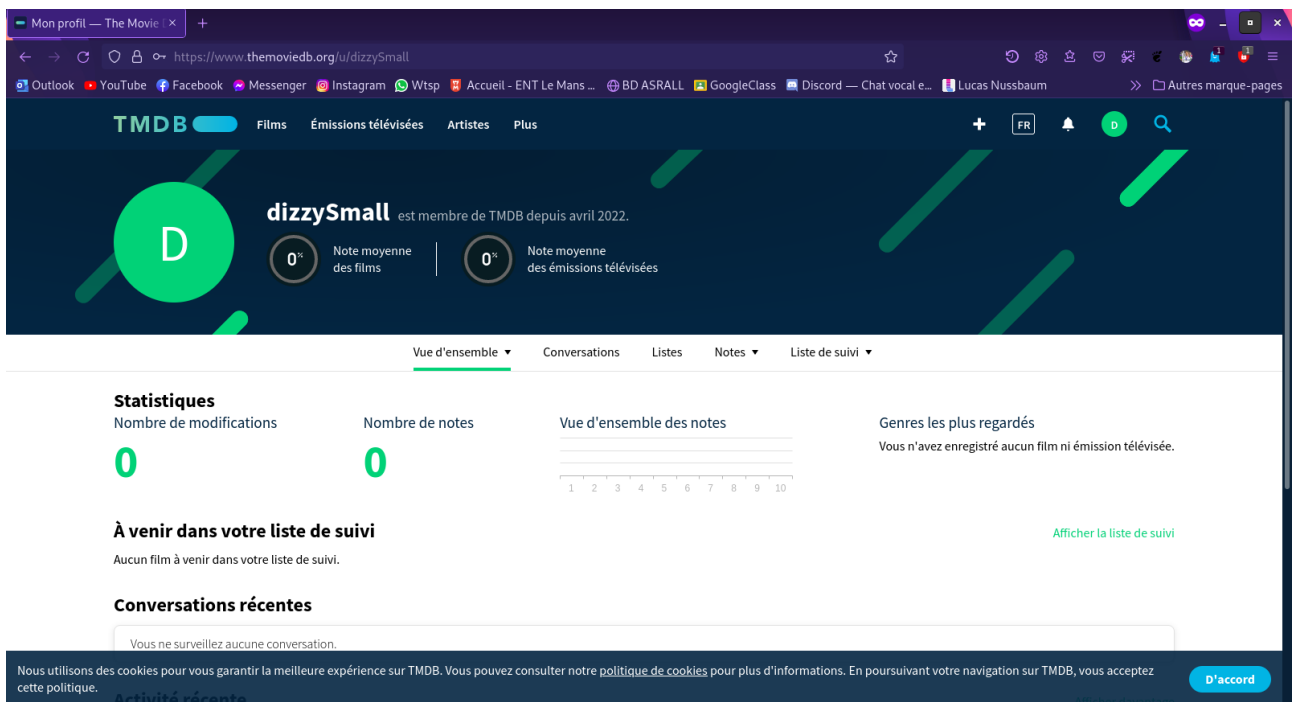


# Tutoriel d'utilisation de l'API TMDB

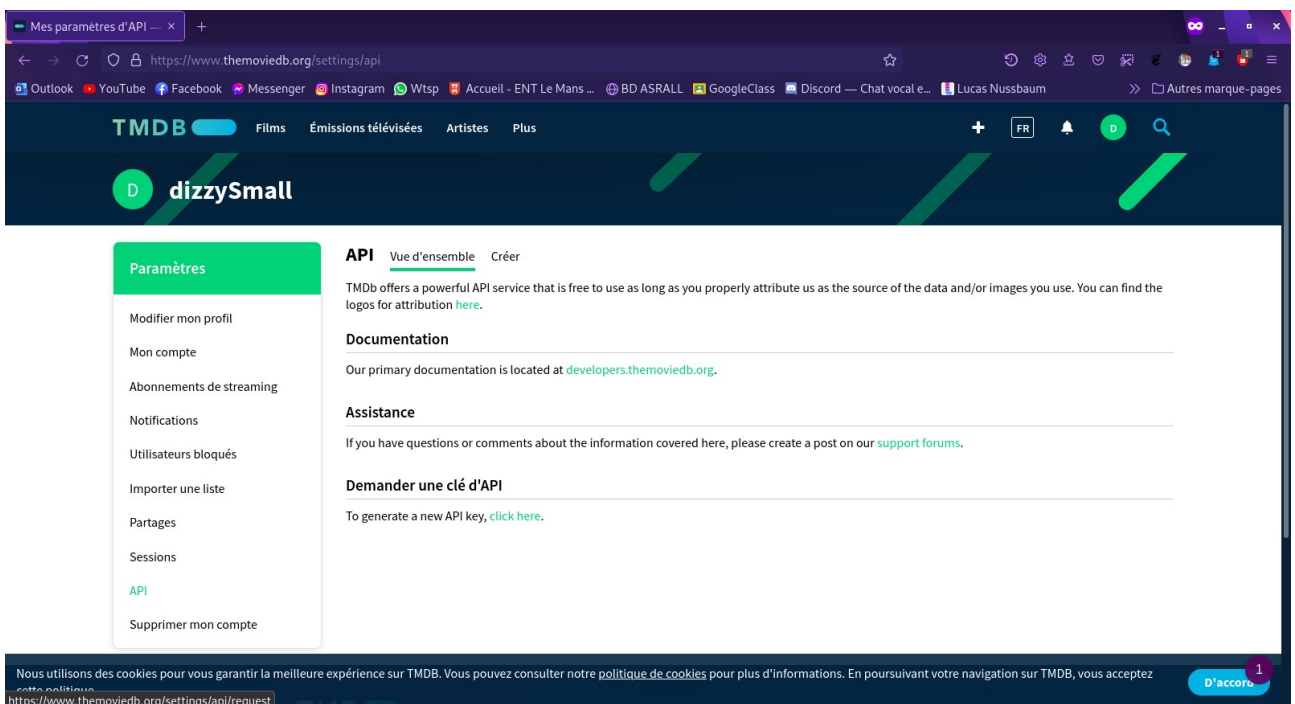
Réalisé par : Aissi Ayoub

## 1) Pré-requis pour utiliser l'API :

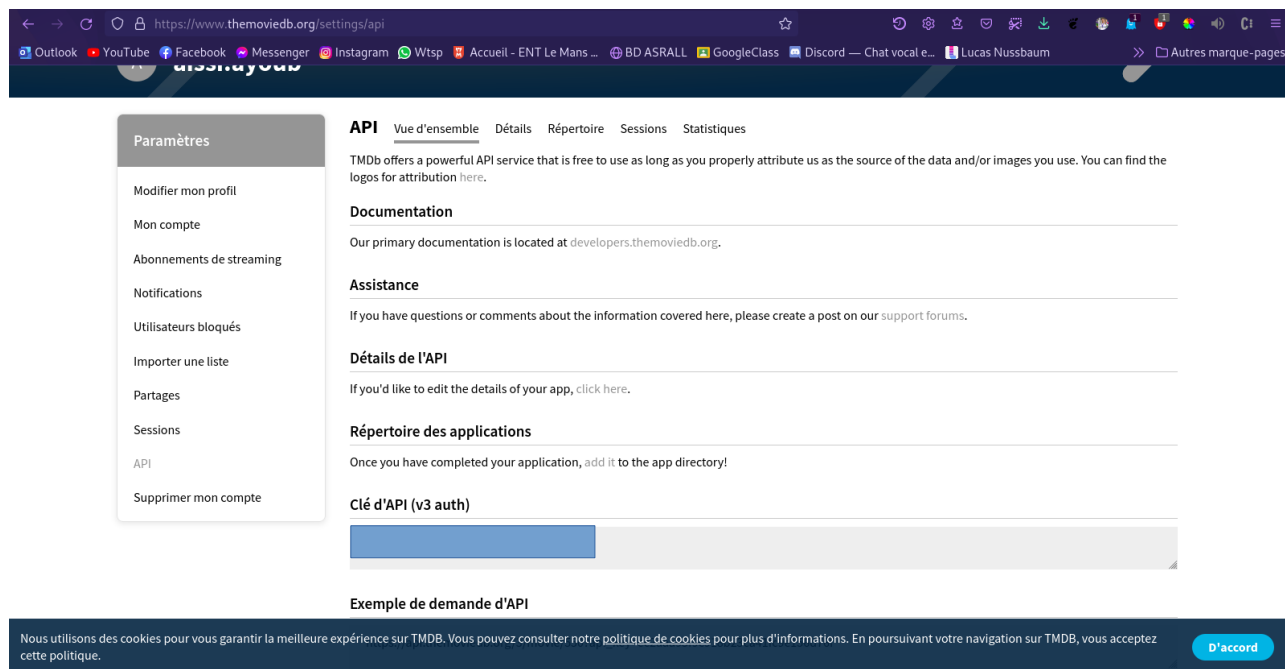
Tout d'abord avant de commencer l'utilisation Il faut se rendre sur le site officiel de l'API <https://www.themoviedb.org> , une fois sur le site on crée notre compte et on tombe sur cette page :



