Documentation du Service API Kitsu

Vue d'ensemble

Ce service TypeScript permet d'interagir facilement avec l'API Kitsu pour récupérer des données d'anime. Il fournit des fonctions typées, une gestion d'erreur robuste et des utilitaires pour manipuler les données.

Base URL : `https://kitsu.io/api/edge`

Installation et Import

```
import {
  fetchCurrentlyAiringAnime,
  fetchUpcomingAnime,
  searchAnime,
  fetchAnimeById,
  fetchEpisodeById,
  fetchEpisodesByAnimeId,
  fetchAnimeCategories,
  fetchAllCategories,
  searchAnimeByCategory,
  getImageUrl,
  getBestTitle,
  type KitsuAnime,
  type KitsuEpisode,
  type KitsuCategory
} from './api';
```

Types Principaux

`KitsuAnime`

Représente un anime avec toutes ses métadonnées :

- `id` : Identifiant unique
- `attributes` : Données principales (titre, synopsis, note, dates, etc.)
- `relationships` : Relations vers épisodes et catégories

`KitsuEpisode`

Représente un épisode d'anime :

- `id` : Identifiant unique
- `attributes` : Données de l'épisode (titre, numéro, synopsis, date de diffusion)
- `relationships` : Relation vers l'anime parent

`KitsuCategory`

Représente une catégorie/genre d'anime :

- `id` : Identifiant unique
- `attributes` : Nom et description de la catégorie

Fonctions Utilitaires

`getImageUrl(imageObj, size)`

Récupère l'URL d'une image avec gestion des fallbacks.

Paramètres :

- `imageObj` (optionnel) : Objet image Kitsu
- `size` : `'tiny' | 'small' | 'medium' | 'large' | 'original'` (défaut: `'medium'`)

Retourne : URL de l'image ou placeholder si aucune image

```
const posterUrl = getImageUrl(anime.attributes.posterImage, 'large');
const thumbnailUrl = getImageUrl(episode.attributes.thumbnail, 'small');
```

`getBestTitle(titles, canonicalTitle)`

Récupère le meilleur titre disponible selon une priorité définie.

Paramètres :

- `titles` (optionnel) : Objet des titres dans différentes langues
- `canonicalTitle` (optionnel) : Titre canonique

Retourne : Meilleur titre disponible

```
const title = getBestTitle(anime.attributes.titles, anime.attributes.canonicalTitle);
```

Fonctions de Récupération des Données

1. Animes Actuellement Diffusés

```
fetchCurrentlyAiringAnime(): Promise<KitsuAnime[]>
```

Récupère les 20 animes actuellement en cours de diffusion, triés par date de début décroissante.

Exemple :

```
const currentAnimes = await fetchCurrentlyAiringAnime();
console.log(`${currentAnimes.length} animes en cours`);

currentAnimes.forEach(anime => {
  console.log(`${getBestTitle(anime.attributes.titles,
anime.attributes.canonicalTitle)} - Note: ${anime.attributes.averageRating}`);
});
```

2. Animes à Venir

```
fetchUpcomingAnime(): Promise<KitsuAnime[]>
```

Récupère les 20 prochains animes à diffuser, triés par date de début croissante.

Exemple :

```
const upcomingAnimes = await fetchUpcomingAnime();
console.log('Prochains animes :');

upcomingAnimes.forEach(anime => {
  console.log(`${getBestTitle(anime.attributes.titles,
anime.attributes.canonicalTitle)} - Sortie: ${anime.attributes.startDate}`);
});
```

3. Recherche d'Animes

```
searchAnime(query: string): Promise<KitsuAnime[]>
```

Recherche des animes par titre.

Paramètres :

- `query` : Terme de recherche

Exemple :

```
const searchResults = await searchAnime('attack on titan');

if (searchResults.length > 0) {
  console.log('Résultats trouvés :');
  searchResults.forEach(anime => {
    console.log(`${getBestTitle(anime.attributes.titles,
anime.attributes.canonicalTitle)} (${anime.attributes.startDate})`);
  });
} else {
  console.log('Aucun résultat trouvé');
}
```

4. Détails d'un Anime

```
fetchAnimeById(id: string): Promise<KitsuAnime | null>
```

Récupère les détails complets d'un anime par son ID.

Exemple :

```
const anime = await fetchAnimeById('1');

if (anime) {
  console.log(`Titre: ${anime.attributes.canonicalTitle}`);
  console.log(`Synopsis: ${anime.attributes.synopsis}`);
  console.log(`Note: ${anime.attributes.averageRating}/100`);
  console.log(`Épisodes: ${anime.attributes.episodeCount}`);
  console.log(`Statut: ${anime.attributes.status}`);
} else {
  console.log('Anime non trouvé');
}
```

5. Détails d'un Épisode

```
fetchEpisodeById(id: string): Promise<KitsuEpisode | null>
```

Récupère les détails d'un épisode par son ID.

Exemple :

```
const episode = await fetchEpisodeById('1');

if (episode) {
  console.log(`Épisode ${episode.attributes.number}:
${episode.attributes.canonicalTitle}`);
  console.log(`Date de diffusion: ${episode.attributes.airdate}`);
  console.log(`Durée: ${episode.attributes.length} minutes`);
  console.log(`Synopsis: ${episode.attributes.synopsis}`);
}
```

6. Épisodes d'un Anime

```
fetchEpisodesByAnimeId(animeId: string): Promise<KitsuEpisode[]>
```

Récupère tous les épisodes d'un anime.

