420-4W6 - TP2 - Spotification 🎵

Date de remise : Semaine 6

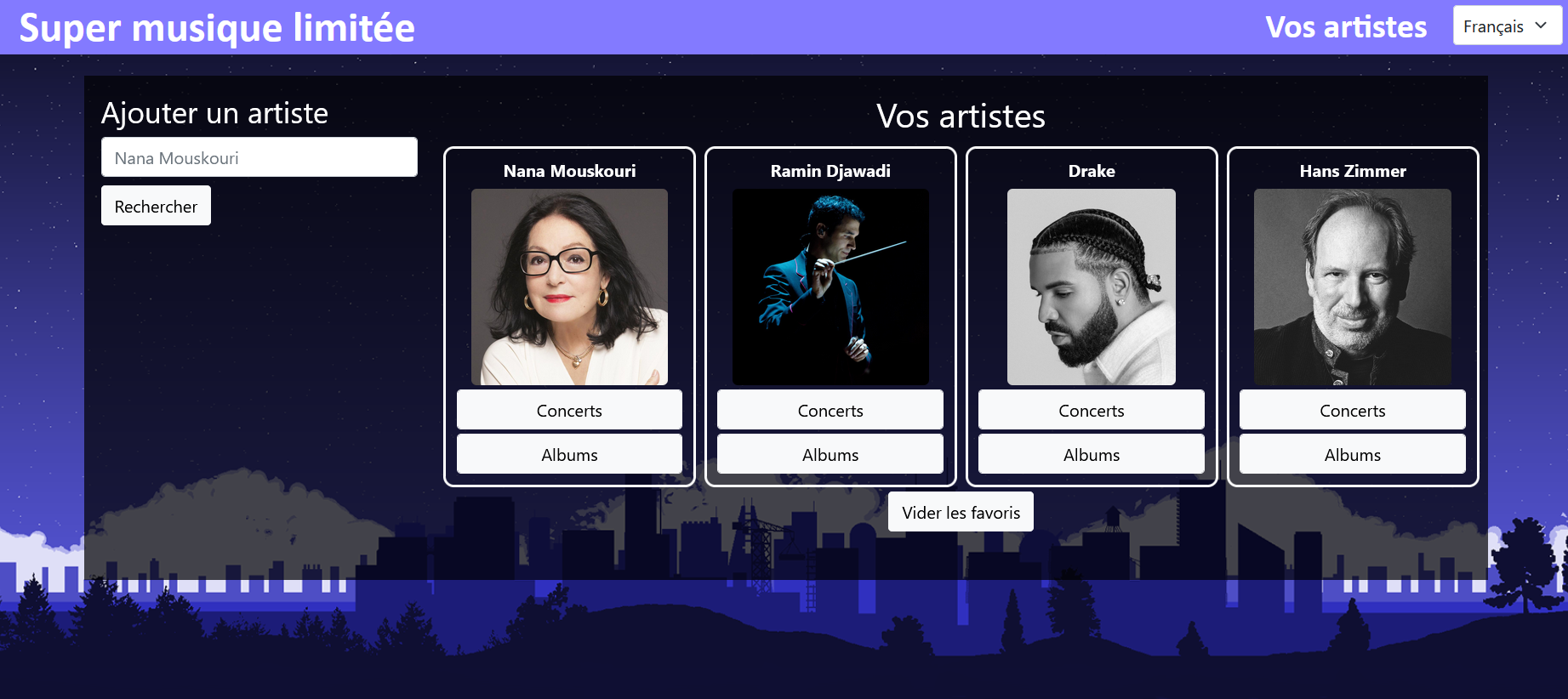
Pondération : 20%

# Consigne

Le projet sera fait de façon individuelle.

Le projet sera fait avec le Framework Angular.

Si vous ne savez pas du tout par où commencer pour ce projet, rendez-vous à la dernière page de ce document.



# Mise en contexte

Vous devez faire une gestion de contenu musical utilisant différents APIs.

L’application permettra de :

* Rechercher des artistes (et les conserver en favoris) à l’aide de l’API de Spotify.
* Rechercher les albums d’un artiste à l’aide de l’API de Spotify.
* Rechercher les chansons d’un album à l’aide de l’API de Spotify.
* Rechercher les concerts d’un artiste à l’aide de l’API BandsInTown ET afficher les emplacements des concerts à l’aide de l’API Google Maps.
* Rechercher une vidéo Youtube à partir d’une chanson pour l’écouter.