On va sur Paramètres du compte, ensuite sur API sur le menu qui s'affichera à gauche :



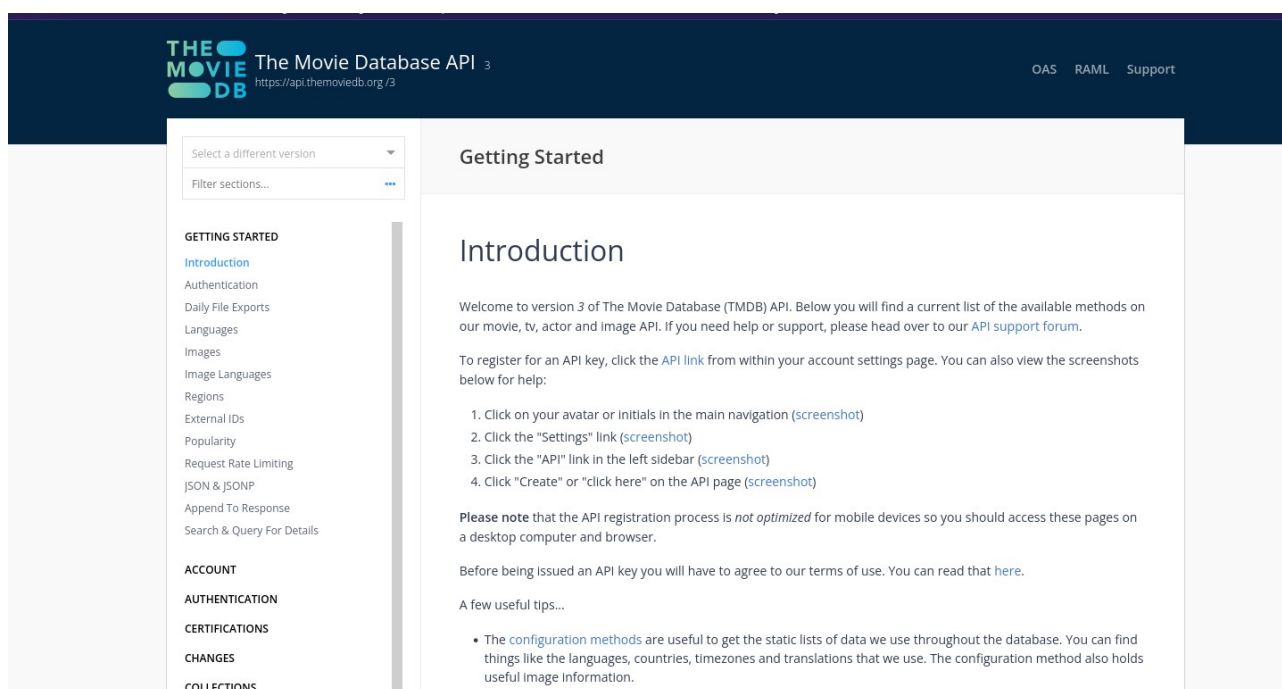
On fait notre demande de clé d'api, pour cela il faudra renseigner les informations demandées, une fois que c'est fait l'organisation traitera notre demande et nous accordera notre clé comme ceci :



On aura donc besoin de la Clé d'API(v3 auth). Il faut bien la conserver secrètement elle appartient qu'à nous maintenant !

## 2) Requêtes d'API :

Une fois qu'on a notre clé les requêtes se feront très facilement maintenant. En effet TMDB offre une énorme documentation très détaillée sur toutes les requêtes possibles, il suffit de se rendre sur cette page et de chercher ce dont on a besoin : <https://developers.themoviedb.org/3/>



Il faut veiller à bien choisir l'api version 3 ! car la documentation pour la v4 n'est pas encore prête.

