Une image contenant texte, capture d’écran, Police, Site web

Description générée automatiquement

Une image contenant texte, capture d’écran, Site web, Page web

Description générée automatiquement

Pour obtenir un jeton d'API auprès de TMdb, voici les étapes à suivre :

1. Créez un compte sur le site de [TMdb](https://www.themoviedb.org/) en cliquant sur "Rejoindre TMdb" et en vous inscrivant avec votre adresse e-mail et un mot de passe.
2. Une fois connecté, accédez à votre compte, puis cliquez sur "Plus -> API".
3. Suivez le lien "API link" pour récupérer votre clé d'API.

Cette clé d'API sera incluse dans les requêtes que vous enverrez au serveur de TMdb pour récupérer des informations sur les films, telles que des listes de films, des images de posters, et des informations détaillées sur les films spécifiques​

<https://www.themoviedb.org/>

Une image contenant texte, capture d’écran, logiciel, Page web

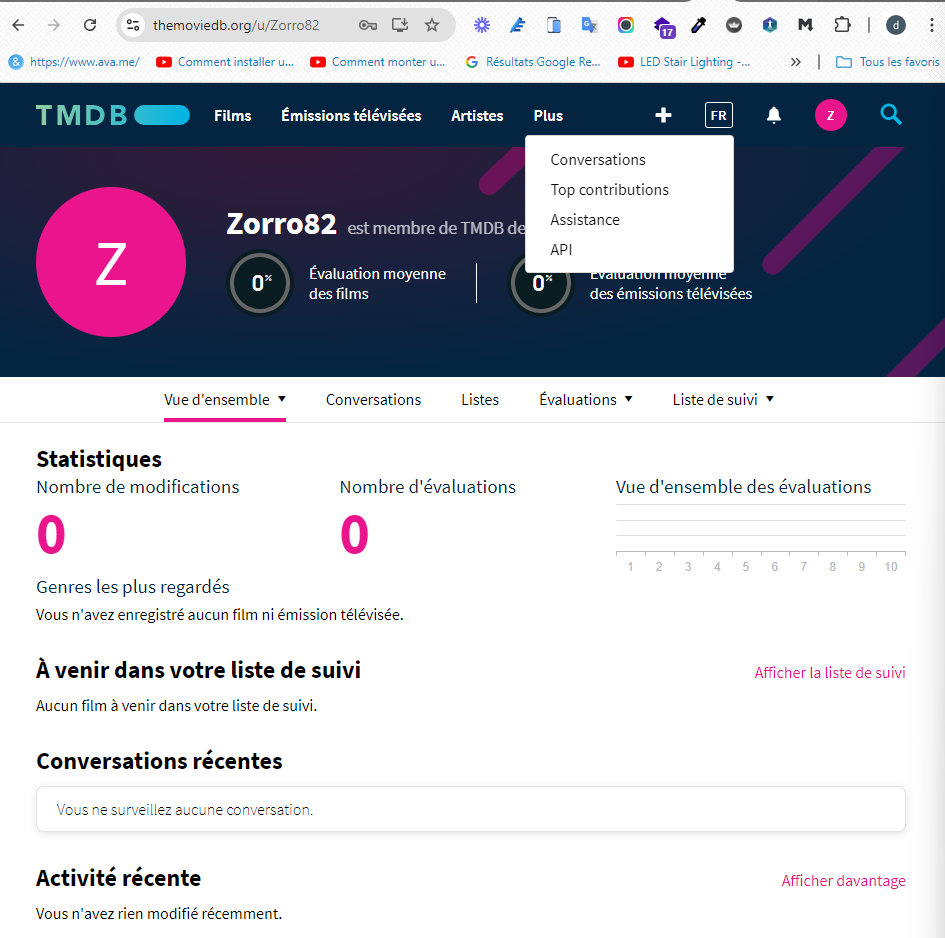
Description générée automatiquement

Une image contenant texte, Appareils électroniques, capture d’écran, logiciel

Description générée automatiquement

Une image contenant texte, capture d’écran, Site web, Page web

Description générée automatiquement



Une image contenant texte, capture d’écran, logiciel, Page web

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, Page web

Description générée automatiquement

Une image contenant texte, capture d’écran, Page web, Site web

Description générée automatiquement

Une image contenant texte, capture d’écran, Page web, Site web

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, Page web

Description générée automatiquement

**Clé d'API :** d806e23f6f54e2b429d698c25b89ef36

**Jeton d'accès en lecture à l'API :** eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiJkODA2ZTIzZjZmNTRlMmI0MjlkNjk4YzI1Yjg5ZWYzNiIsIm5iZiI6MTcyNzk2MzM5NS40NTI1MTMsInN1YiI6IjY2ZmU5ZDcyNmZjNzRlNTc1NmY3ZDQzZSIsInNjb3BlcyI6WyJhcGlfcmVhZCJdLCJ2ZXJzaW9uIjoxfQ.8TXf3YtkqeSJQRK9eV6ETQiDgQDTM0MJoZ4GRvUlWR8

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, Page web

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, Système d’exploitation

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, affichage

Description générée automatiquement

Si le module React-Dom n existe pas il faut l’installer :

Une image contenant capture d’écran, texte

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, Site web

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, affichage

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, Police

Description générée automatiquement

Pour créer un composant "Film" en React qui affichera le résumé et le poster d'un film, suivez ces étapes :