# Consignes générales

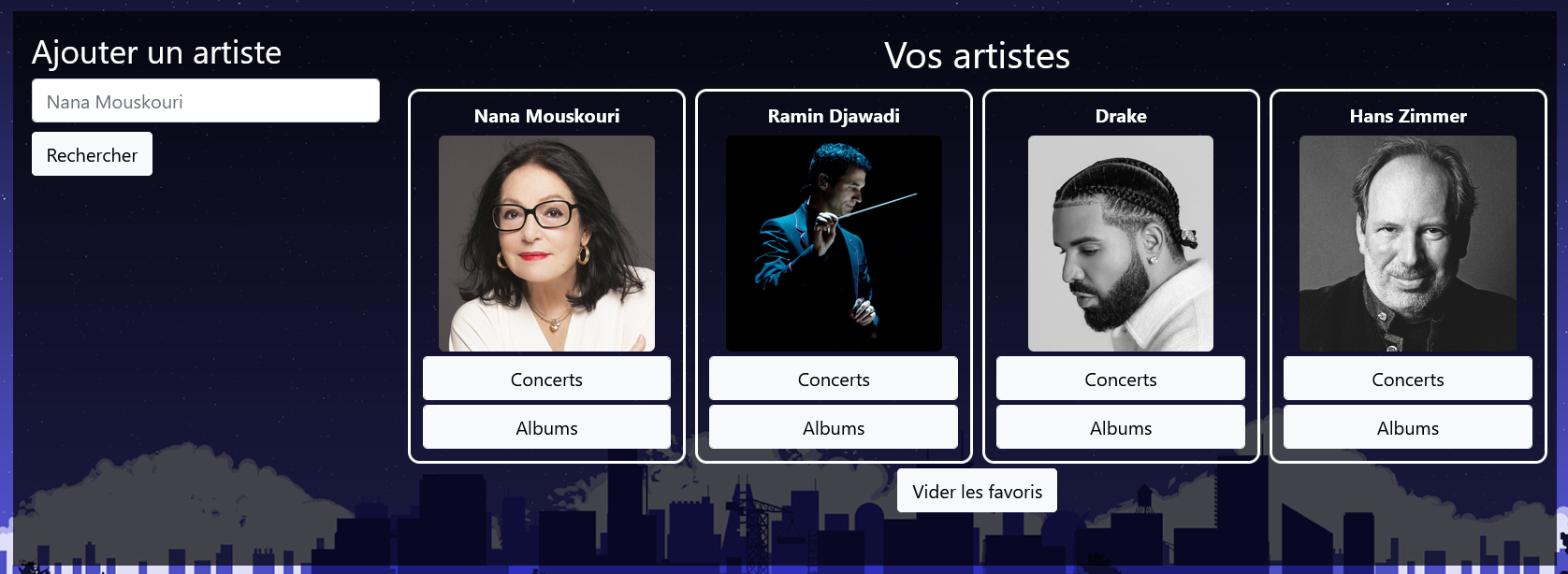
L’application devra être disponible en Anglais et en Français.

# Page d’accueil (Liste d’artistes)

À partir de la page d’accueil, l’utilisateur pourra ajouter des artistes à une liste d’artistes favoris. Cette liste sera persistante (sauvegardée) dans le localStorage du navigateur.

Une requête à Spotify permettra de récupérer des informations sur des artistes. Vous devrez afficher le nom et une image de chaque artiste.

À partir de l’artiste nous pourrons aller voir les spectacles prévus de l’artiste ou encore aller voir les albums de l’artiste.



# Page des concerts

Une page permettra d’afficher les concerts d’un artiste. Nous récupérerons les concerts à partir d’une requête à l’API de BandsInTown.

Le nom de l’artiste devra être fourni en paramètre dans le routage du composant.

Une seule carte Google Maps permettra de voir l’emplacement des concerts.



