FilmItem

+ _ displayFavoriteImage ()
  - Affiche le coeur à côté du nom si le film est en favori

+ render ()
  - Rend le component
  - Responsable de l'agencement des différentes parties

TMDBApi

+ getImageFromApi (name)
  - Cherche l'image choisie dans l'API

+ getFilmsFromApiWithSearchedText(text, page)
  - Récupère la liste de films à partir du texte entré, dans l'API

+ getFilmDetailFromApi (id)
  - Récupère le détail du film id dans l'API

Favorites

+ render ()
  - Rend le component
  - Affiche la liste des films favoris

+ connect ()
  - Abonné à l'élémt favoritesFilm du Store

Search

+ constructor (props)
  - Créée le component Search
  - Initialise les instance de son props et son state

+ searchTextInputChange (text)
  - Actualise l'instance 'searchText' avec ce que l'utilisateur tappe.

+ _ displayDetailForFilm (idFilm)
  - Dirige vers la fiche détail du film

+ _ displayLoading ()
  - Affiche le logo chargemt pdt 1 chargemt

+ _ loadFilms ()
  - Lance la récup' des films dans l'API
  - Met à jour les instances de Search. (props & state) avec la liste récupérée

+ _ searchFilms ()
  - Reset les instance de Search
  - exécute _loadFilms()

+ render ()
  - Rend le component, affiche une case texte et un bouton Rechercher
  - Affiche une FilmList de films recherchés

FilmList

+ constructor (props)
  - Créée le component FilmList
  - Initialise son state

+ _ displayDetailForFilm (idFilm)
  - Dirige vers la fiche détail du film

+ render ()
  - Rend le component
  - Retourne les données de films entrés sous forme d'une liste de FilmItem

+ connect ()

Navigat°

+ SearchStackNavigator
  - Gère la navigat° entre les components Search et FilmDetail

+ FavoritesStackNavigator
  - Gère la navigat° entre les components Favorites et FilmDetail

+ MoviesTabNavigator
  - Gère la navigat° entre les navigat° SearchStackNavigator et FavoritesStackNavigator

+ createAppContainer (MoviesTabNavigator)
  - Détermine la navigat° principale
  - Exporte la navigat° principale

Store

Reducers

+ toggleFavorite (state, act°)
  - Gère les act° liées au toggle Favorite

+ createStore (toggleFavorite)
  - Crée le store

FilmDetail

+ constructor (props)
  - Créée le component
  - Initialise le state

+ _ displayLoading ()
  - Affiche le logo de chargemt pendant un chargemt

+ ComponentDidMount ()
  - Actualise le state de FilmDetail avec les donnéas du film choisi
  - Récupère les données ou bien de la liste des favoris, ou bien de l'API

