

BRAÑOS RODRÍGUEZ ADRIÁN

Índice:

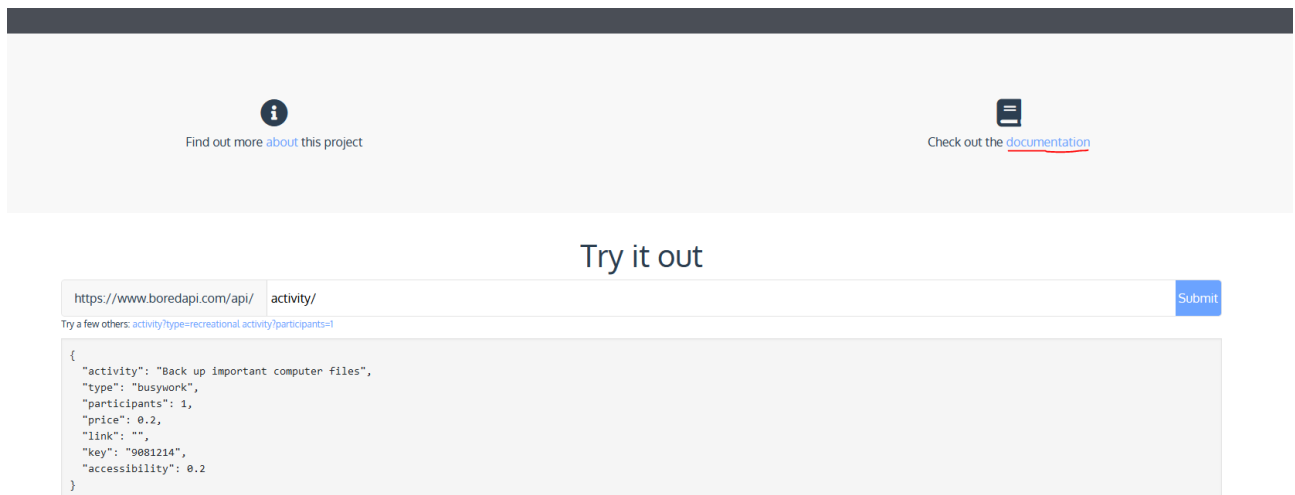
API sin Key	3
API con Key	8
Bibliografía	29

API sin clave

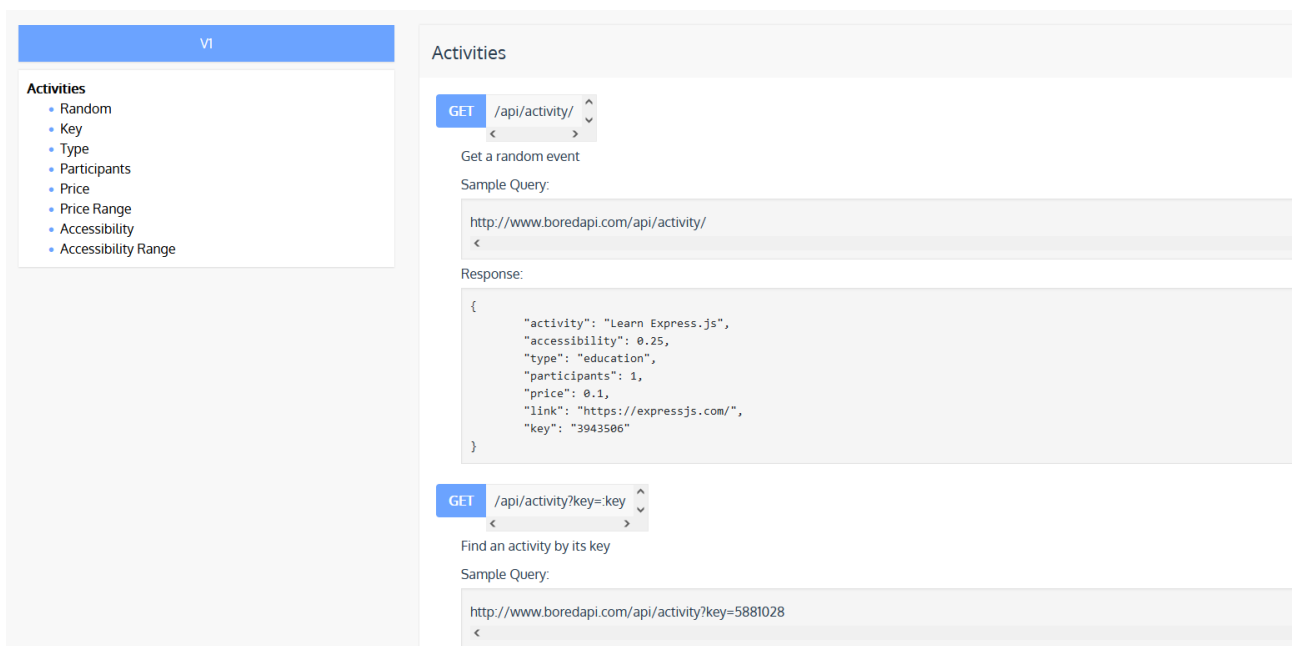
Para la api sin clave escogí la API de [Bored](#)

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

```
► Object { async: true, type: "GET", url: "http://www.boredapi.com/api/activity/" }
```

Si pinchamos en el enlace tendremos los datos que hemos recibido

JSON	Datos sin procesar	Cabeceras
Guardar	Copiar	Contraer todo Expandir todo
		Filtrar JSON
▼ activity:	"Surprise your significant other with something considerate"	
type:	"social"	
participants:	1	
price:	0	
link:	""	
key:	"6204657"	
accessibility:	0	

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

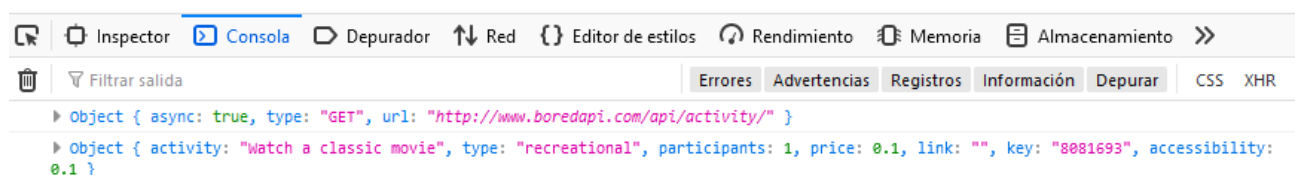
```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
```