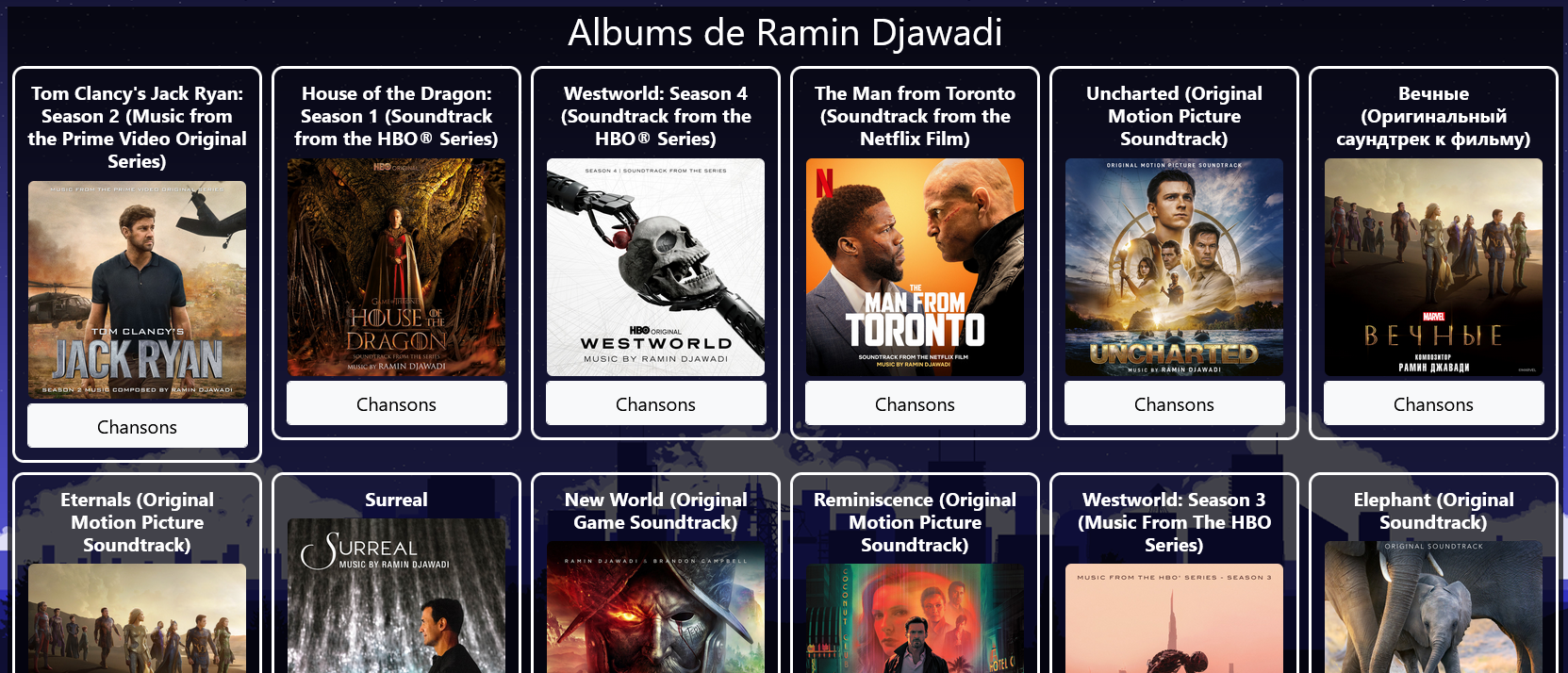


# Page des albums

Une page permettra d’afficher les albums d’un artiste. Nous récupérerons les albums à partir d’une requête à l’API de Spotify. Vous devrez afficher le nom et une image de chaque album.

Le nom de l’artiste (ou son ID Spotify) devra être fourni en paramètre dans le routage du composant.





# Page des chansons

Une page permettra d’afficher les chansons d’un album. Nous récupérerons les chansons à partir d’une requête à l’API de Spotify. Vous devrez seulement afficher le nom de chaque chanson.

Lors du clic sur une chanson, une requête à YouTube sera effectuée pour avoir accès à la chanson. Pour la requête, il est suggéré de placer le **nom de la chanson** et **l’artiste** dans la recherche. (Juste le nom de la chanson donne parfois un mauvais résultat !)

Une fois la requête effectuée, il sera possible d’afficher une vidéo YouTube de la chanson. Pour ce faire, nous utiliserons le Id de la vidéo et un iFrame.

Au minimum, le id Spotify de l’album devra être fourni en paramètre dans le routage du composant.





