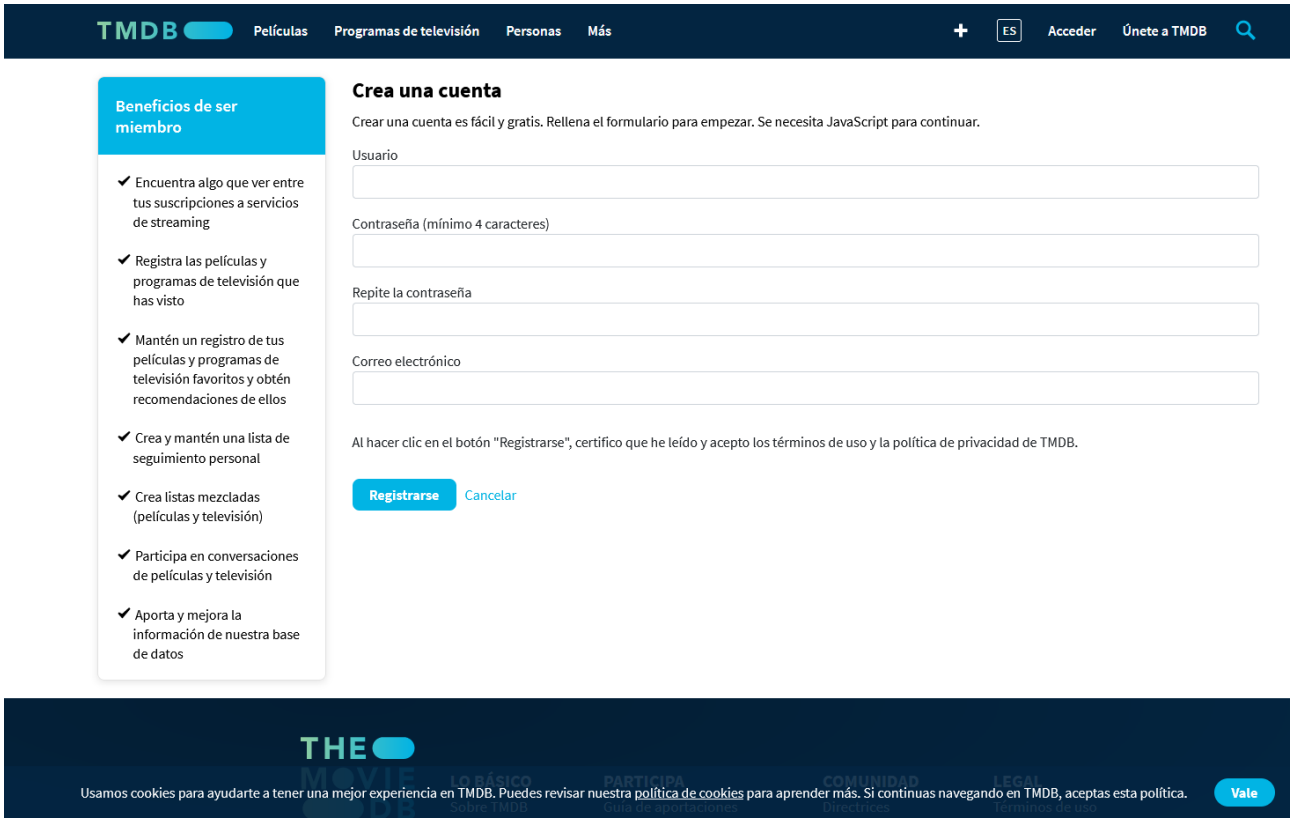


Para poder empezar usar esta API necesitaremos una cuenta

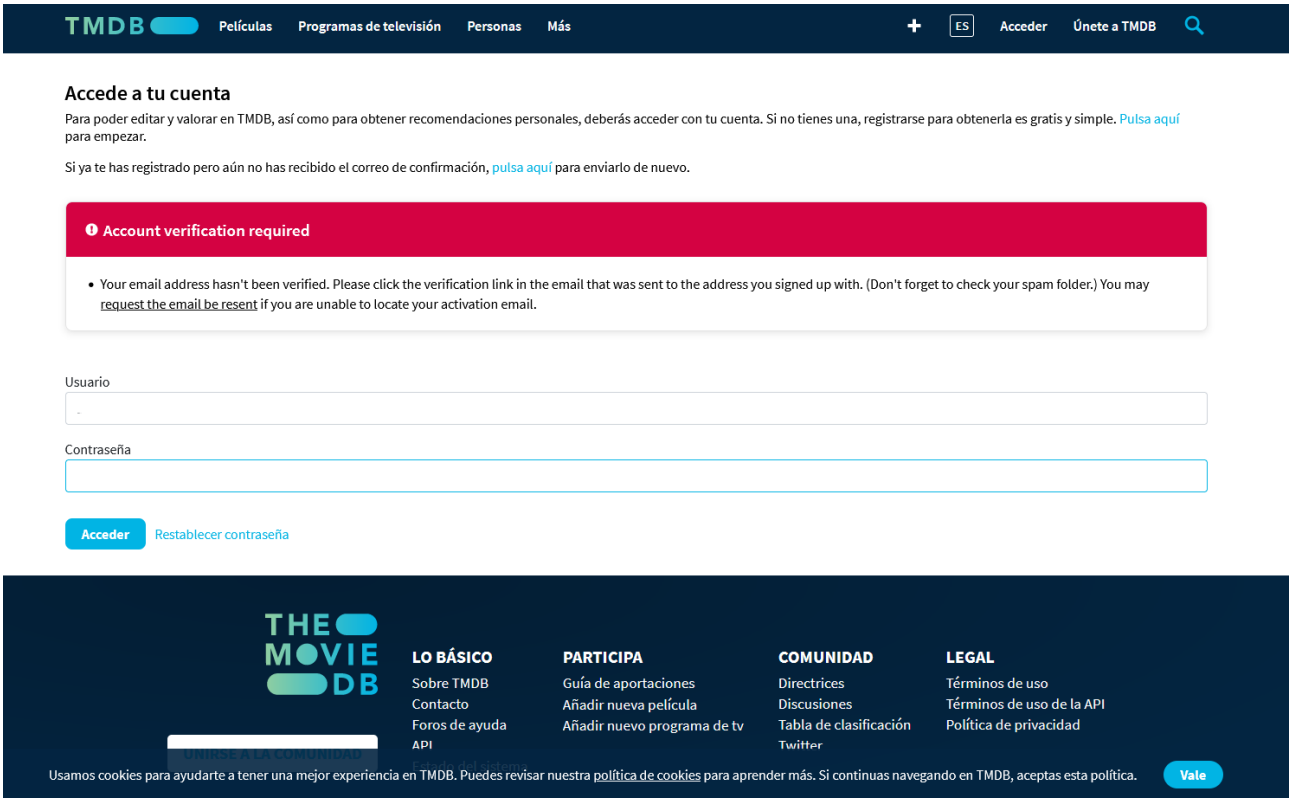
Para ello vamos a la parte superior derecha y le daremos a crear cuenta, a menos que ya tengamos una creada, que si fuera el caso solo iniciaremos sesión.