+ _ displayFilm ()
  - Responsable de l'agencemt des parties dans la vue

+ _ toggleFavorite ()
  - Rend le film favori
  - Envoie une act° au Store

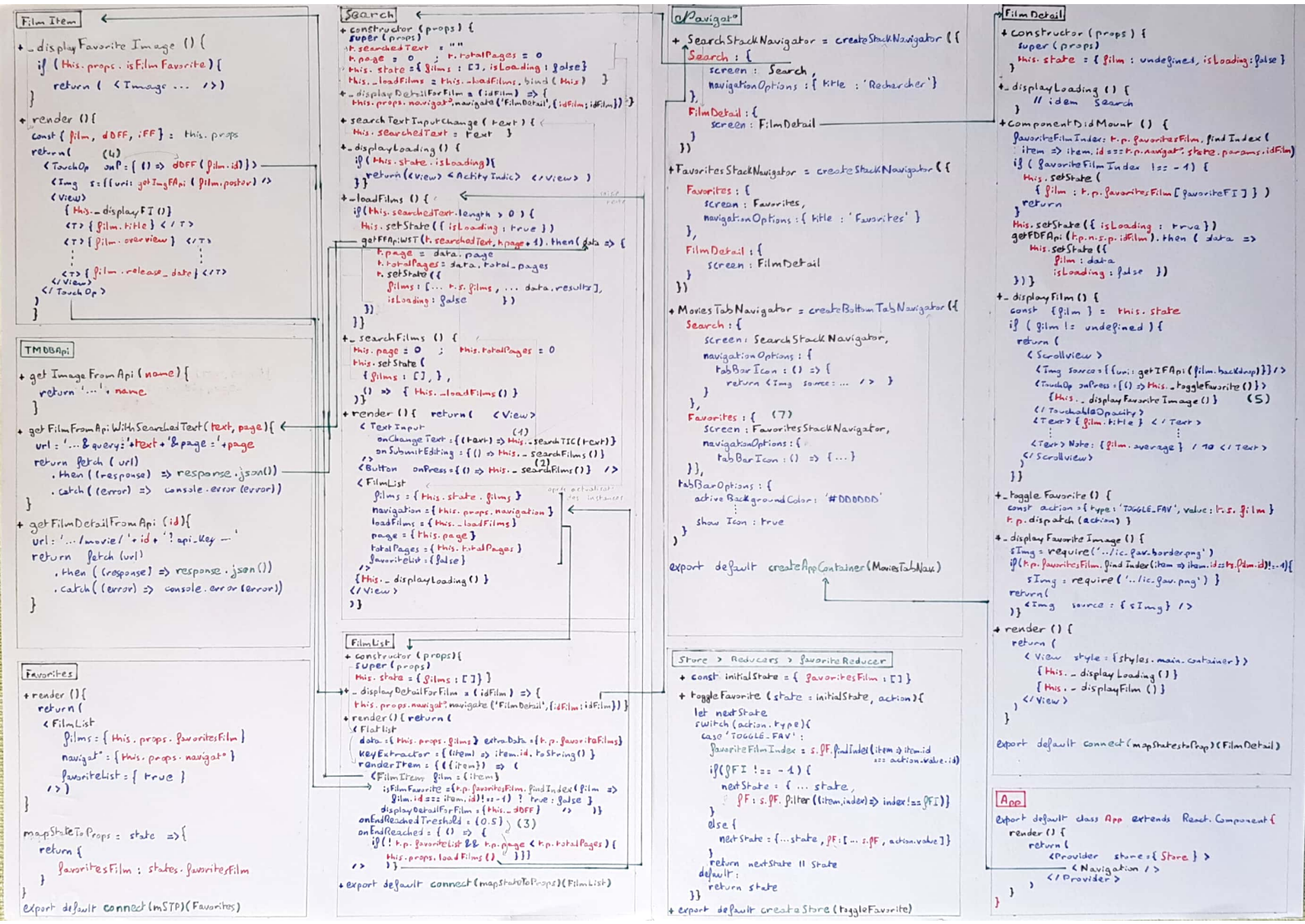
+ _ displayFavoriteImage ()
  - Affiche l'icone favori en fct° de si le film est favori ou non