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



A veces ciertas actividades tienen enlace, pero casi siempre suele estar vacío

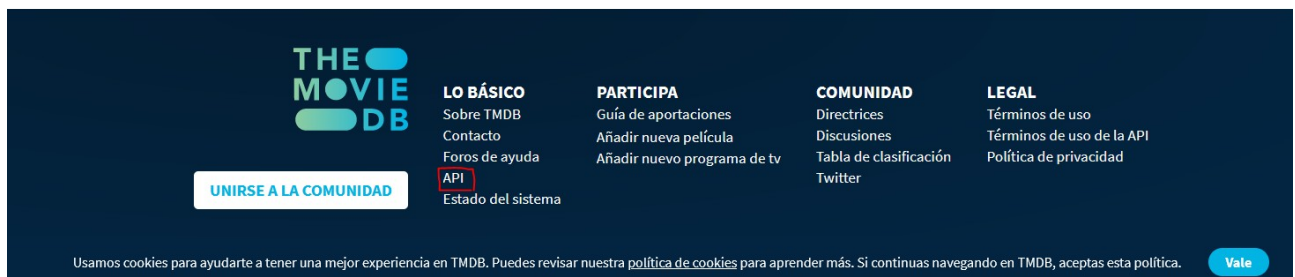
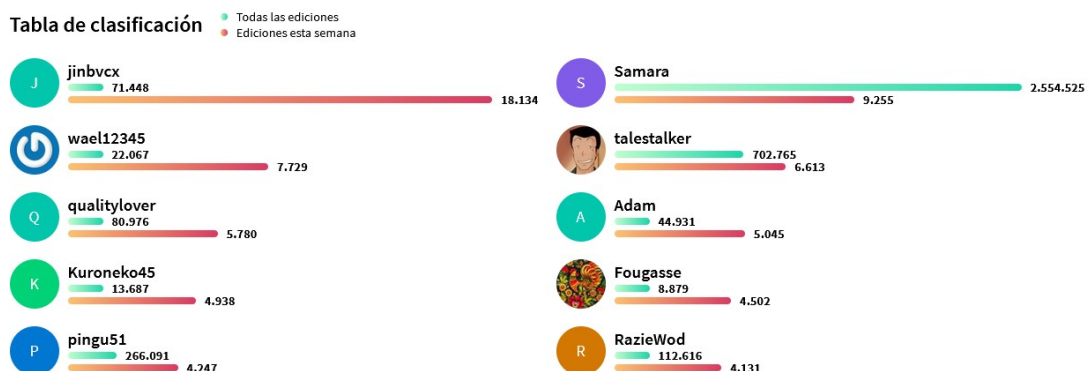
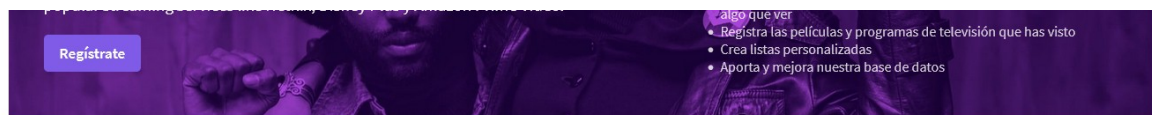
API con Clave:

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

FAQ

- Nuestra historia
- Mantente en contacto
- Logos y atribución
- General
- Cuenta
- Sitio web
- Vista general de la API**
- Descubrir ejemplos
- Sesiones API
- Códigos de estado de API
- Librerías de la API

API Overview

Our API is available for everyone to use. A TMDB user account is required to request an API key. Professional users are approved on a per application basis. As always, you must attribute TMDB as the source of your data. Please be sure to read more about this [here](#).

API Documentation

To view all the methods available, you should head over to developers.themoviedb.org. Everything outlined on this page is simply a high level overview to help you understand what is available.

What is TMDB's API?

The API service is for those of you interested in using our movie, TV show or actor images and/or data in your application. Our API is a system we provide for you and your team to programmatically fetch and use our data and/or images.

Why would I need an API?

The API provides a fast, consistent and reliable way to get third party data.

What is the difference between a commercial API and a developer API?

A commercial API is for commercial projects and a developer API is for developers. Your project is considered commercial if the primary purpose is to create revenue for the benefit of the owner.

How do I apply for an API key?

You can apply for an API key by clicking the "API" link from the left hand sidebar within your account settings page. You need to have a legitimate business name, address, phone number and description to apply for an API key.

Does the API key cost anything?

Our API is free to use as long as you attribute TMDB as the source of the data and/or images. However, we reserve the right to charge for the commercial API key in the future.

Is there an SLA?

We do not currently provide an SLA. However, we make every reasonable attempt to keep our service online and accessible.

Usamos cookies para ayudarte a tener una mejor experiencia en TMDB. Puedes revisar nuestra [política de cookies](#) para aprender más. Si continúas navegando en TMDB, aceptas esta política. [Vale](#)

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.

FAQ

- Nuestra historia
- Mantente en contacto
- Logos y atribución
- General
- Cuenta
- Sitio web
- Vista general de la API**
- Descubrir ejemplos
- Sesiones API
- Códigos de estado de API
- Librerías de la API

API Overview

Our API is available for everyone to use. A TMDB user account is required to request an API key. Professional users are approved on a per application basis. As always, you must attribute TMDB as the source of your data. Please be sure to read more about this [here](#).

API Documentation

To view all the methods available, you should head over to developers.themoviedb.org. Everything outlined on this page is simply a high level overview to help you understand what is available.

What is TMDB's API?

The API service is for those of you interested in using our movie, TV show or actor images and/or data in your application. Our API is a system we provide for you and your team to programmatically fetch and use our data and/or images.

Why would I need an API?

The API provides a fast, consistent and reliable way to get third party data.

What is the difference between a commercial API and a developer API?

A commercial API is for commercial projects and a developer API is for developers. Your project is considered commercial if the primary purpose is to create revenue for the benefit of the owner.

How do I apply for an API key?