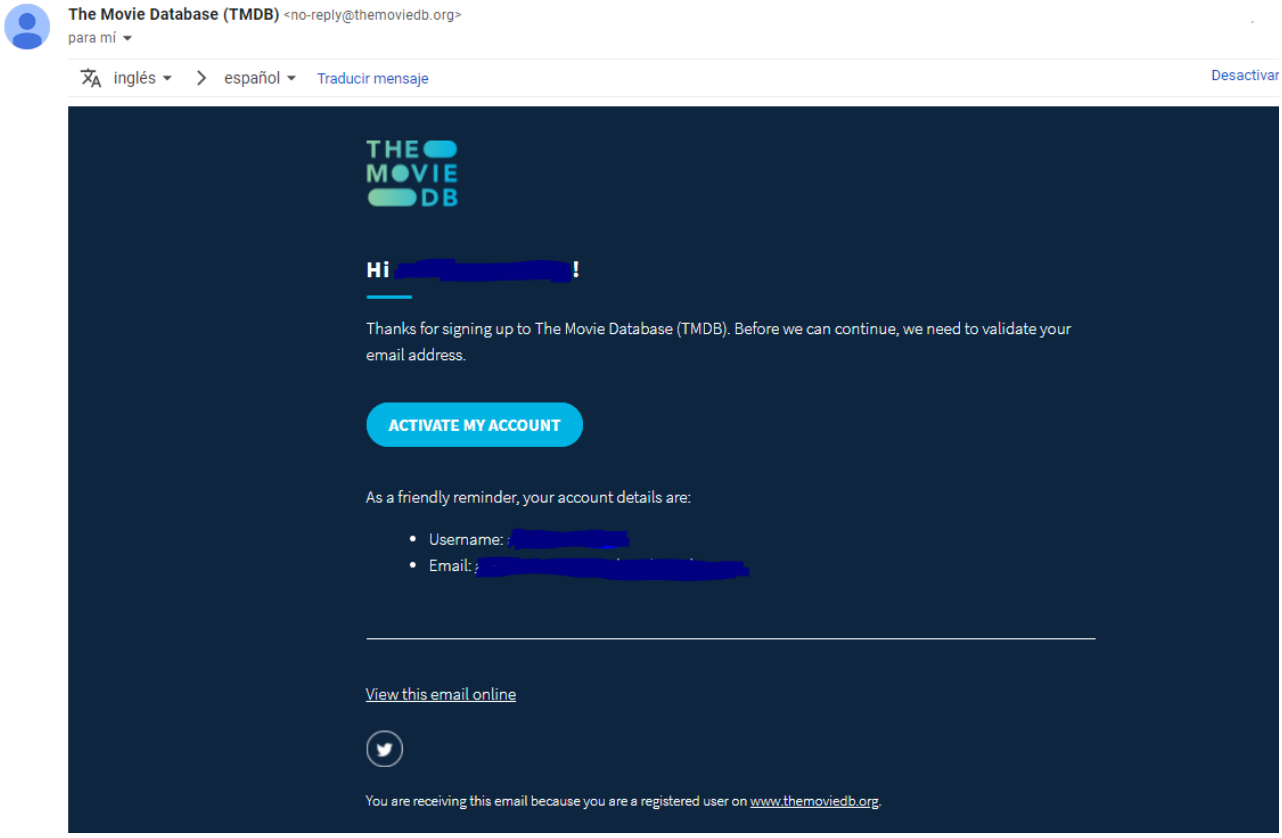
Pondremos los datos necesarios para crear la cuenta



Nos registramos, y tendremos que comprobar nuestro correo electrónico

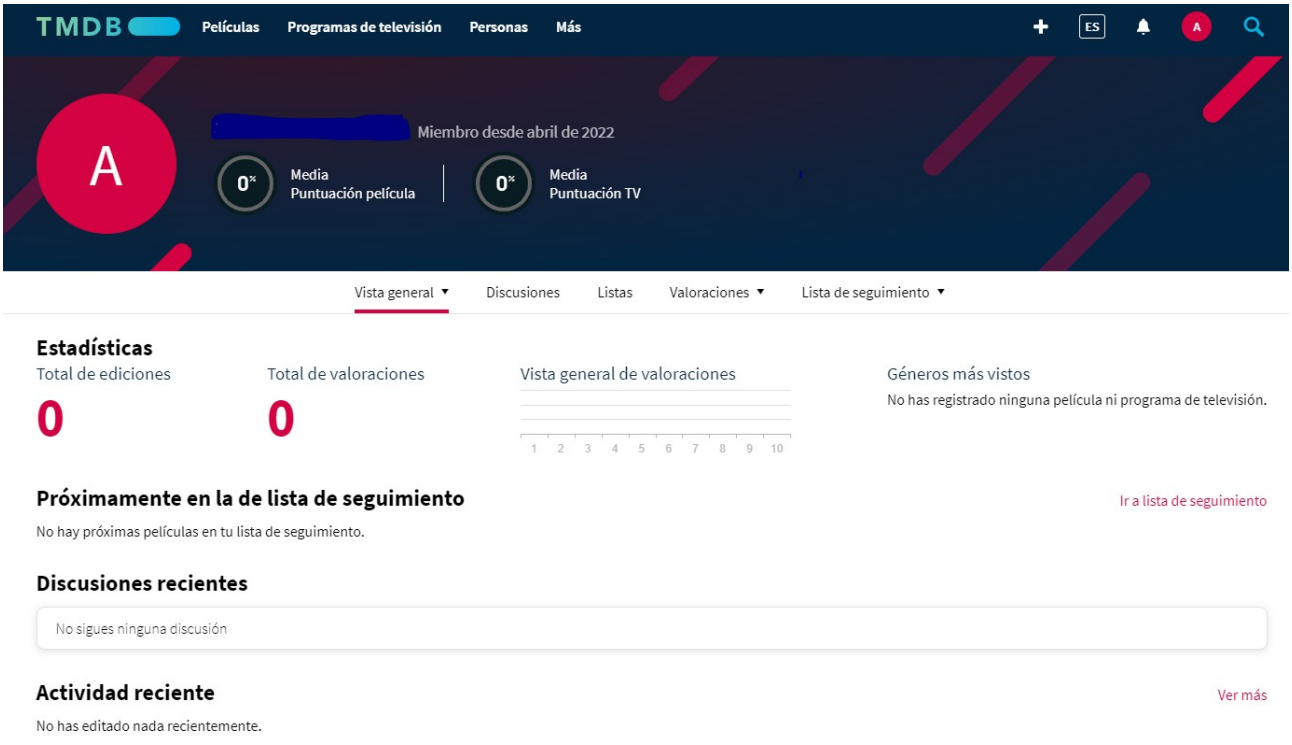


En nuestro correo nos habrá llegado esta notificación, que nos permitirá activar la cuenta



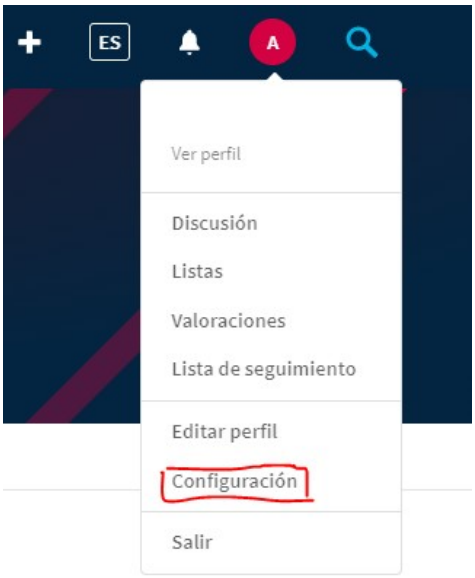
Una vez que activemos la cuenta ya estaría creada correctamente.

Para ello comprobamos que estamos iniciados



Ahora vamos a la parte superior derecha, donde está nuestra foto de perfil.

