TP : Mini-Projet

Créer le moteur logique d'une application de streaming musical. Ce TP valide votre capacité à structurer une application avant d'attaquer React demain.

Le Cahier des Charges

Vous développez le back-office d'une app de musique. Vous devez gérer des chansons, des styles musicaux et des playlists utilisateurs.

1. Architecture du Dossier

```
/src
  /models      (Vos types et classes objets)
  /services    (La simulation d'API externe)
  index.ts     (Le scénario utilisateur)
```

2. Les fondations (src/models)

Fichier Types.ts :

1. Créez une **Enum** StyleMusical avec 4 valeurs (ex: ROCK, POP, ELECTRO, HIPHOP).
2. Créez une **Interface** Chanson :
   - titre: string
   - artiste: string
   - duree: number (en secondes)
   - style: StyleMusical

Fichier Playlist.ts :

Créez une **Classe** Playlist.

- **Propriétés :**
  - nom: string (public)
  - titres: tableau de Chanson (**private**, on ne doit pas pouvoir le modifier sans passer par les méthodes).
- **Méthodes :**
  - ajouter(chanson: Chanson): ajoute le titre à la liste.
  - retirer(index: number): retire le titre à l'index donné.
  - obtenirDureeTotale(): retourne la somme des durées de toutes les chansons.

3. La Simulation d'API (src/services/MusicAPI.ts)

Dans la vraie vie, on cherche les chansons sur un serveur (Spotify, Apple Music). On va

simuler ça.

Créez un module qui exporte une fonction **asynchrone** rechercherTitres(requete: string).

- **Retour :** Promise<Chanson[]> (Une promesse de tableau de chansons).
- **Comportement :**
  1. Affiche "Recherche en cours sur le serveur...".
  2. Attend 2 secondes (simulé avec setTimeout).
  3. Retourne un tableau de 2 ou 3 fausses chansons correspondant (ou non) à la recherche.
  4. **Bonus :** Si la recherche est vide "", la promesse doit être **rejetée** (reject) avec une erreur "Recherche vide interdite".

4. Le Scénario Final (src/index.ts)

C'est l'heure de tester votre moteur. Dans une fonction main asynchrone :

1. Créez une nouvelle Playlist nommée "Mes Favoris 2025".
2. Lancez une recherche pour "Daft Punk" (n'oubliez pas await !).
3. Si la recherche réussit :
   - Ajoutez les chansons trouvées dans votre playlist.
   - Affichez "Playlist mise à jour !".
4. Affichez la durée totale de la playlist.
5. **Gestion d'erreur :** Tentez une recherche vide "" et capturez l'erreur avec un try/catch.

Pour les rapides

1. **Le JukeBox :** Ajoutez une méthode jouerAleatoire() dans la classe Playlist qui affiche un titre au hasard dans la console.
2. **Le Filtre :** Créez une méthode qui retourne seulement les chansons d'un style précis (ex: que du ROCK).
3. **Typage strict de la durée :** Créez une fonction utilitaire qui transforme les secondes (ex: 125) en format lisible "2m 05s".
- **Playlist = State React :** Demain, votre playlist sera stockée dans un useState([]). La logique d'ajout/retrait sera identique.
- **MusicAPI = Fetch :** Demain, on remplacera votre fausse fonction par un vrai appel réseau (fetch). La structure async/await restera exactement la même.