You can apply for an API key by clicking the "API" link from the left hand sidebar within your account settings page. You need to have a legitimate business name, address, phone number and description to apply for an API key.

Does the API key cost anything?

Our API is free to use as long as you attribute TMDB as the source of the data and/or images. However, we reserve the right to charge for the commercial API key in the future.

Is there an SLA?

We do not currently provide an SLA. However, we make every reasonable attempt to keep our service online and accessible.

Usamos cookies para ayudarte a tener una mejor experiencia en TMDB. Puedes revisar nuestra [política de cookies](#) para aprender más. Si continúas navegando en TMDB, aceptas esta política. [Vale](#)

Pondremos los datos necesarios para crear la cuenta

TMDB

Películas

Programas de televisión

Personas

Más

+

ES

Acceder

Únete a TMDB

Q

Beneficios de ser miembro

✓ Encuentra algo que ver entre tus suscripciones a servicios de streaming

✓ Registra las películas y programas de televisión que has visto

✓ Mantén un registro de tus películas y programas de televisión favoritos y obtén recomendaciones de ellos

✓ Crea y mantén una lista de seguimiento personal

✓ Crea listas mezcladas (películas y televisión)

✓ Participa en conversaciones de películas y televisión

✓ Aporta y mejora la información de nuestra base de datos

Crea una cuenta

Crear una cuenta es fácil y gratis. Rellena el formulario para empezar. Se necesita JavaScript para continuar.

Usuario

Contraseña (mínimo 4 caracteres)

Repite la contraseña

Correo electrónico

Al hacer clic en el botón "Registrarse", certifico que he leído y acepto los términos de uso y la política de privacidad de TMDB.

Registrarse

Cancelar

THE MOVIE DB

Usamos cookies para ayudarte a tener una mejor experiencia en TMDB. Puedes revisar nuestra [política de cookies](#) para aprender más. Si continúas navegando en TMDB, aceptas esta política.

Vale

Nos registramos, y tendremos que comprobar nuestro correo electrónico

TMDB

Películas

Programas de televisión

Personas

Más

+

ES

Acceder

Únete a TMDB

Q

Accede a tu cuenta

Para poder editar y valorar en TMDB, así como para obtener recomendaciones personales, deberás acceder con tu cuenta. Si no tienes una, registrarse para obtenerla es gratis y simple. [Pulsa aquí](#) para empezar.

Si ya te has registrado pero aún no has recibido el correo de confirmación, [pulsa aquí](#) para enviarlo de nuevo.

Account verification required

• Your email address hasn't been verified. Please click the verification link in the email that was sent to the address you signed up with. (Don't forget to check your spam folder.) You may [request the email be resent](#) if you are unable to locate your activation email.

Usuario

-

Contraseña

Acceder

Restablecer contraseña

THE MOVIE DB

LO BÁSICO

Sobre TMDB

Contacto

Foros de ayuda

API

PARTICIPA

Guía de aportaciones

Añadir nueva película

Añadir nuevo programa de tv

COMUNIDAD

Directrices

Discusiones

Tabla de clasificación

Twitter

LEGAL

Términos de uso

Términos de uso de la API

Política de privacidad

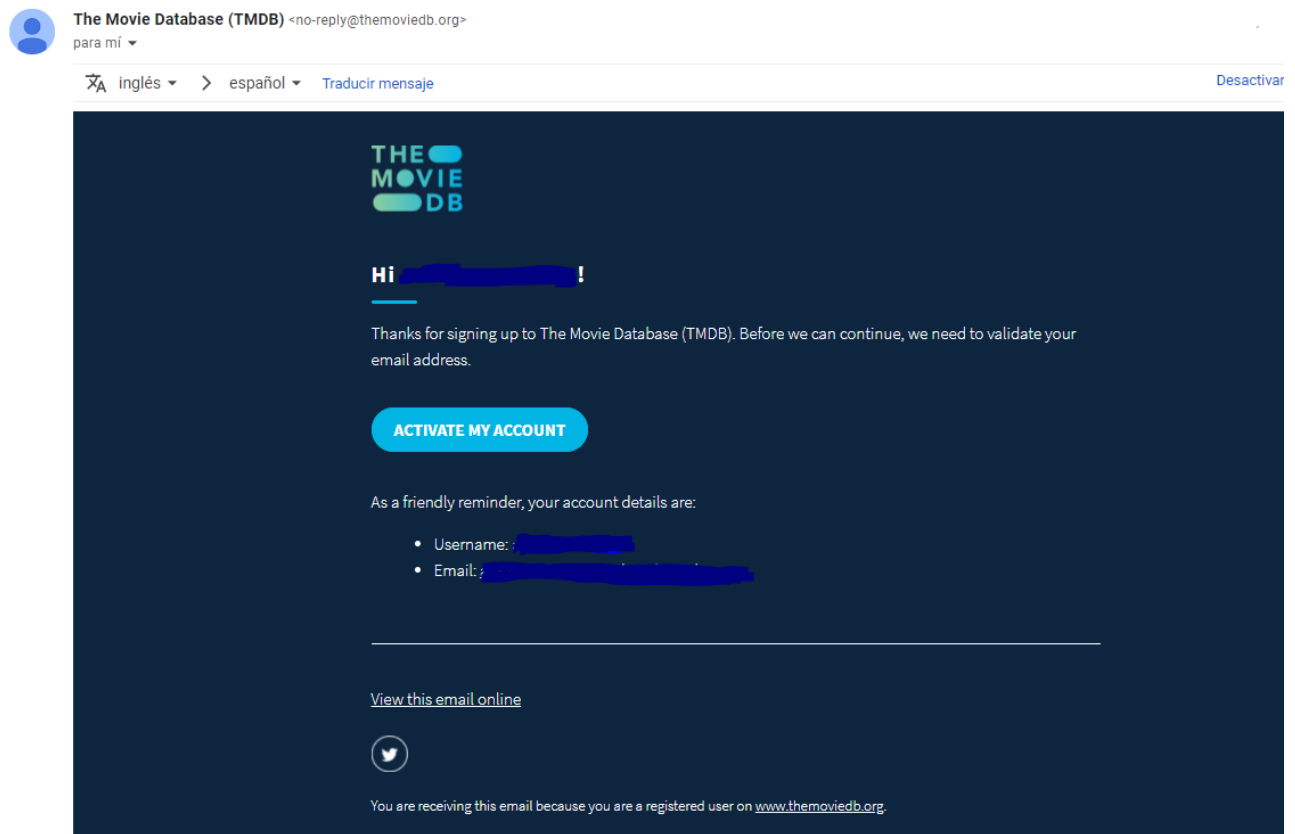
Usamos cookies para ayudarte a tener una mejor experiencia en TMDB. Puedes revisar nuestra [política de cookies](#) para aprender más. Si continúas navegando en TMDB, aceptas esta política.