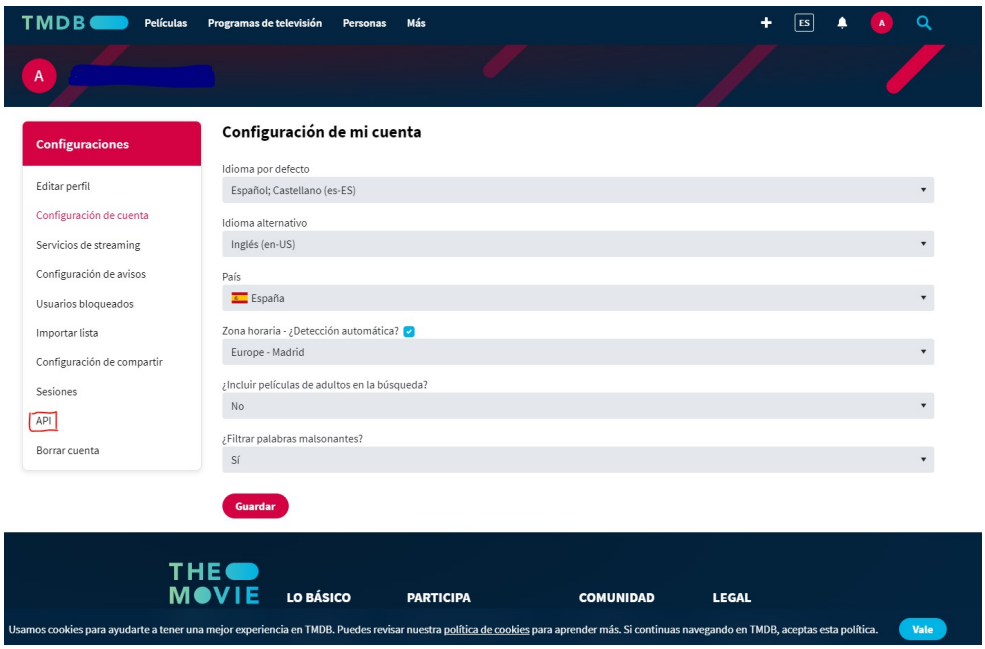
Pinchamos en la foto, se desplegará un menú, y buscamos la opción de ajustes o configuración.



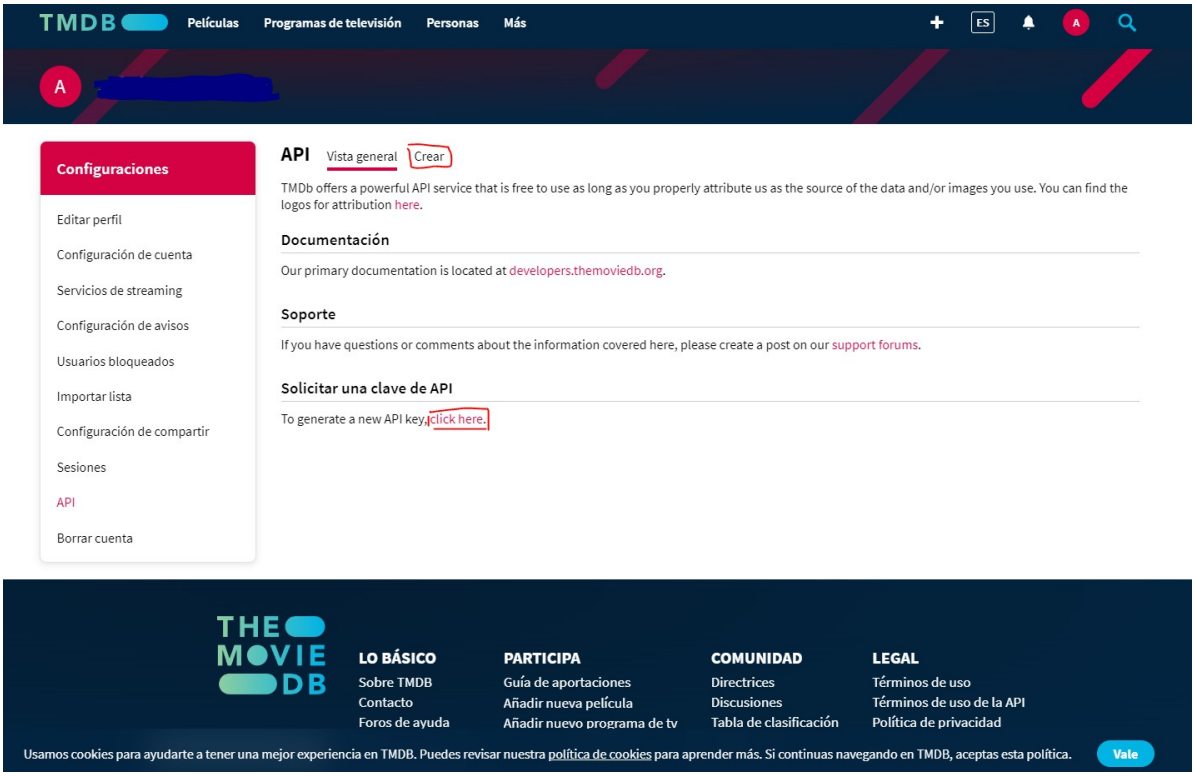
Una vez entremos nos aparecerá ajustes de la cuenta.