1. Créez un fichier nommé Film.js pour définir le composant.
2. Importez le fichier CSS associé (Film.css) pour styliser le composant.
3. Déclarez le composant fonctionnel "Film", qui prendra les abstract et poster comme props.

Voici un exemple du fichier Film.js :

javascript

Copier le code

//\*\*\* fichier Film.js \*\*\*

import './Film.css';

function Film({abstract, poster}) {

return (

<div className="film">

<img src={poster} alt="poster du film" />

<p>{abstract}</p>

</div>

);

}

export default Film;

Le fichier Film.css associé peut contenir les styles suivants :

css

Copier le code

/\* fichier Film.css \*/

.film {

display: flex;

margin: 15px;

border: 1px solid black;

}

.film img {

width: 150px;

padding: 5px;

}

Ce composant affiche un poster d'un film à l'aide d'une balise img et affiche également le résumé (abstract) dans une balise <p>.

**Problème de ne pas pouvoir l’affichage cet image :**

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

L'image ne s'affiche pas parce que le chemin d'accès aux images peut être incorrect ou parce qu'il y a un problème avec la façon dont les images sont chargées. Voici quelques points à vérifier :

1. **Vérifier le chemin relatif des images** :
   * Assurez-vous que le chemin "../images/dwwm\_1.png" est correct. Ce chemin doit être relatif par rapport à l'emplacement du fichier List.js.
   * Vérifiez que le dossier images et le fichier dwwm\_1.png existent bien dans le répertoire attendu.
2. **Problème de casse dans le chemin** :
   * Assurez-vous que la casse des noms de dossiers et de fichiers correspond exactement (par exemple, Images vs images). En JavaScript, les chemins de fichiers sont sensibles à la casse.
3. **Utiliser un chemin absolu** :
   * Si le problème persiste, essayez d'utiliser un chemin absolu pour vérifier si l'image est trouvée. Par exemple :

javascript

Copier le code

{ poster: "/images/dwwm\_1.png", abstract: "Résumé du premier film" }

* + Cela signifie que le dossier images doit être dans le répertoire public de votre projet.

1. **Vérifier la présence du fichier dans le répertoire public** :
   * Il est souvent recommandé de placer les ressources statiques, comme les images, dans le répertoire public si vous voulez y accéder par un chemin direct (/images/dwwm\_1.png).
   * Si l'image est située dans public/images/dwwm\_1.png, le chemin absolu /images/dwwm\_1.png devrait fonctionner.
2. **Inspecter l'élément** :
   * Utilisez les outils de développement de votre navigateur (clic droit > Inspecter) pour vérifier la balise <img>. Cela vous permettra de voir l'URL exacte utilisée pour l'image et de vérifier s'il y a une erreur de chargement.