Une fois sur la donc on a accès à toutes les fonctionnalités. Pour chercher des films par exemple, on va voir la methode Discover movie comme ceci :

The screenshot shows the TMDB API documentation for the **Discover** endpoint. On the left is a sidebar with navigation links: GETTING STARTED, ACCOUNT, AUTHENTICATION, CERTIFICATIONS, CHANGES, COLLECTIONS, COMPANIES, CONFIGURATION, CREDITS, DISCOVER (selected), FIND, GENRES, GUEST SESSIONS, KEYWORDS, LISTS, MOVIES, and NETWORKS. Under DISCOVER, there are links for [GET /discover/movie](#) and [GET /discover/tv](#). The main content area is titled **Discover** and includes the following text:

**Movie Discover**  
**GET** /discover/movie

Discover movies by different types of data like average rating, number of votes, genres and certifications. You can get a valid list of certifications from the [certifications list](#) method.

Discover also supports a nice list of sort options. See below for all of the available options.

Please note, when using `certification` \ `certification.lte` you must also specify `certification.country`. These two parameters work together in order to filter the results. You can only filter results with the countries we have added to our [certifications list](#).

If you specify the `region` parameter, the regional release date will be used instead of the primary release date. The date returned will be the first date based on your query (ie. if a `with.release_type` is specified). It's important to note the order of the release types that are used. Specifying "2|3" would return the limited theatrical release date as opposed to "3|2" which would return the theatrical date.

Also note that a number of filters support being comma (,) or pipe (|) separated. Comma's are treated like an **AND** and query while pipe's are an **OR**.

Some examples of what can be done with discover can be found [here](#).

**Recent Changes**

Date	Change
November 18, 2021	A new <code>without_companies</code> filter is available.
April 13, 2021	A new <code>with_watch_monetization_types</code> filter is available to use with <code>watch_region</code> .
January 2, 2021	A new set of filters are available for watch provider filtering. Check out <code>with_watch_p</code> <code>roviders</code> and <code>watch_region</code> .

On peut voir ici qu'il nous offre deux onglets, un onglet définition qui nous montre toutes les options possible à rajouter dans la requête, comme apr exemple la clé qui est obligatoire et tout le reste optionnel.

The screenshot shows the TMDB API documentation for the **Authentication** and **Query String** sections. On the left is a sidebar with navigation links: MOVIES, NETWORKS, TRENDING, PEOPLE, REVIEWS, SEARCH, TV, TV SEASONS, TV EPISODES, TV EPISODE GROUPS, and WATCH PROVIDERS. The main content area is titled **Authentication** and includes the following text:

**Authentication**  
☒ API Key

**Query String**

Property	Type	Description	Required
api_key	string	default: <<api_key>>	required
language	string	Specify a language to query translatable fields with. minLength: 2 pattern: '^[a-z]{2} -[A-Z]{2}\$' default: en-US	optional
region	string	Specify a ISO 3166-1 code to filter release dates. Must be uppercase. pattern: '^[A-Z]{2}\$'	optional
sort_by	string	Choose from one of the many available sort options. Allowed Values: , popularity.asc, popularity.desc, release_date.asc, release_date.desc, revenue.asc, revenue.desc, primary_release_date.asc, primary_release_date.desc, original_title.asc, original_title.desc, vote_average.asc, vote_average.desc, vote_count.asc, vote_count.desc default: popularity.desc	optional
certification_country	string	Used in conjunction with the certification filter, use this to specify a country with a valid certification.	optional

[...show 32 more properties](#)

Le deuxième onglet qui va nous intéresser c'est Try it out qui nous permet de sélectionner les options qu'on veut mettre dans la requête pour personnaliser notre recherche, et nous génère le lien de notre requête !

Parameter	Type	Required
vote_average.lte	value	optional
with_cast	String	optional
with_crew	String	optional
with_people	String	optional
with_companies	String	optional
with_genres	String	optional
without_genres	String	optional
with_keywords	String	optional
without_keywords	String	optional
with_runtime.gte	value	optional
with_runtime.lte	value	optional
with_original_language	String	optional
with_watch_providers	String	optional
watch_region	String	optional
with_watch_monetization_types	flatrate	optional
without_companies	String	optional

**SEND REQUEST** [https://api.themoviedb.org/3/discover/movie?api\\_key=<<api\\_key>>&language=en-US&sort\\_by=popularity.desc&with\\_watch\\_monetization\\_types=flatrate](https://api.themoviedb.org/3/discover/movie?api_key=<<api_key>>&language=en-US&sort_by=popularity.desc&with_watch_monetization_types=flatrate)