# Correction

|  |  |  |
| --- | --- | --- |
| Spot-moi 🎵 | | |
| Pour chaque item (Artiste, Album et Chanson) | | |
|  | 1 pt | Classe présente (Pas obligé pour chanson, mais possible) |
|  | 3 pts | Requête HTTP et lecture du JSON |
|  | 2 pts | Afficher une liste d'items (avec image pour les artistes et albums) |
|  | 1 p | Un composant par type d'item |
|  | 1 pt | Routage fonctionnel pour afficher les items |
| Total | 24 pts (8 pts x 3) | |
| Concerts | | |
|  | 3 pts | Requête HTTP et lecture du JSON |
|  | 1 pt | Composant pour afficher les concerts |
|  | 1 pt | Message spécial si l’artiste n’a aucun concert |
|  | 1 pt | Routage fonctionnel pour afficher les items |
|  | 2 pts | Afficher les concerts (date facilement **lisible**, ville, pays) |
|  | 5 pts | Afficher des marqueurs pour les concerts dans **une seule** carte Google |
| Total | 13 pts | |
| Autres consignes | | |
|  | 3 pts | Utiliser au moins un service pour les requêtes HTTP |
|  | 1 pt | Le service devrait retourner des Promise<???> |
|  | 1 pt | Les composants ne contiennent pas de requêtes HTTP. (Services !) | |
|  | 3 pts | La liste d'artistes doit être sauvegardée dans le localStorage |
|  | 2 pts | La liste devrait être chargée lorsque l'on revient sur l'application |
|  | 5 pts | L'application doit être disponible en anglais et en français |
|  | 5 pts | On peut cliquer sur une chanson pour lancer une requête à Youtube et afficher une vidéo pour jouer la chanson. |
|  | 3 pts | Utilisation raisonnable de Git (Inviter l’enseignant, commits clairs, respect des conventions du département et au moins 8 commits cohérents) |
| Total | 23 pts |  |
| Grand total | | 60 points |

# URLS et doc

YouTube Search API

<https://developers.google.com/youtube/v3/docs/search/list>

Example d’utilisation

[https://www.googleapis.com/youtube/v3/search?part=id&maxResults=1&key={API\_KEY}&q={MA\_RECHERCHE}](https://www.googleapis.com/youtube/v3/search?part=id&maxResults=1&key=%7bAPI_KEY%7d&q=%7bMA_RECHERCHE%7d)

Vous devrez vous-même créer une clé d’API pour Google / YouTube.

BandsInTown

<https://app.swaggerhub.com/apis/Bandsintown/PublicAPI/3.0.0#/artist%20events/artistEvents>

Clé pour utiliser l’API de BandsInTown

**2b32475766802ac01eefda45e9e42ea0**

<https://rest.bandsintown.com/artists/MON_ARTISTE/events?app_id=API_KEY>

Spotify API

<https://developer.spotify.com/documentation/web-api/reference/#/>

Exemple d’utilisation de l’API dans les notes de cours et le labo 4.

Vous devrez vous-même créer une clé d’API pour Spotify.

# Par où commencer ?

La réalisation d’un projet Angular avec plusieurs composants, modèles et services peut être intimidant. Voici une séquence suggérée (loin d’être incontournable) pour avancer dans le projet :

1. **Créer tous les composants** dont vous aurez besoin. (Semaine 3)
   * Le composant App servira seulement de squelette pour la page. (header, footer, ..) Donc comptez un composant de plus pour chacune des pages dont vous aurez besoin.
   * Configurez le routage de votre application. Êtes-vous capable d’accéder à tous les composants que vous avez créés ?
2. **Préparer des service**s qui serviront à implémenter vos requêtes et stocker certaines données. (Semaine 3)
   * À combien d’API ferez-vous des requêtes ? Cela pourrait influencer le nombre de services pertinents à préparer.
3. **Préparer des modèles** pour encapsuler les données. (Semaine 3)
   * Quel type de données aurez-vous à afficher dans vos pages Web ? Vous pourriez déjà préparer les classes de vos modèles en conséquence.
4. **Rendre vos composants fonctionnels**, un à la fois. (Semaine 4, mais un peu 5 aussi pour Google Maps et Youtube)
   * Le gros du travail ! Un composant à la fois, compléter l’interface HTML puis rendre fonctionnels les formulaires et requêtes qui permettent à l’utilisateur d’interagir avec la page.
   * Généralement, la dynamique ressemble à ceci quand on charge une page :
     1. Un paramètre de route contient le nom de l’artiste / l’album à chercher. On tente de récupérer ce paramètre.
     2. On lance une requête à l’API approprié grâce au paramètre de route reçu.
     3. On obtient les données dans un objet JSON et on les affiche.
5. **Internationaliser**. (Semaine 4)
   * Si tout fonctionne et que vous êtes satisfait(e) de votre application, c’est l’heure de tout traduire !

Certains développeurs auraient préféré créer et compléter un composant à la fois plutôt que de tous les préparer dès le début. (Dans un projet gigantesque, on ne va tout de même pas préparer 50 composants d’un seul coup !)

De même, certains auraient peut-être inversé certaines des étapes citées. À vous de voir ! Bonne chance !