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

3

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Une image contenant capture d’écran, texte, affichage, Rectangle

Description générée automatiquement

Pour récupérer des données de l'API TMDB, voici les étapes à suivre en utilisant React et la fonction fetch :

1. **Créer une fonction pour interagir avec l'API TMDB** :

Vous allez créer un fichier TMDBApi.js dans le répertoire src de votre projet. Cette fonction permettra de faire des requêtes à l'API TMDB.

javascript

Copier le code

//\*\*\* fichier TMDBApi.js \*\*\*

const API\_TOKEN = "votre\_API\_Key\_TMDB";

// le paramètre text = le film recherché, page = le numéro de page voulu

export default function getFilmsFromTMdbWithText(text, page) {

const url = `https://api.themoviedb.org/3/search/movie?api\_key=${API\_TOKEN}&language=fr&query=${text}&page=${page}`;

return fetch(url)

.then((response) => response.json())

.catch((error) => console.log(error));

}

* + Remplacez "votre\_API\_Key\_TMDB" par votre clé d'API obtenue sur le site de TMDB.
  + La fonction getFilmsFromTMdbWithText() retourne une promesse qui renvoie les résultats de la requête sous forme de tableau de films.

1. **Intégrer la récupération des films dans le composant List** :

Pour afficher les films récupérés, vous allez intégrer cette fonction dans votre composant List en utilisant le hook useEffect pour effectuer la requête lorsque le composant se charge ou lorsqu'une recherche est effectuée.

javascript

Copier le code

//\*\*\* fichier List.js \*\*\*

import React, { useState, useEffect } from 'react';

import { useParams } from 'react-router-dom';

import './List.css';

import Film from '../Film/Film';

import getFilmsFromTMdbWithText from '../TMDBApi'; // importer la fonction d'API

function List() {

const [films, setFilms] = useState([]);

const { data } = useParams(); // récupérer la recherche de l'utilisateur

useEffect(() => {

if (data) {

getFilmsFromTMdbWithText(data, 1) // rechercher le film avec le paramètre "data"

.then((response) => {

if (response && response.results) {

setFilms(response.results); // mettre à jour l'état des films

}

});

}

}, [data]); // exécuter à chaque changement du paramètre "data"

return (

<div className="container">

<h2>Liste des films</h2>

{films.map((film) => {

return (

<Film

key={film.id}

poster={`https://image.tmdb.org/t/p/w300${film.poster\_path}`}

abstract={film.overview}

/>

);

})}

</div>

);

}

export default List;

* + **useState** : Utilisé pour stocker la liste des films.
  + **useEffect** : Permet de déclencher la récupération des films à chaque fois que la variable data (paramètre de recherche) change.
  + La fonction getFilmsFromTMdbWithText(data, 1) est appelée pour récupérer les films correspondant au texte de recherche data.
  + Le poster\_path est utilisé avec l'URL de base https://image.tmdb.org/t/p/w300 pour afficher l'image.

1. **Vérifier le rendu des films** :

Une fois que vous avez intégré cette fonctionnalité, les films récupérés devraient s'afficher dans votre composant List, chacun avec son poster et son résumé.

Utilisez la console du navigateur pour vérifier que les données sont bien récupérées en ajoutant un console.log(response.results) dans la fonction then() de getFilmsFromTMdbWithText().

Si vous rencontrez des problèmes, assurez-vous que :

* Votre clé d'API est valide et bien insérée dans API\_TOKEN.
* Le format de la réponse est correctement vérifié (par exemple, response && response.results).
* L'URL utilisée pour récupérer les films est correcte et conforme à la documentation de TMDB.