Powered By Stoplight

**EXEMPLE :** On veut rechercher les films et les classer par popularité, voici donc le lien généré :

[https://api.themoviedb.org/3/discover/movie?api\\_key=<<api\\_key>>&language=en-US&sort\\_by=popularity.desc](https://api.themoviedb.org/3/discover/movie?api_key=<<api_key>>&language=en-US&sort_by=popularity.desc)

il suffit de remplacer <<api\_key>> par notre clé, et si on lance le lien sur un navigateur voilà ce qu'on obtient :

JSON	Données brutes	En-têtes
Enregistrer	Copier	Tout réduire
Tout développer	Filter le JSON	
page:	1	
▼ results:		
▼ 0:		
adult:	false	
backdrop_path:	"/x7472vF8CcYTTpPRC0uRxA2cYy.jpg"	
▼ genre_ids:		
0:	28	
1:	12	
2:	878	
id:	406759	
original_language:	"en"	
original_title:	"Moonfall"	
▼ overview:	"A mysterious force knocks the moon from its orbit around Earth and sends it hurtling on a collision course with life as we know it."	
popularity:	6367.77	
poster_path:	"/odVv1sqVs0KxBkIA8bh1B1Pgalx.jpg"	
release_date:	"2022-02-03"	
title:	"Moonfall"	
video:	false	
vote_average:	6.5	
vote_count:	543	
▼ 1:		
adult:	false	
backdrop_path:	"/10Fcv5GbzZ0MkeyKrxPmwnRo5f1.jpg"	
▼ genre_ids:		
0:	28	
1:	12	
2:	878	
id:	634649	
original_language:	"en"	
original_title:	"Spider-Man: No Way Home"	
▼ overview:	"Peter Parker is unmasked and no longer able to separate his normal life from the high-stakes of being a super-hero. When he asks for help from Doctor Strange the stakes become even more dangerous, forcing him to discover what it truly means to be Spider-Man."	
popularity:	5911.127	
poster_path:	"/1g0dhYtq41rTY1GPXvft6k4YLjm.jpg"	
release_date:	"2021-12-15"	
title:	"Spider-Man: No Way Home"	

Un énorme fichier Json contenant toutes les informations sur les films, on note que chaque objet est un film, et chaque objet contient des attributs, ce sont ces attributs là qui vont nous intéresser par la suite.

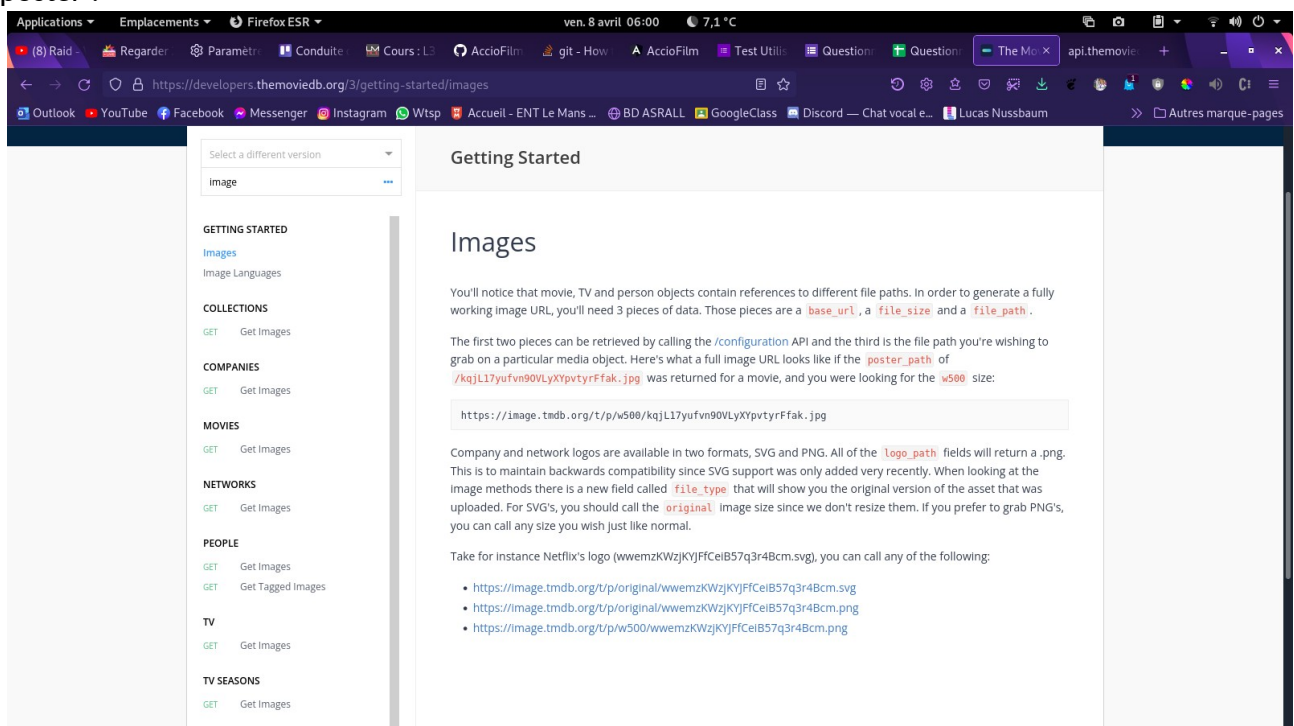
Par exemple On a un pour le premier objet le résultat suivant :

```
{
  "adult":false,
  "backdrop_path":"/x747ZvF0CcYYTTPRC0UrxA2cYy.jpg",
  "genre_ids":[28,12,878],
  "id":406759,
  "original_language":"en",
  "original_title":"Moonfall",
  "overview":"A mysterious force knocks the moon from its orbit around Earth and sends it hurtling on a collision course with life as we know it.",
  "popularity":6367.77,
  "poster_path":"/odVv1sqVs0KxBXiA8bhIBlPgalx.jpg",
  "release_date":"2022-02-03",
  "title":"Moonfall",
  "video":false,
  "vote_average":6.5,
  "vote_count":543}
```