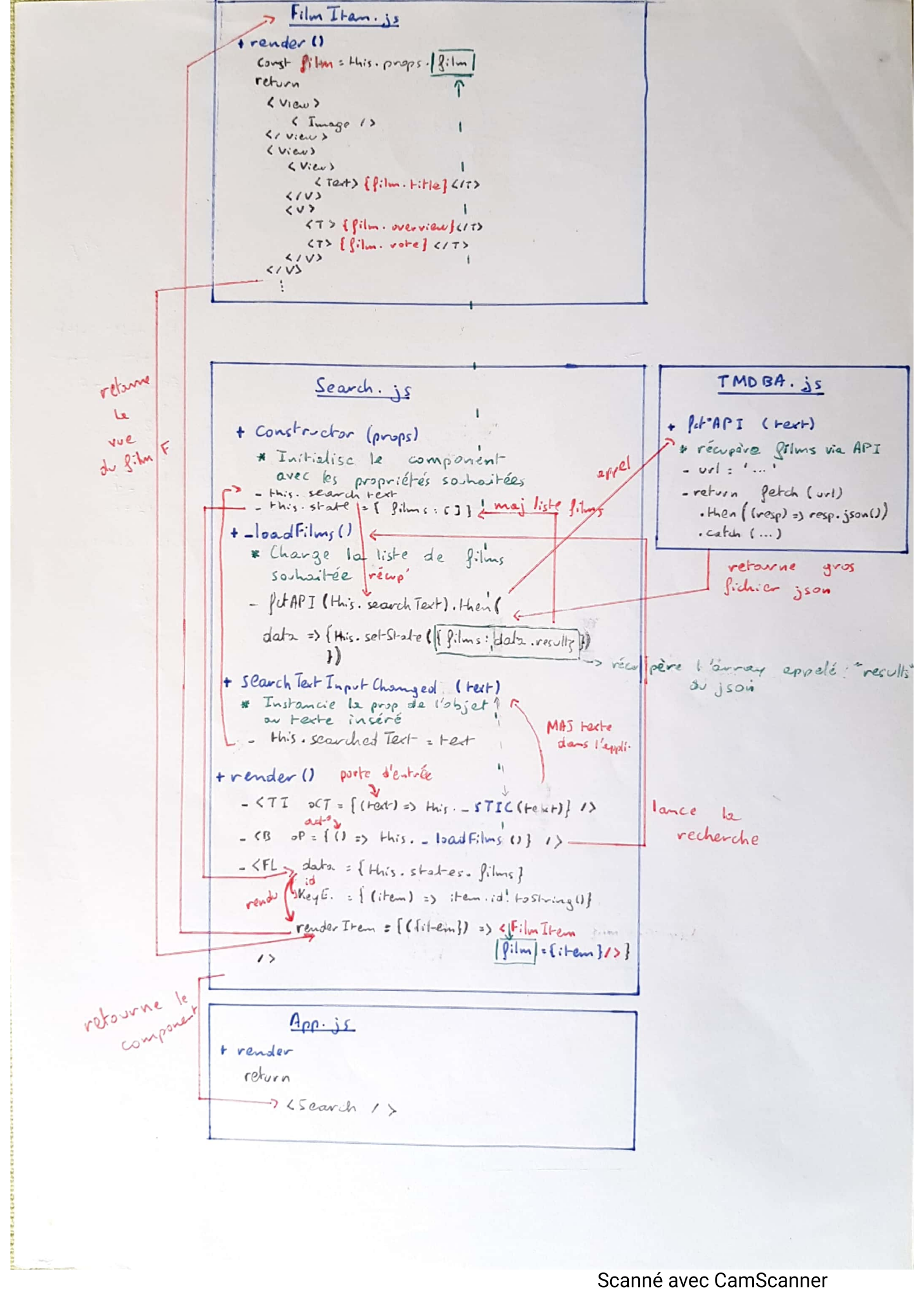
+ render ()
  - Rend le component
  - Appelle la méthode _displayFilm ()

+ connect ()

App

+ render ()
  - Rendu final
  - Appelle le Store et Navigat°




FilmItem.js
+ render()
const film = this.props.film
return
<View>
<Image />
</View>
<View>
<View>
<Text>{film.title}</Text>
</V>
<V>
<T>{film.overview}</T>
<T>{film.vote}</T>
</V>
</V>
Search.js
+ Constructor (props)
* Initialise le component avec les propriétés souhaitées
- this.searchText
- this.state = { films: [] }
+ _loadFilms()
* Charge la liste de films souhaitée
- fetchAPI(this.searchText).then(
data => { this.setState({ films: data.results }) })
+ searchTextInputChanged (text)
* Instancie la prop de l'objet au texte inséré
- this.searchedText = text
+ render()
- <TI onCT = {(text) => this._sTIC(text)} />
- <B onP = {() => this._loadFilms()} />
- <FL data = {this.state.films}
keyE. = {(item) => item.id.toString()}
renderItem = {(film) => <FilmItem film={item} />}
/>
TMDBA.js
+ fetchAPI (text)
* récupère films via API
- url = '...'
- return fetch(url)
.then((resp) => resp.json())
.catch(...)
retourne le vue du film F
appel
maj liste films
retourne gros fichier json
récup' l'array appelé "results" du json
MAJ texte dans l'appli
porte d'entrée
lance la recherche
retourne le component
App.js
+ render
return
<Search />
Scanné avec CamScanner