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, Page web

Description générée automatiquement

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Pour afficher les composants Film dans votre composant List, nous allons maintenant nous assurer que les films récupérés depuis l'API sont correctement rendus en tant que composants React. Nous avons déjà vu comment utiliser useEffect pour obtenir les données des films, et nous allons maintenant nous concentrer sur l'affichage de ces données via des composants Film.

**Étape 3 : Affichage des Composants « Film »**

1. **Déclarer un état films avec useState** : L'état films est utilisé pour stocker la liste des films obtenus à partir de l'API TMDB.

javascript

Copier le code

const [films, setFilms] = useState([]);

1. **Utiliser useEffect pour récupérer les films** : Avec useEffect, nous allons appeler getFilmsFromTMdbWithText() chaque fois que la variable data change. Nous mettrons à jour l'état films avec les résultats obtenus.

javascript

Copier le code

useEffect(() => {

if (data) {

getFilmsFromTMdbWithText(data, 1)

.then((response) => {

if (response && response.results) {

setFilms(response.results); // Mettre à jour l'état avec les films obtenus

}

})

.catch((error) => console.log(error)); // Gérer les erreurs

}

}, [data]);

1. **Utiliser .map() pour afficher chaque film** : Maintenant, nous allons utiliser films.map() pour parcourir la liste des films et rendre un composant Film pour chacun d'eux.

javascript

Copier le code

return (

<div className="container">

<h2>Liste des films</h2>

{films.length > 0 ? (

films.map((film) => (

<Film

key={film.id} // Utiliser l'ID unique du film comme clé

poster={`https://image.tmdb.org/t/p/w300${film.poster\_path}`} // URL complète de l'image du poster

abstract={film.overview} // Résumé du film

/>

))

) : (

<p>Aucun film trouvé pour la recherche : "{data}"</p> // Message si aucun film n'est trouvé

)}

</div>

);

1. **Définition du Composant Film** : Assurez-vous que le composant Film prend en charge les props poster et abstract. Voici un exemple du composant Film :

javascript

Copier le code

// Film.js

import React from 'react';

import './Film.css'; // Assurez-vous que le fichier CSS contient les styles nécessaires

function Film({ poster, abstract }) {

return (

<div className="film">

<img src={poster} alt="Poster du film" /> {/\* Affichage du poster \*/}

<p>{abstract}</p> {/\* Affichage du résumé \*/}

</div>

);

}

export default Film;

1. **CSS pour styliser les composants Film** (optionnel) : Vous pouvez ajouter du CSS pour améliorer l'affichage des composants Film. Par exemple :

css

Copier le code

/\* Film.css \*/

.film {

display: flex;

flex-direction: column;

align-items: center;

margin: 20px;

border: 1px solid #ddd;

padding: 10px;

border-radius: 5px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

.film img {

width: 150px;

height: auto;

margin-bottom: 10px;

}

.film p {

font-size: 14px;

color: #333;

}

**Résumé**

* **Étape 1** : Utilisez useState pour créer un état films qui stocke les données des films.
* **Étape 2** : Utilisez useEffect pour appeler l'API TMDB et mettre à jour l'état avec les films récupérés.
* **Étape 3** : Utilisez .map() pour afficher chaque film en utilisant le composant Film.
* **Étape 4** : Créez le composant Film qui prend en charge l'affichage du poster et du résumé.
* **Étape 5** : Stylisez les composants Film avec du CSS pour un rendu agréable.

En suivant ces étapes, vos films récupérés depuis l'API TMDB seront affichés sous forme de composants Film dans la page List.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Pour afficher correctement le poster de chaque film dans le composant Film, vous devez vous assurer que l'URL de l'image est bien formée, en utilisant la propriété poster\_path retournée par l'API TMDB. Voici les étapes détaillées pour vous assurer que le poster de chaque film s'affiche correctement :

**Étape 4 : Affichage du Poster de Chaque Film**

1. **URL de Base pour le Poster** : Le poster des films est accessible via l'URL de base de l'API TMDB. Vous devez ajouter la valeur poster\_path retournée par l'API à cette URL de base. L'URL de base pour les images TMDB est :

ruby

Copier le code

https://image.tmdb.org/t/p/w300

Ici, w300 correspond à la largeur de l'image en pixels. Vous pouvez utiliser d'autres tailles disponibles, par exemple w500 pour une meilleure qualité.