Dans notre back-end quand on va traiter ces données on récupérera ce dont on a besoin par exemple le titre avec l'attribut title, le résumé avec overview etc.

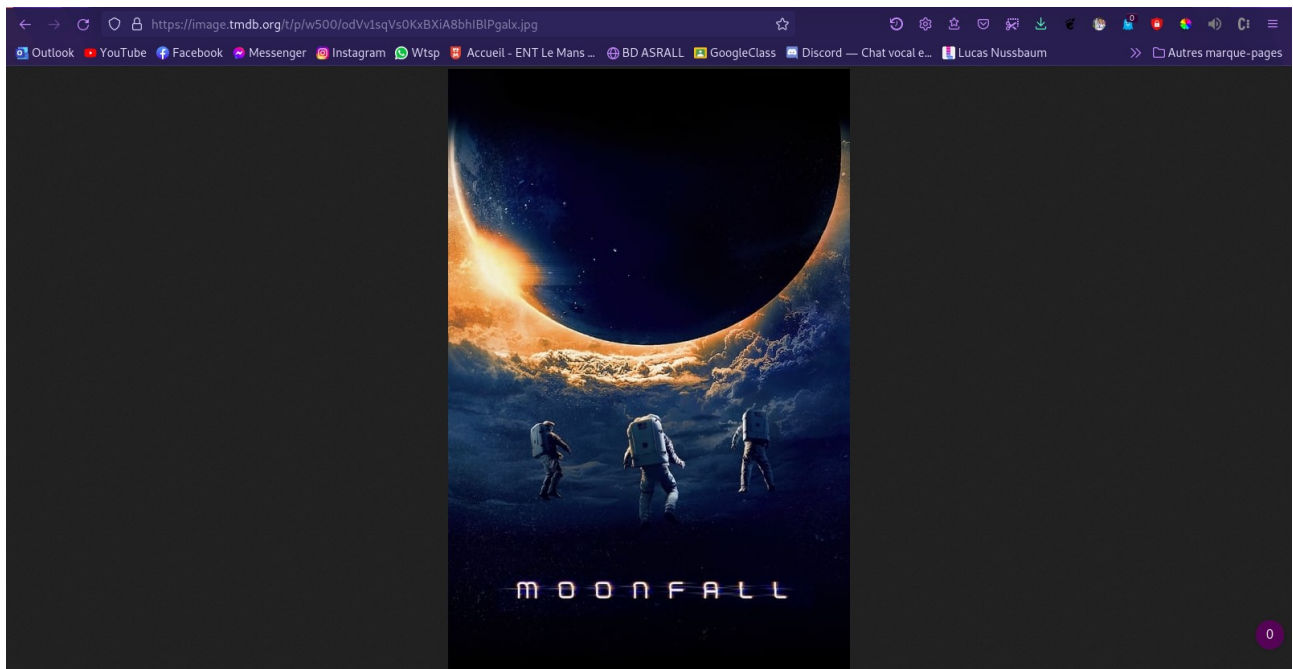
Pour les séries c'est la même chose il suffit de changer movie par tv dans le lien de la requête ! Et oui les séries se nomment tv en référence au mot anglais (tv show) et non pas serie tout simplement!