En el lado izquierdo inferior encontraremos una opción que ponga ***API***

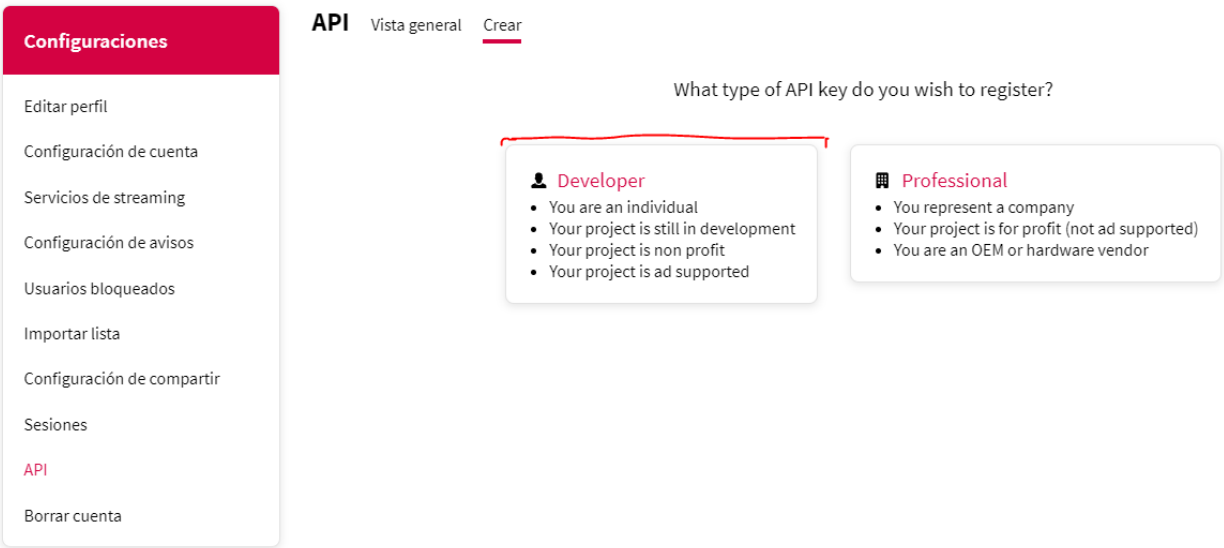
Le pinchamos



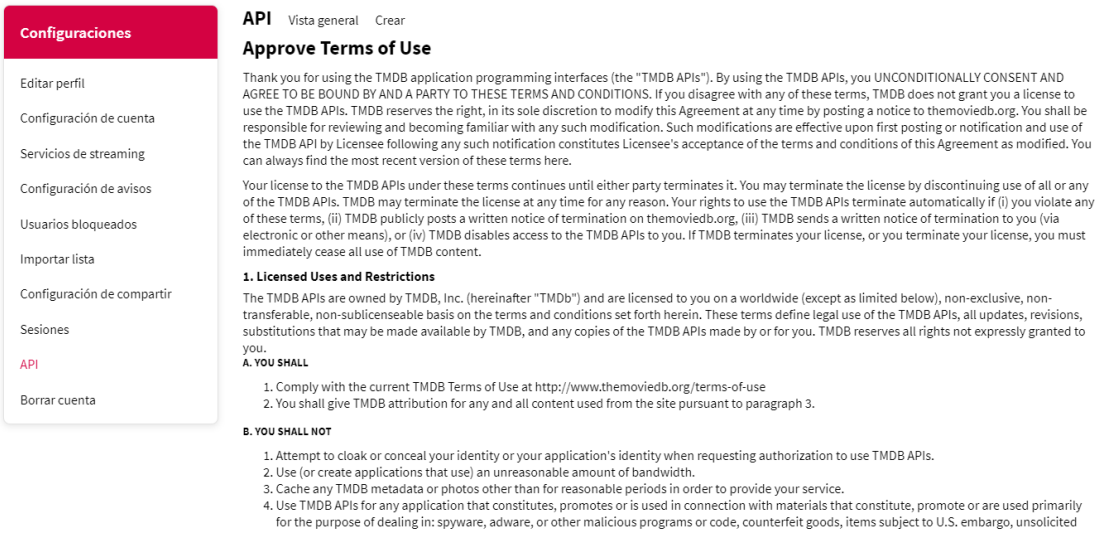
Una vez esteamos en la pestaña de API tendremos que solicitar nuestra clave, para ello podemos darle a ***crear*** o donde pone ***click here*** *(Llevan al mismo sitio)*



Como la API la vamos a usar para uso personal sin tener ninguna ganancia económica escogeremos la opción de developer. En caso contrario la de profesional

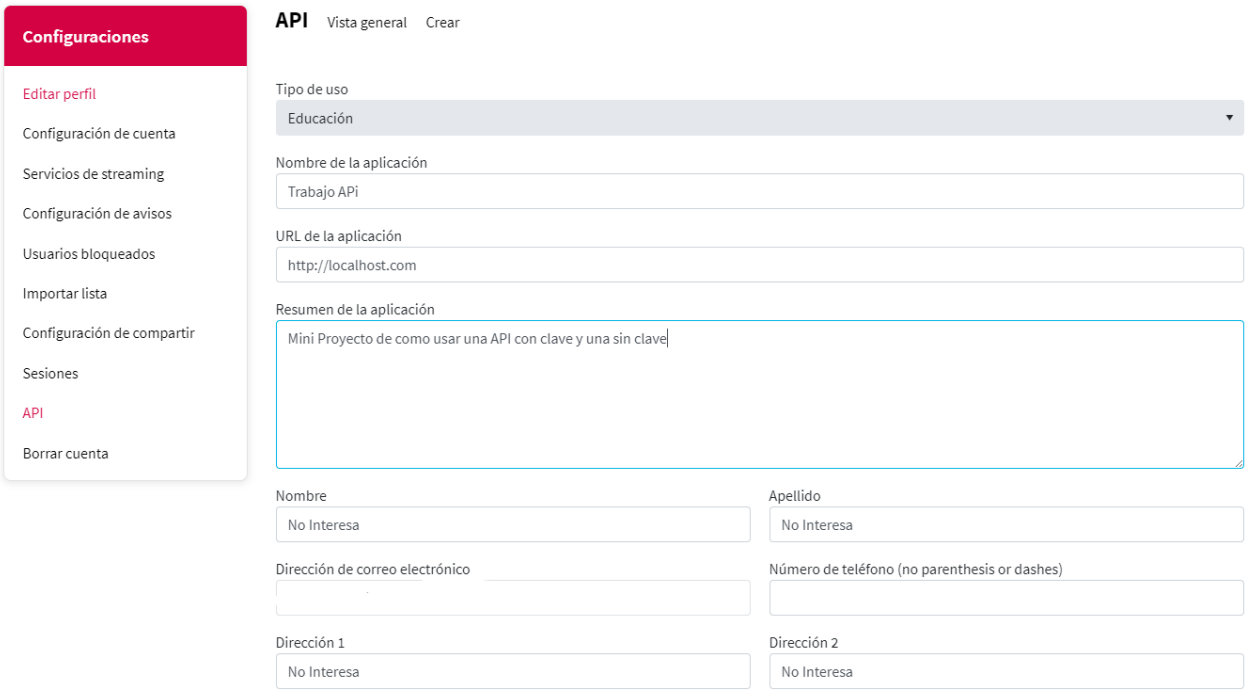


Una vez que nos escojamos la opción que queramos nos aparecerán unos términos y condiciones.



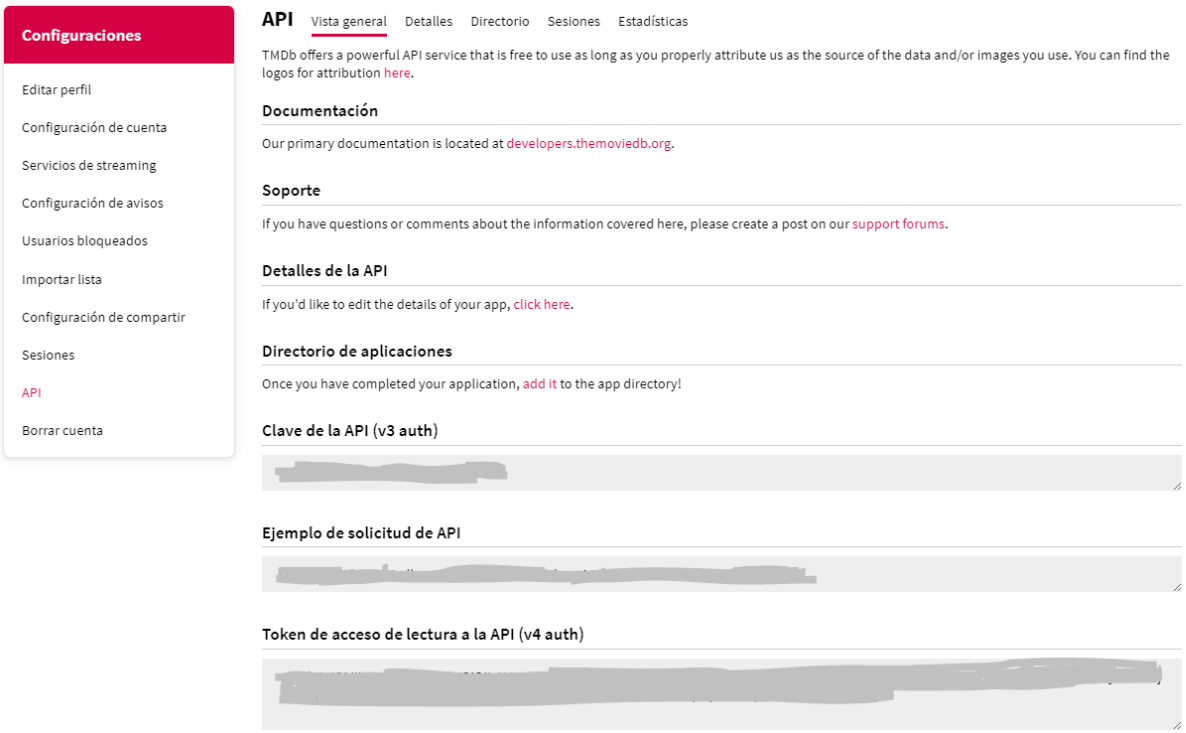
Si no se quieren leer bajamos hasta abajo y pulsamos aceptar

Nos aparecerá un pequeño formulario, lo más importante a rellenar sería la selección, el nombre de la aplicación,la url y el resumen