1. **Mettre à Jour le Composant Film** : Mettez à jour votre composant Film pour qu'il prenne en compte l'URL complète du poster, en combinant l'URL de base avec poster\_path.

javascript

Copier le code

// Film.js

import React from 'react';

import './Film.css';

function Film({ poster, abstract }) {

return (

<div className="film">

<img

src={`https://image.tmdb.org/t/p/w300${poster}`} // Construction de l'URL complète de l'image

alt="Poster du film" // Texte alternatif

/>

<p>{abstract}</p> {/\* Affichage du résumé du film \*/}

</div>

);

}

export default Film;

1. **Assurer la Correction dans le Composant List** : Dans votre composant List, lorsque vous mappez la liste des films pour afficher chaque composant Film, utilisez la propriété poster\_path :

javascript

Copier le code

return (

<div className="container">

<h2>Liste des films</h2>

{films.length > 0 ? (

films.map((film) => (

<Film

key={film.id} // Utiliser l'ID du film comme clé unique

poster={film.poster\_path} // Passer "poster\_path" comme prop au composant Film

abstract={film.overview} // Utiliser "overview" pour le résumé du film

/>

))

) : (

<p>Aucun film trouvé pour la recherche : "{data}"</p> // Message si aucun film n'est trouvé

)}

</div>

);

1. **Gérer les Postes Manquants** : Il est possible que certains films n'aient pas de poster\_path. Pour éviter d'afficher une image cassée, vous pouvez vérifier si poster\_path est disponible avant de construire l'URL ou fournir un poster par défaut.

javascript

Copier le code

function Film({ poster, abstract }) {

const posterUrl = poster

? `https://image.tmdb.org/t/p/w300${poster}`

: "https://via.placeholder.com/300x450?text=No+Image+Available"; // Image de remplacement si le poster est manquant

return (

<div className="film">

<img

src={posterUrl}

alt="Poster du film"

/>

<p>{abstract}</p>

</div>

);

}

1. **Ajouter du CSS pour Styliser le Poster** : Pour améliorer l'apparence des posters des films, vous pouvez ajouter du CSS au fichier Film.css :

css

Copier le code

/\* Film.css \*/

.film {

display: flex;

flex-direction: column;

align-items: center;

margin: 20px;

padding: 10px;

border: 1px solid #ddd;

border-radius: 5px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

.film img {

width: 150px;

height: auto;

margin-bottom: 10px;

border-radius: 5px;

}

.film p {

font-size: 14px;

color: #333;

text-align: center;

}

**Résumé**

* Utilisez l'URL de base https://image.tmdb.org/t/p/w300 pour former l'URL complète du poster en ajoutant poster\_path.
* Mettez à jour le composant Film pour qu'il reçoive le poster\_path via ses props et construise l'URL complète du poster.
* Gérer les cas où poster\_path est manquant en fournissant une image par défaut.
* Ajoutez du CSS pour styliser l'affichage des posters.

Ces étapes garantiront que chaque film affiche correctement son poster dans votre application React, offrant ainsi une meilleure expérience visuelle pour l'utilisateur.

Pour entrer votre clé API TMDB dans votre application React, vous pouvez la stocker dans un fichier séparé pour éviter qu'elle soit directement codée en dur dans vos composants. Voici plusieurs façons de procéder :

**1. Créer un Fichier de Configuration**

Vous pouvez créer un fichier de configuration où vous stockez votre clé API.