Pour les poster, qui sont des images, il existe des requêtes spécifiques également comme la methode discover, Il suffit de taper Image dans la recherche et on aura le premier résultat. Pour récupérer le poster d'un film par exemple, il faut d'abord avoir la base de l'url de l'image qui est donné sur la doc : « <https://image.tmbd.org/t/p/w500> », puis il suffit de rajouter le Poster\_Path qu'on peut facilement récupérer avec la requette qu'on a vu juste avant et le combiner avec la base de l'url image :( <https://image.tmbd.org/t/p/w500> )+Poster\_path du film et ça vous donne le poster !

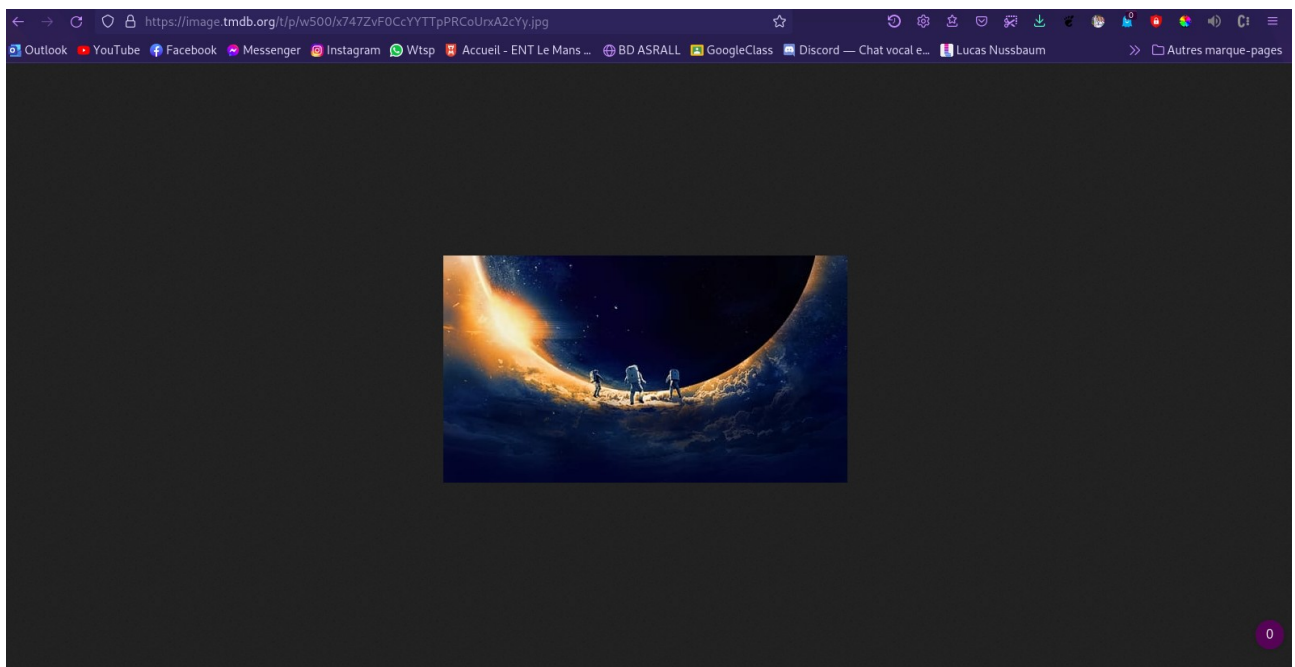
The screenshot shows a web browser displaying the 'Getting Started' page for the TMDB API's image endpoints. The left sidebar contains a navigation menu with categories like 'GETTING STARTED', 'COLLECTIONS', 'COMPANIES', 'MOVIES', 'NETWORKS', 'PEOPLE', 'TV', and 'TV SEASONS', each with a 'Get Images' link. The main content area is titled 'Images' and explains how to construct a full image URL by combining a base URL with a file path. It provides an example URL: `https://image.tmbd.org/t/p/w500/kqjL17yufvn90VlyXypvtyrFfak.jpg`. Below this, it mentions that company and network logos are available in SVG and PNG formats and provides a list of example URLs for Netflix's logo in both formats.