Si no queremos poner todos los datos en el formulario, tales como el nombre o teléfono, podemos rellenarlos con *(No Interesa, 0000,…)*

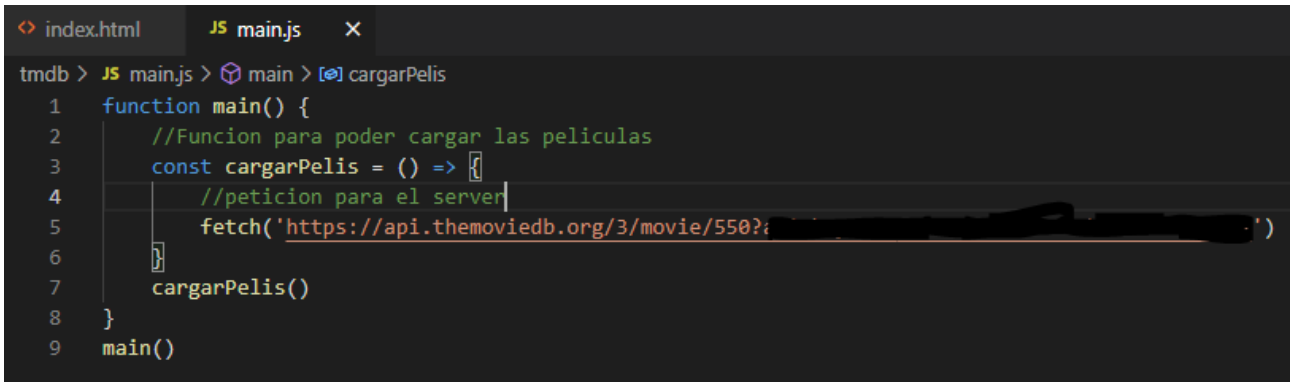
Una vez finalizado el formulario esperamos o tendremos automáticamente la clave. Hay 2 tipos de clave, una versión v3.0 & v4.0



Código:

Crearemos un index.html y un main.js.

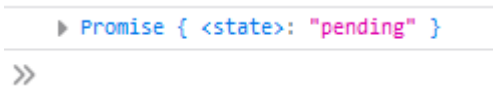
En el main crearemos un funcion que nos permita recoger las peliculas, mandarle una solicitud al server.



Si queremos saber si está funcionando

Declaramos una variable y la igualamos, la sacamos por consola.

Vamos al navegador y comprobamos

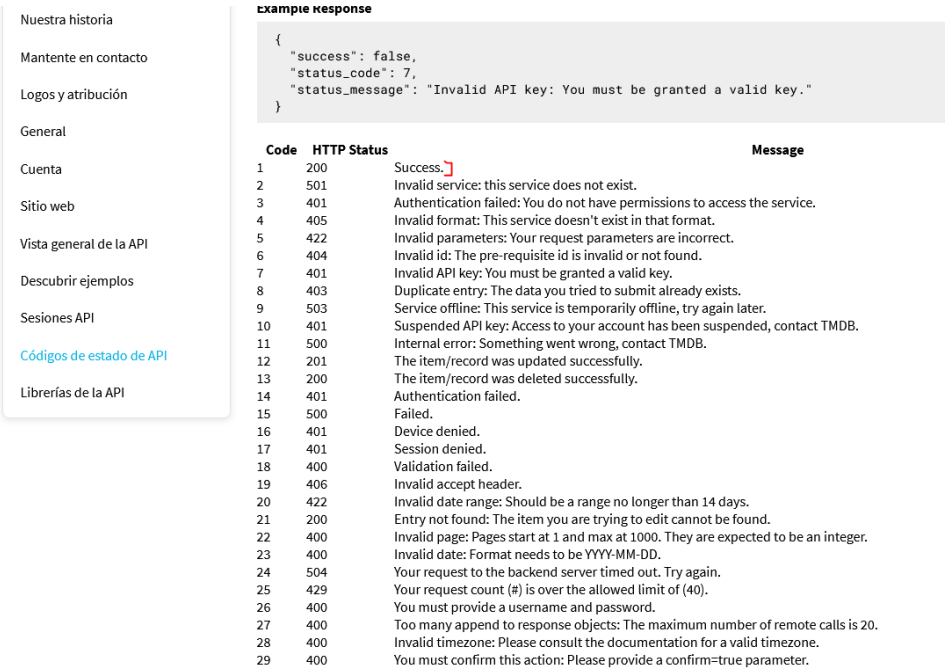


Pero esto es una respuesta no correcta, estamos esperando una respuesta del servidor. Ya que el servidor tiene que procesar la petición, y este tipo de peticiones suele tardar tenemos que hacer que dicha petición se quede ahí y cuando finalice dicha petición pase a la siguiente linea.

Comprobación de la petición

Response { type: "cors", url: "https://api.themoviedb.org/3/movie/550?", redirected: false, status: 200, ok: true, statusText: "OK", headers: Headers, body: ReadableStream, bodyUsed: false }

Uno de los datos más importantes será el status. Dependiendo de que número tengamos tendremos éxito con la conexión o fail, y dependiendo de que número tengamos puede ser uno distintos. Por si fallamos para comprobar en este enlace



Cuando hacemos dicha petición al servidor no es inmediata como hemos comentado previamente, esto puede generar errores. Para ello pondremos un try y catch.Para saber si falla. *Explicación mas detallada*

```
function main() {
    //Funcion para poder cargar las peliculas
    const cargarPelis = async () => {

        try {
            //Peticion para el server
            respuestaServer = await fetch('https://api.themoviedb.org/3/movie/550?[redacted]')
            console.log(respuestaServer)
        } catch (error) {
            console.log("Error: " + error)
        }
    }
    cargarPelis()

}
main()
```

Para acceder a la información que nos devolvió la petición usaremos json, (json es asíncrono)

```
function main() {
    //Funcion para poder cargar las peliculas
    const cargarPelis = async () => {

        try {
            //Peticion para el server
            respuestaServer = await fetch('https://api.themoviedb.org/3/movie/550?a
            console.log(respuestaServer)
            const datos = await respuestaServer.json()
            console.log(datos)
        } catch (error) {
            console.log("Error: " + error)
        }
    }
    cargarPelis()

}
main()
```

Nos devuelve un objecto con propiedades, tales como si dicha película es para adultos, el nombre del titulo, la foto de la pelicula

Si bien solo queremos acceder a cierta información de la película tendremos que poner la variable que queremos buscar, en este caso será el titulo:

```
function main() {
    //Funcion para poder cargar las peliculas
    const cargarPelis = async () => {

        try {
            //Peticion para el server
            respuestaServer = await fetch('https://api.themoviedb.org/3/movie/550?a                    ')
            console.log(respuestaServer)
            const datos = await respuestaServer.json()
            console.log(datos.title)
        } catch (error) {
            console.log("Error: " + error)
        }
    }
    cargarPelis()

}
main()
```

Solamente nos devuelve el nombre del titulo

```
▸ Response {
Fight Club
```

Ahora bien, que pasa si en la url está mal, falla y no encuentra ninguna película, vamos a forzar que falle, para eso ponemos una letra antes del 550

```
function main() {
    //Funcion para poder cargar las peliculas
    const cargarPelis = async () => {

        try {
            //Peticion para el server
            respuestaServer = await fetch('https://api.themoviedb.org/3/movie/a550[redacted]')
            console.log(respuestaServer)
            const datos = await respuestaServer.json()
            console.log(datos.title)
        } catch (error) {
            console.log("Error: " + error)
        }
    }
    cargarPelis()

}
main()
```