Vale

Braños Rodríguez Adrián

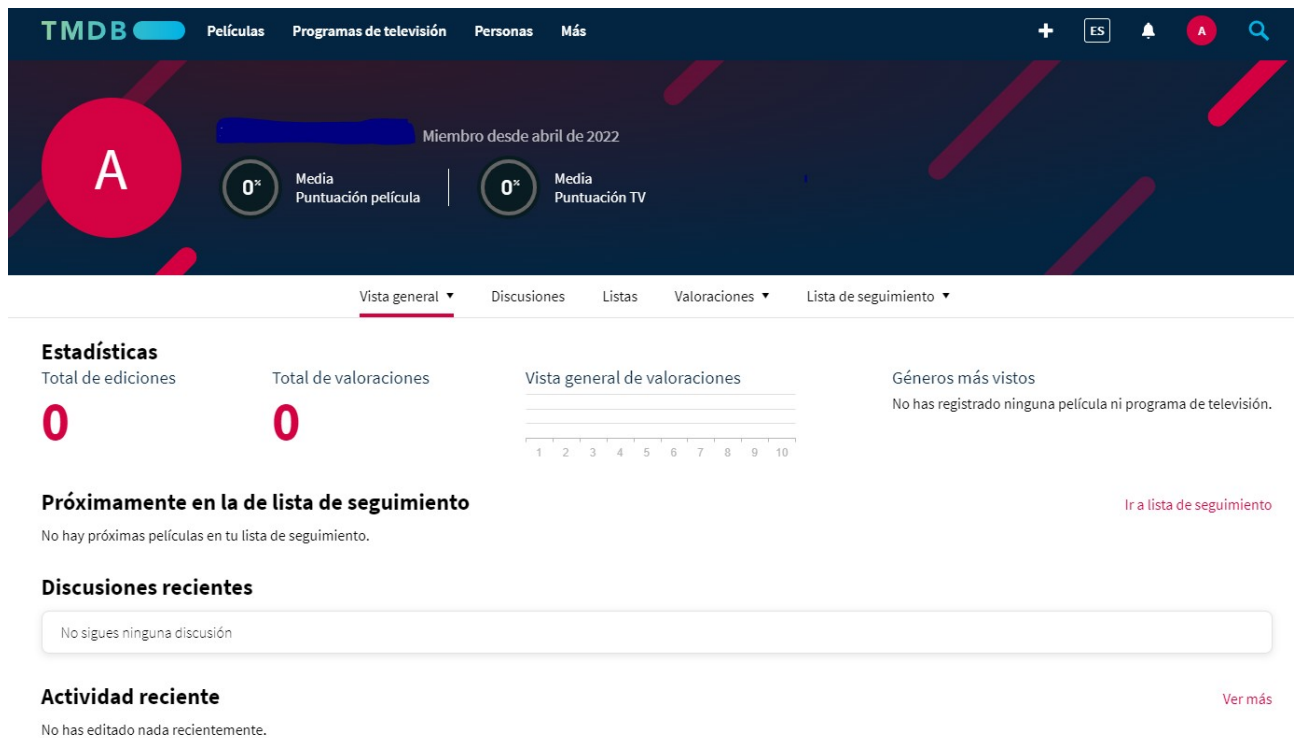
10

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



Una vez que activemos la cuenta ya estaría creada correctamente.

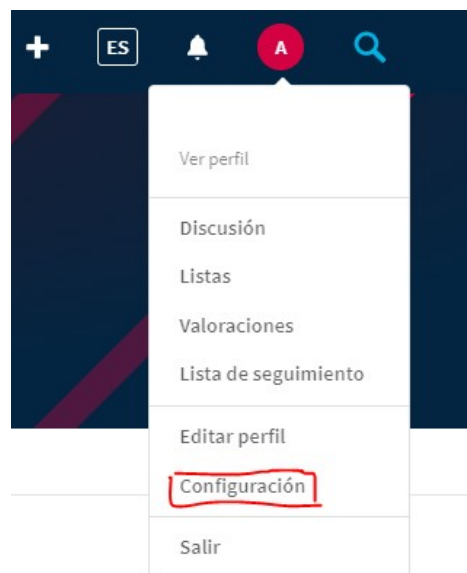
Para ello comprobamos que estamos iniciados



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



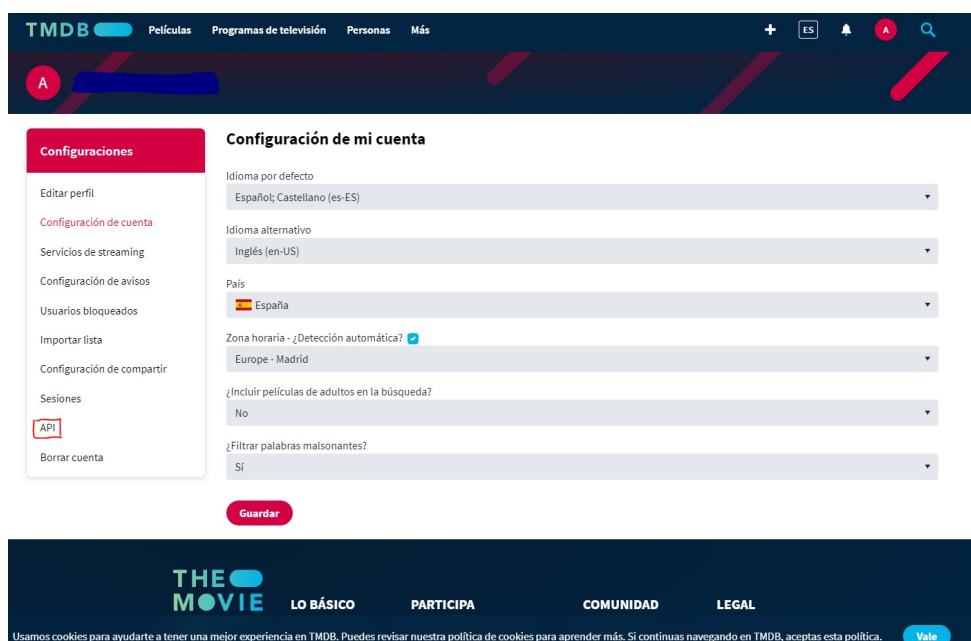
Pinchamos en la foto, se desplegará un menú, y buscamos la opción de ajustes o configuración.



