# Général

* API: <https://tvshowdbapi.herokuapp.com/>

# Modalités

* À réaliser individuellement.

# Objectif

* Développer un site web de type Netflix/Crunchyroll où des utilisateurs/utilisatrices peuvent s'inscrire et écouter des téléséries.

# Consignes

* Un maximum de points sera accordé aux applications écrites à l'aide des meilleures pratiques :
  + Utilisation de components (TvShow, Episode, Footer, autres?)
  + Clarté du code
  + Pas de warnings ou erreurs react (vérifiez la console du navigateur).
  + Pas d'erreur de lint (utilisez "npm run lint" et "npm run fix"). Ne modifiez pas le fichier .eslintrc.cjs pour désactiver des règles de validation eslint. N'utilisez pas non plus de directives pour désactiver des erreurs eslint.
* N'utilisez pas de composantes externes (par exemple pour la pagination, codez ces composants vous-mêmes).
* Vous pouvez utiliser les comptes suivants pour des tests, mais il n'est pas possible de les modifier d'aucune façon.
  + e1234567/e1234567

# Accessibilité

Pour toutes les vues, assurez-vous que vos pages sont accessibles. Ça signifie:

* Utiliser l'attribut role pour identifier à quoi servent les différentes parties de la vue.
* L'utilisation judicieuse de l'attribut alt.
* L'utilisation judicieuse de l'attribut aria-required.
* L'utilisation judicieuse de l'attribut aria-describeby.
* Lorsque des erreurs se produisent, placez un descriptif des erreurs qui se sont produites dans une balise où la touche de tabulation s'arrête et placez-y le focus.

# Home.jsx

* La vue HomeView affiche les téléséries disponibles sur le site.
* Les téléséries disponibles sont retournées par l'appel à la méthode [/tvshows](https://tvshowdbapi.herokuapp.com/#/default/get_tvshows).

Une image contenant texte, capture d’écran, Visage humain, Logiciel multimédia

Description générée automatiquement

Nombre de téléséries par page

Pagination

Filtre

* Des zones de saisie permettent à l'utilisateur de filtrer les téléséries par titre (*title*) et/ou par studio. Les changements apportés dans les filtres sont immédiatement appliqués dans l'interface.
* Le filtre par titre permet de limiter l'affichage des téléséries en spécifiant quelques lettres du titre de la télésérie. La casse (majuscule/minuscule) n'a pas d'importance.
* Le filtre par studio permet de limiter l'affichage des téléséries au studio sélectionné dans la liste, si tel est le cas. La liste des studios disponibles est obtenue par un appel à la méthode [/studios](https://tvshowdbapi.herokuapp.com/#/default/get_studios).
* Les filtres sont cumulatifs.
* Affichez 4 téléséries par ligne en mode desktop, 3 par ligne sur une tablette et 2 par ligne en mode mobile.
* Les téléséries sont paginées (4/8/12 ou 16 téléséries par page). Le système de pagination permet le déplacement entre les pages d’albums en cliquant sur numéros de page ou encore en utilisant le bouton précédent ou suivant pour changer la page courante. Le système de pagination indique la page courante. Les liens de pagination suivant et précédent sont désactivés aux moments opportuns.



* Voir [https://bulma.io/.../pagination/](https://bulma.io/documentation/components/pagination/).
* Initialement, le nombre de téléséries par page est 8. Cette information est sauvegardée dans le navigateur et chargée, si présente, au chargement de la vue.
* Le système de pagination indique la page courante à l'aide de la classe is-current.
* Les liens de pagination suivant et précédent sont désactivés (disabled) aux moments opportuns.
* Le nombre de téléséries par page est soit 4, 8, 12 ou 16.