Nos dirá que no se puede conseguir la información y la el titulo que intentamos conseguir nos sale indefinido

```
XHR GET https://api.themoviedb.org/3/movie/a550?ap[redacted]
▶ Response { type: "cors", url: "https://api.themoviedb.org/3/movie/a550?api_key=14db5f47dbb7[redacted]
undefined
```

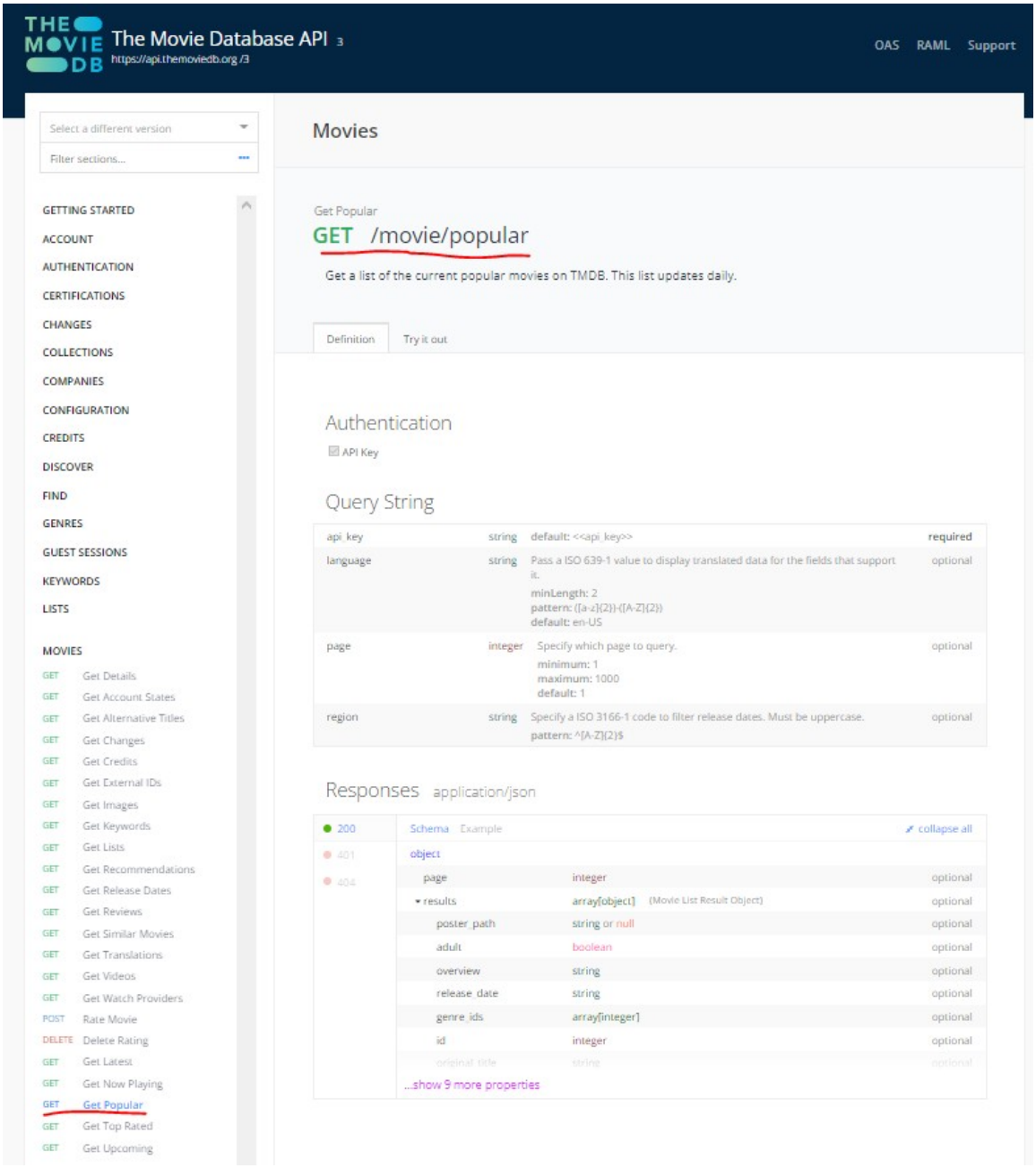
Para solucionar y tener información de donde puede estar el error en el código pondremos varias comprobaciones de estado con el server

```
function main() {
    //Funcion para poder cargar las peliculas
    const cargarPelis = async () => {

        try {
            //Peticion para el server
            respuestaServer = await fetch('https://api.themoviedb.org/3/movie/a550?[redacted]')
            console.log(respuestaServer)

            //Si se conecta correctamente
            if (respuestaServer.status === 200) {
                const datos = await respuestaServer.json()
                console.log(datos.title)
            } else if (respuestaServer.status === 401) {
                console.log("Error en url")
            } else if (respuestaServer.status === 404) {
                console.log("No existe la pelicula")
            } else {
                console.log("Error")
            }

        } catch (error) {
            console.log("Error: " + error)
        }
    }
    cargarPelis()

}
main()
```

Ahora comprobaremos como saca la información, ahora nos dice que no existe la pelicula en vez de indefinido

```
XHR GET https://api.themoviedb.org/3/movie/a550?
▶ Response { type: "cors", url: "https://api.themoviedb.org/3/movie/a550?api_key=14db5f47dbb79
No existe la pelicula
```

Con esto que llevamos echo nos saca una película en específica, ahora bien, queremos sacar varias películas, digamos las películas mas populares. Para ello volvemos a la página de desarrollo de tmdb.



Cuando estemos por la página de desarrollo buscaremos películas, por popular. Si nos fijamos el get sería /movie/popular.

Tendremos que cambiar la url en el código, sustituyendo el 550 por popular y quitar el title de datos, ya que no queremos solo el titulo

```
function main() {
    //Funcion para poder cargar las peliculas
    const cargarPelis = async () => {

        try {
            //Peticion para el server
            respuestaServer = await fetch('https://api.themoviedb.org/3/movie/popular?[redacted]')
            console.log(respuestaServer)

            //Si se conecta correctamente
            if (respuestaServer.status === 200) {
                const datos = await respuestaServer.json()
                console.log(datos)
            } else if (respuestaServer.status === 401) {
                console.log("Error en url")
            } else if (respuestaServer.status === 404) {
                console.log("No existe la pelicula")
            } else {
                console.log("Error")
            }

        } catch (error) {
            console.log("Error: " + error)
        }
    }
    cargarPelis()

}
main()
```

Si comprobamos, ya nos da mas que un resultado, si mostramos más los datos podremos ver la información como anteriormente sabíamos, si dicha película es para mayores o no, el nombre de la película ...

```
▸ Response { type: "cors", url: "https://api.themoviedb.org/3/movie/popular?c[redacted]" ,
▸ Object { page: 1, results: (20) [_], total_pages: 33215, total_results: 66429 }
```

Responses application/json

	Schema	Example	collapse all
200	object		
401	page	integer	optional
404	results	array[object] {Movie List Result Object}	optional
	poster_path	string or null	optional
	adult	boolean	optional
	overview	string	optional
	release_date	string	optional
	genre_ids	array[integer]	optional
	id	integer	optional
	original_title	string	optional

...show 9 more properties

Esto serían las respuestas del servidor. Las que hemos echo conforme si fallaba

Vamos paso por paso, queremos que nos muestre que por cada película nos muestre un dato determinado.