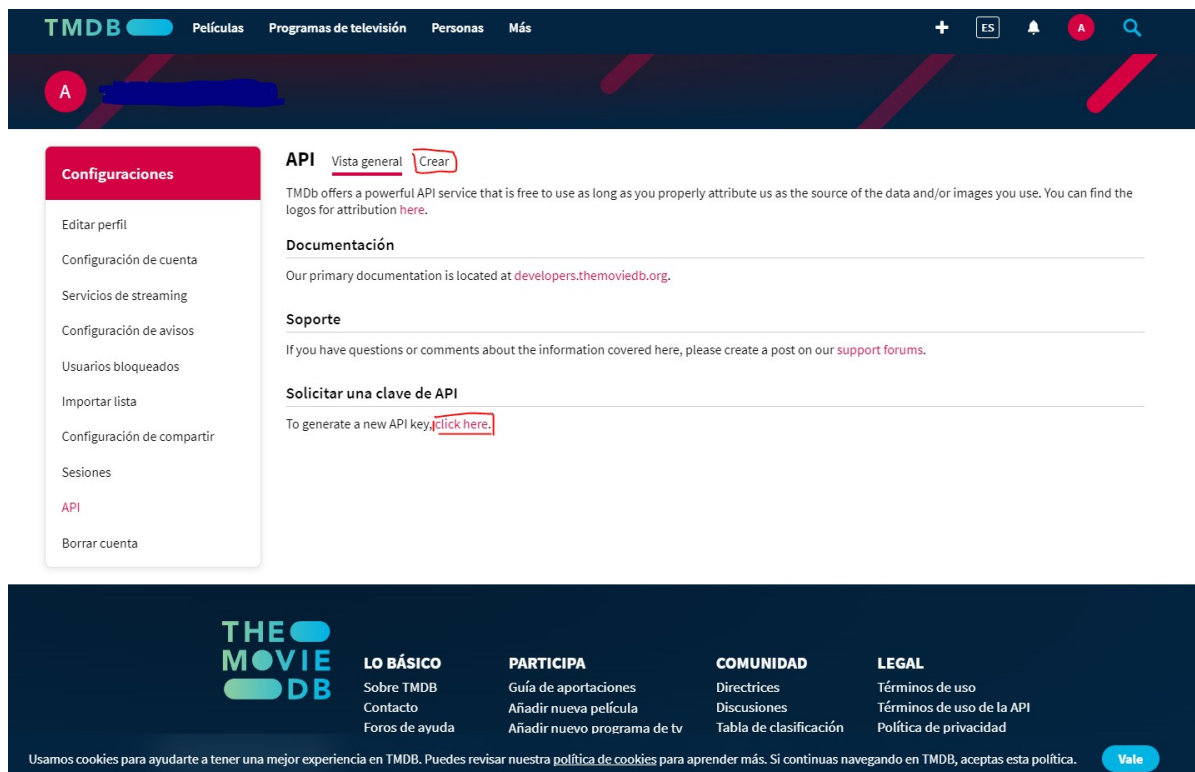
Una vez entremos nos aparecerá ajustes de la cuenta.

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

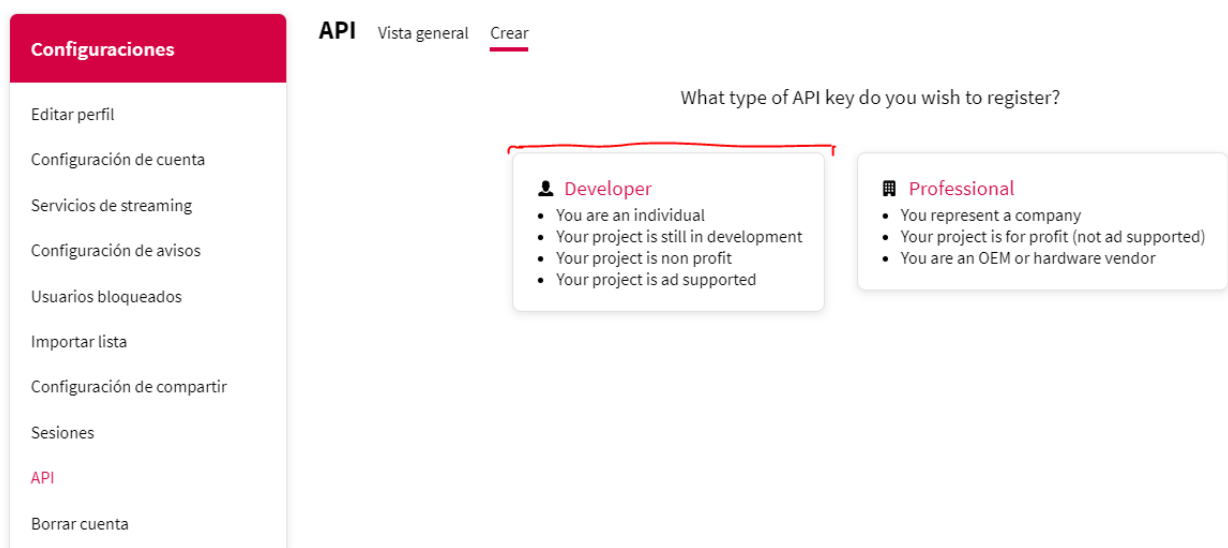
Le pinchamos



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.