1. **Créer un fichier config.js dans votre répertoire src** :

javascript

Copier le code

// src/config.js

export const TMDB\_API\_KEY = 'votre\_clé\_API\_TMDB'; // Remplacez par votre clé API

1. **Importer la clé API dans TMDBApi.js** :

javascript

Copier le code

// TMDBApi.js

import { TMDB\_API\_KEY } from '../config'; // Importer la clé API depuis le fichier de configuration

export default function getFilmsFromTMdbWithText(text, page) {

// Construction de l'URL avec la clé API

const url = `https://api.themoviedb.org/3/search/movie?api\_key=${TMDB\_API\_KEY}&language=fr&query=${text}&page=${page}`;

// Faire la requête fetch

return fetch(url)

.then((response) => response.json()) // Transformer la réponse en JSON

.catch((error) => console.log(error)); // Gérer les erreurs

}

**2. Utiliser un Fichier .env pour les Variables d'Environnement**

Pour des raisons de sécurité, il est recommandé de stocker les clés API dans un fichier .env (variables d'environnement). Cela évite de laisser des informations sensibles dans votre code source.

1. **Créer un fichier .env dans le répertoire racine de votre projet** :

bash

Copier le code

REACT\_APP\_TMDB\_API\_KEY=votre\_clé\_API\_TMDB

**Remarque :** Le préfixe REACT\_APP\_ est obligatoire pour que les variables soient accessibles dans une application React.

1. **Ajouter .env au fichier .gitignore** : Cela permet d'éviter de commettre ce fichier dans votre dépôt git, car il contient des informations sensibles.
2. **Accéder à la clé API dans votre code** :

Dans votre fichier TMDBApi.js, vous pouvez accéder à la clé API à l'aide de process.env :

javascript

Copier le code

// TMDBApi.js

const API\_TOKEN = process.env.REACT\_APP\_TMDB\_API\_KEY; // Accéder à la clé API depuis les variables d'environnement

export default function getFilmsFromTMdbWithText(text, page) {

// Construction de l'URL avec la clé API

const url = `https://api.themoviedb.org/3/search/movie?api\_key=${API\_TOKEN}&language=fr&query=${text}&page=${page}`;

// Faire la requête fetch

return fetch(url)

.then((response) => response.json()) // Transformer la réponse en JSON

.catch((error) => console.log(error)); // Gérer les erreurs

}

1. **Redémarrer le Serveur de Développement** : Chaque fois que vous modifiez le fichier .env, vous devez redémarrer votre serveur de développement (npm start ou yarn start).

**Avantages d'Utiliser .env :**

* **Sécurité** : Ne pas exposer directement la clé API dans votre code source.
* **Flexibilité** : Permet de changer la clé API facilement sans toucher au code.

**Exemples d'Utilisation dans Votre Code**

Si vous suivez l'une des méthodes ci-dessus, l'utilisation de la clé API est simplifiée dans votre fichier TMDBApi.js et vous pouvez l'appeler chaque fois que vous avez besoin d'envoyer une requête à l'API TMDB.

**Remarque**

Bien que ces méthodes permettent de sécuriser la clé API dans le code source, si votre application est déployée côté client (comme une application React classique), la clé API reste accessible via les outils de développement du navigateur. Pour une meilleure sécurité, envisagez de créer un backend qui interagit avec l'API TMDB et cache la clé API côté serveur.

**Rappel :**

**Clé d'API :** d806e23f6f54e2b429d698c25b89ef36

**Jeton d'accès en lecture à l'API :** eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiJkODA2ZTIzZjZmNTRlMmI0MjlkNjk4YzI1Yjg5ZWYzNiIsIm5iZiI6MTcyNzk2MzM5NS40NTI1MTMsInN1YiI6IjY2ZmU5ZDcyNmZjNzRlNTc1NmY3ZDQzZSIsInNjb3BlcyI6WyJhcGlfcmVhZCJdLCJ2ZXJzaW9uIjoxfQ.8TXf3YtkqeSJQRK9eV6ETQiDgQDTM0MJoZ4GRvUlWR8