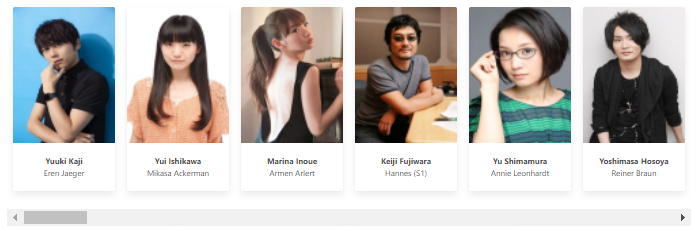
# Details.jsx

* La page Details affiche les informations détaillées concernant une télésérie.
* Elle apparaît au clic d'une télésérie sur la page Home.
* L'api exploitée est [/tvshow?tvshowId=X](https://tvshowdbapi.herokuapp.com/#/default/get_tvshow) où X est un tvshowId valide.
* On retrouve, dans la partie supérieure, les informations détaillées de la télésérie, soit le titre (*title*), l'année de production (*year*), le nombre d'épisodes (*episodeCount*), les genres séparés par une virgule, le studio, et la description (*plot*). L'image (*imgURL*) est dans une colonne à gauche alors que les éléments d'informations précédemment décrits sont à droite[[1]](#footnote-2).

Une image contenant texte, capture d’écran, Visage humain

Description générée automatiquement

* Au chargement de la page, la bande sonore (*audioURL*) de la télésérie est jouée.
* Ensuite sont affichés les acteurs et actrices (en spécifiant leur rôle) ayant participé à la télésérie. Une barre de défilement horizontale permet de consulter tous les acteurs et actrices impliqués. Utilisez un component pour l'affichage d'un artiste et son rôle.



* Finalement, on retrouve, en bas de page, les saisons disponibles sur le site. Les saisons sont disposées horizontalement. Le poster pour la saison est affiché (*imgURL*), ainsi que le nombre d'épisodes de la saison (*episodeCount*) et le numéro de la saison (*number*). Une barre de défilement horizontale apparaît au besoin. Utilisez un component pour l'affichage d'une saison.

Une image contenant Site web

Description générée automatiquement

import { useEffect, useState } from "react";

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

export function Details() {

const [tvShowDetails, setTvShowDetails] = useState(null);

const { tvshowId } = useParams();

useEffect(() => {

const fetchTvShowDetails = async () => {

try {

const response = await fetch(`https://tvshowdbapi.herokuapp.com/tvshow?tvshowId=${tvshowId}`);

if (response.ok) {

const data = await response.json();

setTvShowDetails(data);

} else {

throw new Error("Failed to fetch tv show details");

}

} catch (error) {

console.error("Error fetching tv show details:", error);

}

};

fetchTvShowDetails();

}, [tvshowId]);

if (!tvShowDetails) {

return <div>Loading...</div>;

}

return (

<div className="container">

<div className="section">

<div className="columns">

<div className="column is-one-quarter">

<img src={tvShowDetails.imgURL} alt={tvShowDetails.title} />

</div>

<div className="column">

<h1 className="title">{tvShowDetails.title}</h1>

<p><strong>Year:</strong> {tvShowDetails.year}</p>

<p><strong>Episode Count:</strong> {tvShowDetails.episodeCount}</p>

<p><strong>Genres:</strong> {tvShowDetails.genres.map(genre => genre.name).join(", ")}</p>

<p><strong>Studio:</strong> {tvShowDetails.studio.name}</p>

<p><strong>Description:</strong> {tvShowDetails.plot}</p>

</div>

</div>

<audio controls>

<source src={tvShowDetails.audioURL} type="audio/mp3" />

Your browser does not support the audio element.

</audio>

<div className="section">

<h2 className="subtitle">Cast</h2>

<div className="cast">

{tvShowDetails.roles.map(role => (

<div key={role.roleId} className="actor">

<img src={role.imgURL} alt={role.character} />

<p><strong>{role.character}</strong> - {role.name}</p>

</div>

))}

</div>

</div>

<div className="section">

<h2 className="subtitle">Seasons</h2>

<div className="seasons">

{tvShowDetails.seasons.map(season => (

<div key={season.seasonId} className="season">

<img src={season.imgURL} alt={`Season ${season.number}`} />

<p><strong>Season {season.number}</strong> - {season.episodeCount} episodes</p>

</div>

))}

</div>

</div>

</div>

</div>

);

}

1. L'affichage mobile montre une seule colonne, où l'image apparaît en haut suivi du titre, description et autres éléments. [↑](#footnote-ref-2)