Para ello tendremos que hacer un for each.

```
function main() {
    //Funcion para poder cargar las peliculas
    const cargarPelis = async () => {

        try {
            //Peticion para el server
            respuestaServer = await fetch('https://api.themoviedb.org/3/movie/popular?api_[redacted]')
            console.log(respuestaServer)

            //Si se conecta correctamente
            if (respuestaServer.status === 200) {
                const datos = await respuestaServer.json()
                datos.results.forEach(pelicula => {
                    console.log(pelicula.title)
                });
            } else if (respuestaServer.status === 401) {
                console.log("Error en url")
            } else if (respuestaServer.status === 404) {
                console.log("No existe la pelicula")
            } else {
                console.log("Error")
            }

        } catch (error) {
            console.log("Error: " + error)
        }
    }
    cargarPelis()

}
main()
```

Si comprobamos, nos mostrará solamente el título, ya que hemos querido nosotros

```
▶ Response { type: "cors", url: "
Spider-Man: No Way Home
The Batman
War of the Worlds: Annihilation
Sonic the Hedgehog 2
Turning Red
Moonfall
```

Ahora que tenemos los títulos, vamos a mostrarlos en el html, en vez de la consola.
Para ello crearemos un variable donde sea vacía. Dicha variable hacemos que sea igual a una etiqueta con el dato que queremos de la película, en nuestro caso mostraremos el título.
Y cogeremos el id en de la etiqueta que tengamos en el html que nos permitirá colocar la información.

```
if (respuestaServer.status === 200) {
    const datos = await respuestaServer.json()

    let peliculas = ''
    datos.results.forEach(pelicula => {
        peliculas += `<h1>${pelicula.title}</h1>`
    });

    document.getElementById("contain").innerHTML = peliculas
} else if (respuestaServer.status === 401) {
    console.log("Error en url")
} else if (respuestaServer.status === 404) {
    console.log("No existe la pelicula")
} else {
    console.log("Error")
}
```

De esta manera conseguimos mostrar los títulos

All the Old Knives

Black Crab

Pil's Adventures

Fantastic Beasts: The Secrets of Dumbledore

No Exit

The Grandmother

The In Between

Gold

Sonic the Hedgehog

Yaksha: Ruthless Operations

The Ice Age Adventures of Buck Wild

Ahora pondremos la imagen correspondiente de cada película
Para saber como recoger la imagen, cuando le pedimos la información al servidor esta se guarda en el poster_path, lo podemos ver en:

Responses application/json

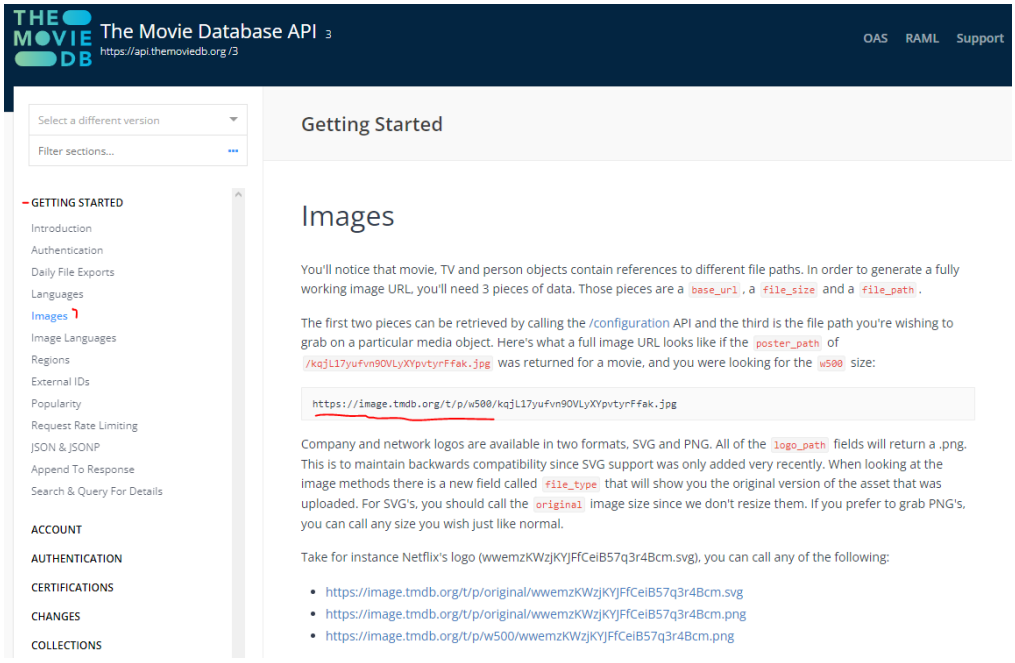
	Schema Example	collapse all	
200	object		
401	page	integer	optional
404	results	array[object] {Movie List Result Object}	optional
	poster_path	string or null	optional
	adult	boolean	optional
	overview	string	optional
	release_date	string	optional
	genre_ids	array[integer]	optional
	id	integer	optional
	original_title	string	optional

...show 9 more properties

```
original_title: "Spider-Man: No Way Home"
overview: "Peter Parker is unmasked and no longe
popularity: 9695.89
poster_path: "/1g0dhYtq4irTY1GPXvft6k4YLjm.jpg"
release_date: "2021-12-15"
title: "Spider-Man: No Way Home"
```

Ahora que sabemos como recoger de una manera la imagen, ahora vamos a ver como mostrarla. Para ello vamos a buscar información, en empezando, imágenes.

Ahí nos dice que las imágenes se mostrarán con una url. Si nos fijamos en dicha url (la de ejemplo) a partir de w500 tiene puesto el poster path



Nosotros para poner el poster path con un url lo haremos de la siguiente manera. Pondremos dicha url y le sumaremos la viable pelicula y su poster path correspondiente, como antes hacíamos con el titulo

```
if (respuestaServer.status === 200) {
    const datos = await respuestaServer.json()

    let peliculas = '';
    datos.results.forEach(pelicula => {
        peliculas += `
        <div class="pelicula">
            <img class="poster" src="https://image.tmdb.org/t/p/w500/${pelicula.poster_path}">
            <h3 class="titulo">${pelicula.title}</h3>
        </div>
        `;

    });

    document.getElementById("contenedor").innerHTML = peliculas
```