On essaye avec notre exemple précédent ! On avait

"poster\_path":"/odVv1sqVs0KxBXiA8bhIBIPgalx.jpg", pour obtenir donc le poster on lance ce lien : <https://image.tmdb.org/t/p/w500/odVv1sqVs0KxBXiA8bhIBIPgalx.jpg> et on obtient ceci :



Essayons avec "backdrop\_path":"/x747ZvF0CcYYTTPRCOUrXA2cYy.jpg", qui correspond au poster mais en format horizontale, adaptée pour les slideshow par exemple :



Et voilà le tour est joué ! Il reste plus qu'à utiliser ces liens pour charger l'image dans notre site web !



### 3) Back-end traitement des données renvoyées par les requêtes

Vous pouvez utiliser tous les langage de backend pour profiter de cette API, dans mon cas j'utilise JavaScript, un langage assez facile à manipuler et qui offre beaucoup de possibilités !

Regardons un exemple de traitement de données en premier lieu : (utilisez le zoom)


```
1 //##### Récupération des données de l'api avec une requette et
2 var url = "https://api.themoviedb.org/3/discover/movie?"+API_KEY+"&sort_by=popularity.desc";
3
4 function getContent(url){
5   //On fetch l'url qu'on a demandé sur la page, il s'agit d'une requette qui retourne un fichier Json avec des données dedans,
6   //On passe ces données à une autre fonction qui les traite et prend ce dont elle a besoin pour afficher le contenu de la page demandée
7   fetch(url).then(res => res.json()).then(data => {
8     if(data.results.length!=0){
9       //On appelle la fonction d'affichage de films dans le container
10      showSeries(data.results);
11    }
12  })
13 }
```

Ici on passe notre url de requette à notre fonction, elle fetch l'url et récupère les données du fichier Json, ces données sont donc transmis à une deuxième fonction qui va s'occuper de les traiter comme suit :

```
//Traitement des données de films retournées par l'api et création des containers remplis
function showMovies(data) {
  main.innerHTML="" //On vide le contenu de la bbalise sur la page pour mettre un nouveau contenu
  //on traite les données json et on récupère les données qui nous intéressent pour construire et remplir notre container
  data.forEach(movie => {
    //pour chaque objet on récupère le titre, le chemin du poster, la note du film
    //le résumé et son id, on stock ça dans l'objet movie
    const {title, poster_path, vote_average, overview, id} = movie;
    //ici on commence la création dynamique de nos composants en remplissant avec les données récupérées de l'objet movie
    const movieEl = document.createElement('div');
    movieEl.classList.add('movie');
    //Construction du container
    movieEl.innerHTML = `
      
      <div class="movie-info">
        <h3>${title}</h3>
        <span class="${couleurNote(vote_average)}">${vote_average}</span>
      </div>
      <div class="overview">
        <h3>Overview</h3>
        ${overview}
        <button class="voir-plus" id="${id}">Voir plus</button>
      </div>
    `
    //Ajout du container dans le document
    main.appendChild(movieEl);
  })
}
```

elle prend la data en paramètres, il s'agit d'un énorme tableau d'objets movie qui contiennent chacun des attributs et informations sur les films comme on a vu auparavant.


Pour chaque objet on récupère les attributs et on les affiche dans notre html et le résultat est le suivant (après quelques modifications de mon css biensur:) il faudrait taffer le votre aussi ! )



### Overview

Peter Parker is unmasked and no longer able to separate his normal life from the high-stakes of being a super-hero. When he asks for help from Doctor Strange the stakes become even more dangerous, forcing him to discover what it truly means to be Spider-Man.


[Voir plus](#)



### Turning Red

7.5


et le site ressemblera à ça :



### Overview


Un astéroïde percute la Lune. L'orbite de cette dernière est déviée et se dirige vers la Terre. Une équipe d'astronautes, composée entre autres de l'administratrice de la NASA et d'un génie scientifique, se rend sur l'astre afin de sauver la planète. Mais sur place, ils devront faire face à une force inconnue qui pourrait bien changer notre vision de l'Univers.

[Voir plus](#)




### Spider-Man: No Way Home

8.2




### Alerte rouge


7.5





### Sonic 2, le film

7.6









La documentation comporte toutes les fonctionnalités possibles, il faut juste savoir les utiliser derrière sur votre back-end !