Configuraciones

Editar perfil

Configuración de cuenta

Servicios de streaming

Configuración de avisos

Usuarios bloqueados

Importar lista

Configuración de compartir

Sesiones

API

Borrar cuenta

API

Vista general

Crear

Approve Terms of Use

Thank you for using the TMDB application programming interfaces (the "TMDB APIs"). By using the TMDB APIs, you UNCONDITIONALLY CONSENT AND AGREE TO BE BOUND BY AND A PARTY TO THESE TERMS AND CONDITIONS. If you disagree with any of these terms, TMDB does not grant you a license to use the TMDB APIs. TMDB reserves the right, in its sole discretion to modify this Agreement at any time by posting a notice to themoviedb.org. You shall be responsible for reviewing and becoming familiar with any such modification. Such modifications are effective upon first posting or notification and use of the TMDB API by Licensee following any such notification constitutes Licensee's acceptance of the terms and conditions of this Agreement as modified. You can always find the most recent version of these terms here.

Your license to the TMDB APIs under these terms continues until either party terminates it. You may terminate the license by discontinuing use of all or any of the TMDB APIs. TMDB may terminate the license at any time for any reason. Your rights to use the TMDB APIs terminate automatically if (i) you violate any of these terms, (ii) TMDB publicly posts a written notice of termination on themoviedb.org, (iii) TMDB sends a written notice of termination to you (via electronic or other means), or (iv) TMDB disables access to the TMDB APIs to you. If TMDB terminates your license, or you terminate your license, you must immediately cease all use of TMDB content.

1. Licensed Uses and Restrictions

The TMDB APIs are owned by TMDB, Inc. (hereinafter "TMDB") and are licensed to you on a worldwide (except as limited below), non-exclusive, non-transferable, non-sublicenseable basis on the terms and conditions set forth herein. These terms define legal use of the TMDB APIs, all updates, revisions, substitutions that may be made available by TMDB, and any copies of the TMDB APIs made by or for you. TMDB reserves all rights not expressly granted to you.

A. YOU SHALL

1. Comply with the current TMDB Terms of Use at <http://www.themoviedb.org/terms-of-use>

2. You shall give TMDB attribution for any and all content used from the site pursuant to paragraph 3.

B. YOU SHALL NOT

1. Attempt to cloak or conceal your identity or your application's identity when requesting authorization to use TMDB APIs.

2. Use (or create applications that use) an unreasonable amount of bandwidth.

3. Cache any TMDB metadata or photos other than for reasonable periods in order to provide your service.

4. Use TMDB APIs for any application that constitutes, promotes or is used in connection with materials that constitute, promote or are used primarily for the purpose of dealing in: spyware, adware, or other malicious programs or code, counterfeit goods, items subject to U.S. embargo, unsolicited

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

Configuraciones

Editar perfil

Configuración de cuenta

Servicios de streaming

Configuración de avisos

Usuarios bloqueados

Importar lista

Configuración de compartir

Sesiones

API

Borrar cuenta

API

Vista general

Crear

Tipo de uso

Educación

Nombre de la aplicación

Trabajo API

URL de la aplicación

<http://localhost.com>

Resumen de la aplicación

Mini Proyecto de como usar una API con clave y una sin clave

Nombre

No Interesa

Apellido

No Interesa

Dirección de correo electrónico

Número de teléfono (no parenthesis or dashes)

Dirección 1

No Interesa

Dirección 2

No Interesa

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

Configuraciones

- Editar perfil
- Configuración de cuenta
- Servicios de streaming
- Configuración de avisos
- Usuarios bloqueados
- Importar lista
- Configuración de compartir
- Sesiones
- API**
- Borrar cuenta

API [Vista general](#) [Detalles](#) [Directorio](#) [Sesiones](#) [Estadísticas](#)

TMDB offers a powerful API service that is free to use as long as you properly attribute us as the source of the data and/or images you use. You can find the logos for attribution [here](#).

Documentación

Our primary documentation is located at [developers.themoviedb.org](#).

Soporte

If you have questions or comments about the information covered here, please create a post on our [support forums](#).

Detalles de la API

If you'd like to edit the details of your app, [click here](#).

Directorio de aplicaciones

Once you have completed your application, [add it](#) to the app directory!

Clave de la API (v3 auth)

[Blurred API Key]

Ejemplo de solicitud de API

[Blurred API Request]

Token de acceso de lectura a la API (v4 auth)

[Blurred API Token]

Código:

Crearemos un index.html y un main.js.

```
index.html x JS main.js
tmdb > index.html > html > body > div#contain.conta
1 <!DOCTYPE html>
2 <html>
3
4 <body>
5   <div class="contain" id="contain">
6
7   </div>
8 </body>
9 <footer>
10   <script src="main.js"></script>
11 </footer>
12
13 </html>
```

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

```
tmdb > JS main.js > main > cargarPelis
```

```
1 function main() {  
2     //Funcion para poder cargar las peliculas  
3     const cargarPelis = () => {  
4         //peticion para el server  
5         fetch('https://api.themoviedb.org/3/movie/550?api_key=86e79d71f581c6ffac498bb6fc31fb91')  
6     }  
7     cargarPelis()  
8 }  
9 main()
```

Si queremos saber si está funcionando

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

Vamos al navegador y comprobamos

```
function main() {
  //Funcion para poder cargar las peliculas
  const cargarPelis = () => {
    //peticion para el server
    respuestaServer = fetch('https://api.themoviedb.org/3/movie/550?<img alt="redacted URL" data-bbox="648 428 968 458"')
    console.log(respuestaServer)
  }
  cargarPelis()
}
main()
```

```
► Promise { <state>: "pending" }
```

>>

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 línea.

```
function main() {
  //Funcion para poder cargar las peliculas
  const cargarPelis = async () => {
    //peticion para el server
    respuestaServer = await fetch('https://api.themoviedb.org/3/movie/550?l=[REDACTED]')
    console.log(respuestaServer)
  }
  cargarPelis()
}
main()
```


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](#)

Nuestra historia

Mantente en contacto

Logos y atribución

General

Cuenta

Sitio web

Vista general de la API

Descubrir ejemplos

Sesiones API

Códigos de estado de API

Librerías de la API

example response

```
{  "success": false,  "status_code": 7,  "status_message": "Invalid API key: You must be granted a valid key."}
```