Exemple :

```
const episodes = await fetchEpisodesByAnimeId('1');

console.log(`${episodes.length} épisodes trouvés`);
episodes.forEach(episode => {
  console.log(`Épisode ${episode.attributes.number}:
${episode.attributes.canonicalTitle}`);
});
```

7. Catégories d'un Anime

```
fetchAnimeCategories(animeId: string): Promise<KitsuCategory[]>
```

Récupère les catégories/genres associés à un anime.

Exemple :

```
const categories = await fetchAnimeCategories('1');

console.log('Genres:');
categories.forEach(category => {
  console.log(`- ${category.attributes.title}`);
});
```

8. Toutes les Catégories

```
fetchAllCategories(): Promise<KitsuCategory[]>
```

Récupère toutes les catégories disponibles (40 premières, triées alphabétiquement).

Exemple :

```
const allCategories = await fetchAllCategories();

console.log('Catégories disponibles:');
allCategories.forEach(category => {
  console.log(`${category.attributes.title} (${category.attributes.childCount} sous-
catégories)`);
});
```

9. Recherche par Catégorie

```
searchAnimeByCategory(categoryId: string): Promise<KitsuAnime[]>
```

Récupère les animes appartenant à une catégorie spécifique.

Exemple :

```
// D'abord récupérer l'ID de la catégorie
const categories = await fetchAllCategories();
const actionCategory = categories.find(cat =>
cat.attributes.title.toLowerCase().includes('action'));

if (actionCategory) {
  const actionAnimes = await searchAnimeByCategory(actionCategory.id);
  console.log(`${actionAnimes.length} animes d'action trouvés`);
}
```

Exemple d'Utilisation Complète

```
async function displayAnimeInfo() {
  try {
    // Rechercher un anime
    const searchResults = await searchAnime('naruto');

    if (searchResults.length > 0) {
      const naruto = searchResults[0];
      console.log(`  ${getBestTitle(naruto.attributes.titles,
naruto.attributes.canonicalTitle)}`);

      // Récupérer les détails complets
      const animeDetails = await fetchAnimeById(naruto.id);
      if (animeDetails) {
        console.log(`  Note: ${animeDetails.attributes.averageRating}/100`);
        console.log(`  Épisodes: ${animeDetails.attributes.episodeCount}`);
        console.log(`  Type: ${animeDetails.attributes.subtype}`);

        // Récupérer les genres
        const categories = await fetchAnimeCategories(naruto.id);
        console.log(`  Genres: ${categories.map(c => c.attributes.title).join(',
')}`);
```

```
        // Récupérer quelques épisodes
        const episodes = await fetchEpisodesByAnimeId(naruto.id);
        console.log(`📺 Premiers épisodes:`);
        episodes.slice(0, 3).forEach(ep => {
          console.log(`  ${ep.attributes.number}. ${ep.attributes.canonicalTitle}`);
        });

        // Afficher l'image
        const posterUrl = getImageUrl(animeDetails.attributes.posterImage, 'large');
        console.log(`🖼️ Poster: ${posterUrl}`);
      }
    }
  } catch (error) {
    console.error('Erreur lors de la récupération des données:', error);
  }
}

displayAnimeInfo();
```

Gestion des Erreurs

Toutes les fonctions incluent une gestion d'erreur robuste :

- **Erreurs réseau** : Catchées et loggées, retournent des valeurs par défaut (tableau vide ou `null`)
- **Erreurs API** : Status HTTP vérifié, messages d'erreur détaillés
- **Données manquantes** : Fonctions utilitaires avec fallbacks appropriés

Exemple de gestion d'erreur personnalisée :

```
try {
  const anime = await fetchAnimeById('invalid-id');
  if (!anime) {
    console.log('Anime non trouvé avec cet ID');
    return;
  }
  // Traiter l'anime...
} catch (error) {
  console.error('Erreur inattendue:', error);
  // Gérer l'erreur selon vos besoins
}
```

Limites et Considérations

- **Rate Limiting** : L'API Kitsu peut limiter le nombre de requêtes
- **Pagination** : Les résultats sont limités (généralement 20 éléments)
- **Cache** : Considérez implémenter un cache pour améliorer les performances
- **Erreurs réseau** : Toujours gérer les cas où l'API est indisponible

Endpoints Utilisés

Fonction	Endpoint Complet

fetchCurrentlyAiringAnime	/anime?filter[status]=current&sort=-startDate&page[limit]=20&include=categories
fetchUpcomingAnime	/anime?filter[status]=upcoming&sort=startDate&page[limit]=20&include=categories
searchAnime	/anime?filter[text]={query}&page[limit]=20
fetchAnimeById	/anime/{id}
fetchEpisodeById	/episodes/{id}?include=media
fetchEpisodesByAnimeId	/anime/{id}/episodes?sort=number&page[limit]=20
fetchAnimeCategories	/anime/{id}/categories
fetchAllCategories	/categories?page[limit]=40&sort=title
searchAnimeByCategory	/categories/{id}/anime?page[limit]=20

Paramètres Importants

Filtres :

- `filter[status]=current` : Animes en cours de diffusion
- `filter[status]=upcoming` : Animes à venir
- `filter[text]={query}` : Recherche textuelle

Tri :

- `sort=-startDate` : Par date de début décroissante (récents d'abord)
- `sort=startDate` : Par date de début croissante (prochains d'abord)
- `sort=number` : Par numéro d'épisode
- `sort=title` : Par ordre alphabétique

Inclusions :

- `include=categories` : Inclut les genres/catégories
- `include=media` : Inclut les données de l'anime parent

Pagination :

- `page[limit]=X` : Limite le nombre de résultats