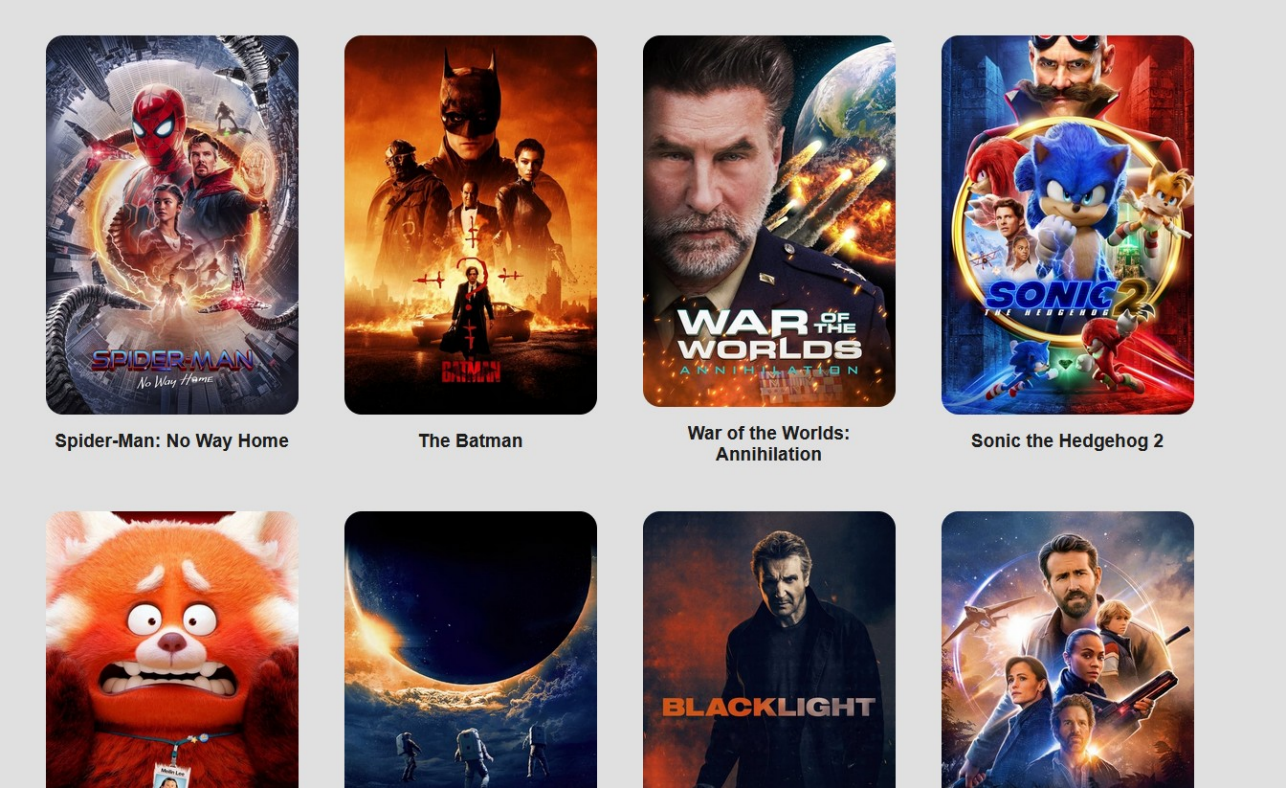
Guardamos y comprobamos

Spider-Man: No Way Home

Ahora nos muestra el cartel de la película mas el titulo de dicha película

Ahora si queremos para que no quede tan feo, podemos añadirle un poco de css.

Como hacer más pequeña la foto etc...



Ahora es más fácil la visualización para ver las películas más populares,solamente podemos ver las 20 primeras.

Si queremos ver más películas populares tendremos que hacer unos botones para pasar a la siguiente página y uno para volver a la anterior

Una vez creado los botones pasaremos a crear un evento de pulsación.

Primero haremos el de siguiente, para ello creamos el evento de escuchar cuando se hace el click le suma uno a la página y cargará la función que recoge las películas

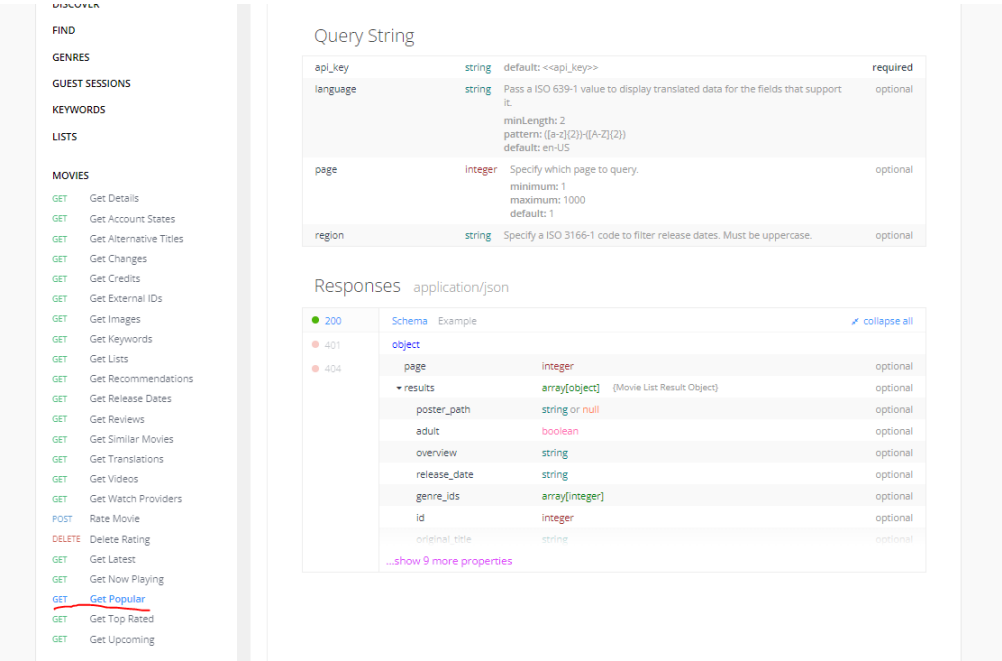
```
let pagina = 1;
const Anterior = document.getElementById('BotonAnterior');
const Siguiente = document.getElementById('BotonSiguiente');

Siguiente.addEventListener('click', () => {
    pagina += 1
    cargarPelis()
});
```

Si intentamos probarlo no va a funcionar, tenemos que añadir en la url de nuestra api en que página estamos.

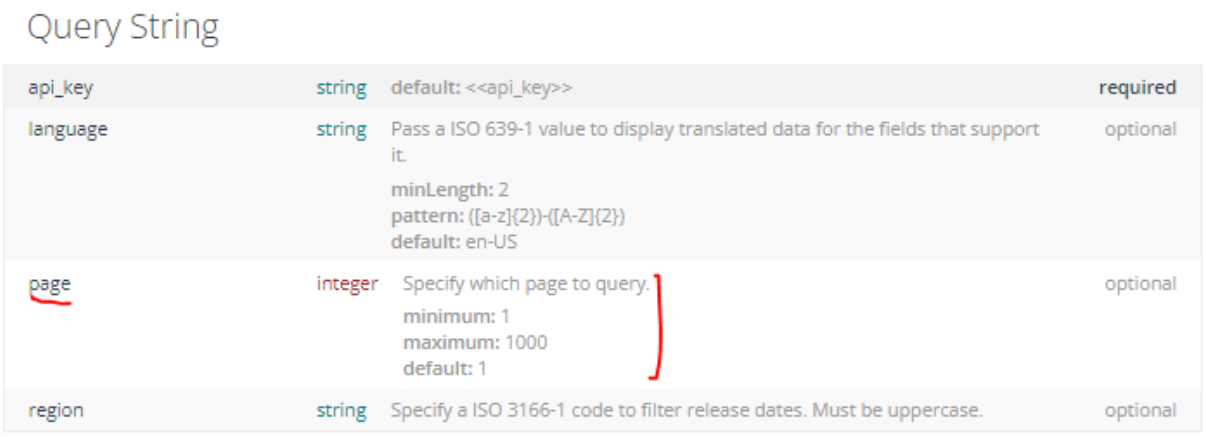
```
//Peticion para el server
const respuestaServer = await fetch(`https://api.themoviedb.org/3/movie/popular?api_[redacted]&page=${pagina}`)
console.log(respuestaServer)
```

Cuando este el final de la key añadimos que la página es dicha varible que hemos escrito anteriormente. Tenemos que cambiar las comillas por backtips para que funcione correctamente.

Para saber lo de las páginas y poder entender tenemos que ir al apartador de películas, películas populares



Donde buscaremos el apartado de página



La página cuando cargaba de base cargaba el 1, tiene un tamaño máximo de 1000, esto nos permitirá movernos 1000 veces y cada página tener 20 películas.
Tenemos que solucionar que si un usuario no pueda pasar más de las 1000, ya que sería el tope máximo que se puede alcanzar, para ello lo solucionamos con un if

```
Siguiente.addEventListener('click', () => {
    if (pagina < 1000) {
        pagina += 1
        cargarPelis()
    }
});
```

Ahora para visitar la página anterior sería lo mismo que la siguiente pero negativamente

```
Anterior.addEventListener('click', () => {
    if (pagina > 1) {
        pagina -= 1;
        cargarPelis();
    }
});
```

De tal manera que este sería nuestro resultado: Ver

Bibliografía:

- Lista de Apis
- Bored Api
- Bored Api – Documentación
- API TMDB
- Develop TMDB