	Code	HTTP Status	Message
1	200	Success.	
2	501	Invalid service: this service does not exist.	
3	401	Authentication failed: You do not have permissions to access the service.	
4	405	Invalid format: This service doesn't exist in that format.	
5	422	Invalid parameters: Your request parameters are incorrect.	
6	404	Invalid id: The pre-requisite id is invalid or not found.	
7	401	Invalid API key: You must be granted a valid key.	
8	403	Duplicate entry: The data you tried to submit already exists.	
9	503	Service offline: This service is temporarily offline, try again later.	
10	401	Suspended API key: Access to your account has been suspended, contact TMDB.	
11	500	Internal error: Something went wrong, contact TMDB.	
12	201	The item/record was updated successfully.	
13	200	The item/record was deleted successfully.	
14	401	Authentication failed.	
15	500	Failed.	
16	401	Device denied.	
17	401	Session denied.	
18	400	Validation failed.	
19	406	Invalid accept header.	
20	422	Invalid date range: Should be a range no longer than 14 days.	
21	200	Entry not found: The item you are trying to edit cannot be found.	
22	400	Invalid page: Pages start at 1 and max at 1000. They are expected to be an integer.	
23	400	Invalid date: Format needs to be YYYY-MM-DD.	
24	504	Your request to the backend server timed out. Try again.	
25	429	Your request count (#) is over the allowed limit of (40).	
26	400	You must provide a username and password.	
27	400	Too many append to response objects: The maximum number of remote calls is 20.	
28	400	Invalid timezone: Please consult the documentation for a valid timezone.	
29	400	You must confirm this action: Please provide a confirm=true parameter.	

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

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

```
> response { type: "cors", url: "https://api.themoviedb.org/3/movie/550?api_key=...", redirected: false, status: 200, ok: true, statusText: "OK", headers: Headers, body: ReadableStream, bodyUsed: true }
> object { adult: false, backdrop_path: "/rr7E8M0dXvBk89etis0forjpa.jpg", belongs_to_collection: null, budget: 63000000, genres: (1) [...], homepage: "http://www.foxmovies.com/movies/fight-club", id: 550, imdb_id: "tt0137523", original_language: "en", original_title: "Fight Club", ... }
```

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 {
      //Petición para el server
      respuestaServer = await fetch('https://api.themoviedb.org/3/movie/550?api_key=...')
      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 {
      //Petición para el server
      respuestaServer = await fetch('https://api.themoviedb.org/3/movie/a550?api_key=14db5f47dbb75')
      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

```
GET https://api.themoviedb.org/3/movie/a550?api_key=14db5f47dbb75
Response { type: "cors", url: "https://api.themoviedb.org/3/movie/a550?api_key=14db5f47dbb75" }
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 {
      //Petición para el server
      respuestaServer = await fetch('https://api.themoviedb.org/3/movie/a550?api_key=14db5f47dbb75')
      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 película 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 película
```

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](https://api.themoviedb.org/3).

The screenshot shows the 'The Movie Database API' documentation page. The left sidebar contains a navigation menu with categories like 'GETTING STARTED', 'ACCOUNT', 'AUTHENTICATION', 'CERTIFICATIONS', 'CHANGES', 'COLLECTIONS', 'COMPANIES', 'CONFIGURATION', 'CREDITS', 'DISCOVER', 'FIND', 'GENRES', 'GUEST SESSIONS', 'KEYWORDS', 'LISTS', and 'MOVIES'. Under 'MOVIES', the 'Get Popular' endpoint is highlighted.

The main content area is titled 'Movies' and shows the 'Get Popular' endpoint: `GET /movie/popular`. Below this, there is a description: 'Get a list of the current popular movies on TMDB. This list updates daily.' There are tabs for 'Definition' and 'Try it out'.

Below the endpoint information, there is an 'Authentication' section with a checkbox for 'API Key'.

The 'Query String' section contains a table with the following parameters:

Parameter	Type	Default	Required
api_key	string	<<api_key>>	required
language	string	Pass a ISO 639-1 value to display translated data for the fields that support it. minLength: 2 pattern: ([a-z]{2})-([A-Z]{2}) default: en-US	optional
page	integer	Specify which page to query. minimum: 1 maximum: 1000 default: 1	optional
region	string	Specify a ISO 3166-1 code to filter release dates. Must be uppercase. pattern: ^[A-Z]{2}\$	optional

The 'Responses' section shows the response format: `application/json`. It includes a table with columns for status code, schema, and example. The response is an object with the following properties:

Status Code	Schema	Example
200	object	<pre>{ "page": 1, "results": [{ "poster_path": "/...", "adult": false, "overview": "...", "release_date": "...", "genre_ids": [...], "id": ..., "original_title": "..." }], ...show 9 more properties }</pre>

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 título

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

    try {
      //Petición para el server
      respuestaServer = await fetch('https://api.themoviedb.org/3/movie/popular?r=...')
      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 película")
      } 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?r=...", ... }
► Object { page: 1, results: (20) [...], total_pages: 33215, total_results: 66429, ... }
```

Responses application/json

200	Schema	Example	collapse all
401	object		
404	page	integer	optional
	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_...')  
      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: "https://api.themoviedb.org/3/movie/popular?api_..."}  
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

200	Schema	Example	collapse all
401	object		
404	page	integer	optional
	▼ results	array[object] {Movie List Result Object}	optional
	<u>poster_path</u>	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 longer"
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

The Movie Database API 3
https://api.themoviedb.org/3

[OAS](#)
[RAML](#)
[Support](#)

Select a different version
Filter sections...

- GETTING STARTED
- Introduction
- Authentication
- Daily File Exports
- Languages
- Images
- Image Languages
- Regions
- External IDs
- Popularity
- Request Rate Limiting
- JSON & JSONP
- Append To Response
- Search & Query For Details
- ACCOUNT
- AUTHENTICATION
- CERTIFICATIONS
- CHANGES
- COLLECTIONS

Getting Started

Images

You'll notice that movie, TV and person objects contain references to different file paths. In order to generate a fully working image URL, you'll need 3 pieces of data. Those pieces are a `base_url`, a `file_size` and a `file_path`.

The first two pieces can be retrieved by calling the `/configuration` API and the third is the file path you're wishing to grab on a particular media object. Here's what a full image URL looks like if the `poster_path` of `/kajL17yufv90VlyXyptyrFfak.jpg` was returned for a movie, and you were looking for the `w500` size:

```
https://image.tmdb.org/t/p/w500/kajL17yufv90VlyXyptyrFfak.jpg
```

Company and network logos are available in two formats, SVG and PNG. All of the `logo_path` fields will return a `.png`. This is to maintain backwards compatibility since SVG support was only added very recently. When looking at the image methods there is a new field called `file_type` that will show you the original version of the asset that was uploaded. For SVG's, you should call the `original` image size since we don't resize them. If you prefer to grab PNG's, you can call any size you wish just like normal.

Take for instance Netflix's logo (`wwemzKWzjKjYFFCeIB57q3r4Bcm.svg`), you can call any of the following:

- `https://image.tmdb.org/t/p/original/wwemzKWzjKjYFFCeIB57q3r4Bcm.svg`
- `https://image.tmdb.org/t/p/original/wwemzKWzjKjYFFCeIB57q3r4Bcm.png`
- `https://image.tmdb.org/t/p/w500/wwemzKWzjKjYFFCeIB57q3r4Bcm.png`

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

```
if (respuestaServer.status === 200) {  
  const datos = await respuestaServer.json()  
  
  let peliculas = '';  
  datos.results.forEach(pelicula => {  
    peliculas += `  
      <div class="pelicula">  
          
        <h3 class="titulo">${pelicula.title}</h3>  
      </div>  
    `;  
  });  
  
  document.getElementById("contenedor").innerHTML = peliculas
```

Guardamos y comprobamos

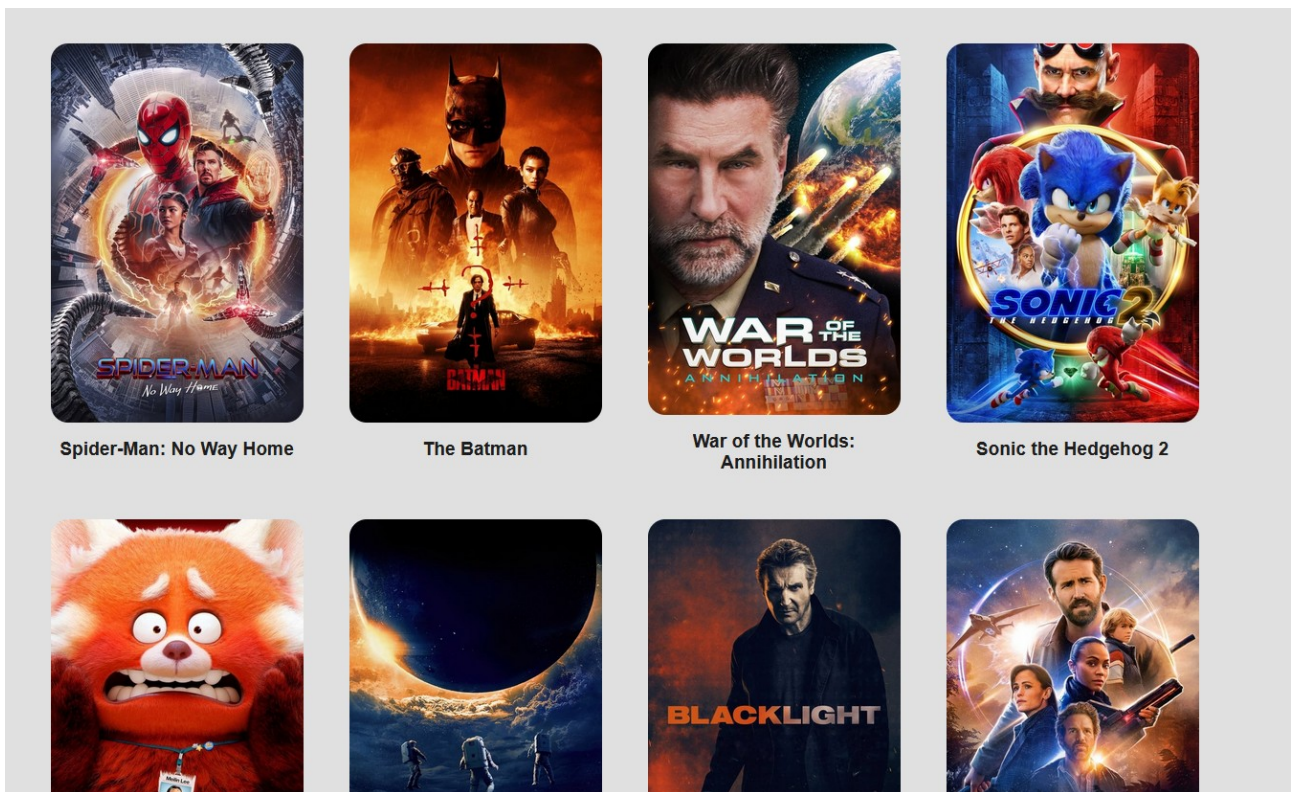


Ahora nos muestra el cartel de la película
mas el titulo de dicha película

Spider-Man: No Way Home



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.

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

Cuando este el final de la key añadimos que la página es dicha variable 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 apartado de películas, películas populares

Query String

Parameter	Type	Description	Required
api_key	string	default: <<api_key>>	required
language	string	Pass a ISO 639-1 value to display translated data for the fields that support it. minLength: 2 pattern: ([a-z]{2})-([A-Z]{2}) default: en-US	optional
page	integer	Specify which page to query. minimum: 1 maximum: 1000 default: 1	optional
region	string	Specify a ISO 3166-1 code to filter release dates. Must be uppercase.	optional

Responses application/json

200 Schema Example

```
{
  "page": 1,
  "results": [
    {
      "poster_path": "/...",
      "adult": false,
      "overview": "...",
      "release_date": "...",
      "genre_ids": [1, 2],
      "id": 123,
      "original_title": "..."
    }
  ]
}
```

Donde buscaremos el apartado de página

Query String

api_key	string	default: <<api_key>>	required
language	string	Pass a ISO 639-1 value to display translated data for the fields that support it. minLength: 2 pattern: ([a-z]{2})-([A-Z]{2}) default: en-US	optional
page	integer	Specify which page to query. minimum: 1 maximum: 1000 default: 1	optional
region	string	Specify a ISO 3166-1 code to filter release dates. Must be uppercase.	optional

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](#)