Rapport de Projet - SpotYnov

**Application de gestion de groupes musicaux avec Spotify API Date de rendu : 10/05/2025**

**Auteurs : Matt, Thomas et Hugo**

**Projet réalisé dans le cadre du module « Développement API »**

# Présentation

Spotynov est une application API développée avec NestJS et Spotify Web API, dont le but est de regrouper les utilisateurs autour de leurs goûts musicaux. En se basant sur les morceaux likés ou écoutés, l’application propose des analyses de personnalité, ainsi que des fonctionnalités collaboratives comme les groupes et la synchronisation musicale.

## Objectifs du projet :

* Mise en place d’un système d’authentification sécurisé via Spotify OAuth2
* Gestion des groupes musicaux avec des interactions simplifiées
* Analyser les goûts musicaux et synchroniser la lecture entre membres d’un même groupe
* Offrir la possibilité de générer automatiquement des playlists personnalisées sur Spotify

**Choix Techniques Justifiés**

**NestJS**Nous avons utilisé NestJS pour sa structure et son intégration avec TypeScript. Ca nous a permis de séparer proprement les domaines fonctionnels :

* authentification,
* gestion des utilisateurs,
* Spotify,
* groupes, …

Son système nous a beaucoup aidés à maintenir un code lisible et testable, ce qui était crucial vu le nombre de fonctionnalités à intégrer…

**JWT**Le JWT a été utilisé pour sécuriser l’accès aux routes et permettre une authentification sécurisée.

L'intégration s’est faite naturellement avec le module Passport, qui est recommandé dans la doc de NestJS. On l’a trouvé simple à prendre en main, et surtout super pratique pour mettre en place une sécurité claire, on l'a d'ailleurs réutilisé dans tout le projet.

Une fois connecté, chaque utilisateur reçoit un token qu’il peut utiliser pour interagir avec les endpoints protégés, notamment ceux liés à Spotify ou aux groupes. Cela nous a permis de gérer facilement les autorisations et les accès.

**Spotify Web API**L’API Spotify est au cœur du projet. Elle a permis de :

* Authentifier les utilisateurs via OAuth2,
* Récupérer leurs données (titres likés, profil, morceaux écoutés récemment),
* Analyser leur personnalité musicale,
* Synchroniser la lecture entre plusieurs comptes,
* Créer dynamiquement des playlists personnalisées.

Elle a donc été utilisée de façon avancée et variée.

**Docker**Docker a permis de standardiser notre environnement de développement. Grâce au docker-compose, nous avons pu lancer simultanément le backend (NestJS) et le frontend (Vite) dans des conteneurs isolés. Cela a grandement facilité :

* Le travail en équipe sans conflits,
* et une meilleure stabilité lors du développement, sans peur des “ça marche chez moi mais pas chez toi”.

**Documentation des Endpoints – SpotYnov API**

**Connexion utilisateur**

*Méthode :* **POST**

*Route :* **/auth/login**

*Description :* **Connexion utilisateur**

*Corps attendu :*

**- username: string**

**- password: string**

*Exemple de réponse :*

**- 200 OK**

**- { "access\_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6InRlc3RvdSIsImlhdCI6MTc0Njg4NzMwNywiZXhwIjoxNzQ2ODkwOTA3fQ.FQjzLBJLiM\_1\_GS1tGd99ft4vMG\_cw2NH2e3EvrGis0"**

**}**

**Récupérer le profil utilisateur**

*Méthode* **: GET**

*Route :* **/auth/profile**

*Description :* **Récupérer le profil utilisateur**

*Authentification requise :* **✅**

*Exemple de réponse :*

**- 200 OK**

**- { "username": "hgrs" }**

**Déconnexion utilisateur**

*Méthode :* **POST**

*Route :* **/auth/logout**

*Description :* **Déconnexion utilisateur**

*Authentification requise :* **✅**

*Exemple de réponse :*

**- 200 OK**

**- { "message": "Déconnexion réussie. Veuillez supprimer votre token JWT côté client." }**

**Créer un utilisateur**

*Méthode :* **POST**

*Route :* **/users**

*Description :* **Créer un utilisateur**

*Corps attendu :*

**- username: string**

**- password: string**

*Exemple de réponse :*

**- 201 Created**

**- { "username": "thomas" }**

**Récupérer tous les utilisateurs**

*Méthode :* **GET**

*Route :* **/users**

*Description :* **Récupérer tous les utilisateurs**

*Authentification requise :* **❌**

*Exemple de réponse :*

**- 200 OK**

**- [ { "username": "hugo", ... } ]**

**Récupérer un utilisateur par son Username**

*Méthode :* **GET**

*Route :* **/users/{username}**

*Description :* **Récupérer un utilisateur par son Username**

*Authentification requise :* **❌**

*Exemple de réponse :*

**- 200 OK**

**- { "username": "thomas" }**

**Mettre à jour un utilisateur**

*Méthode :* **PATCH**

*Route :* **/users/{username}**

*Description :* **Mettre à jour un utilisateur**

*Authentification requise :* **❌**

*Corps attendu :*

**- username?: string**

**- password?: string**

**- groupName?: string**

*Exemple de réponse :*

**- 200 OK**

**- { "username": "thomas2" }**

**Supprimer un utilisateur**

*Méthode :* **DELETE**

*Route :* **/users/{username}**

*Description* **: Supprimer un utilisateur**

*Authentification requise :* **❌**

*Exemple de réponse :*

**- 200 OK**

**- { "message": "Utilisateur supprimé" }**

**Créer un groupe**

*Méthode :* **POST**

*Route :* **/groups/create**

*Description :* **Créer un groupe**

*Authentification requise :* **✅**

*Corps attendu :*

**- groupName: string**

*Exemple de réponse :*

**- 201 Created**

**- { "message": "Groupe créé", "admin": "thomas2" }**

**Rejoindre un groupe**

*Méthode :* **POST**

*Route :* **/groups/join**

*Description :* **Rejoindre un groupe**

*Authentification requise :* **✅**

*Corps attendu :*

**- groupName: string**

*Exemple de réponse :*

**- 200 OK**

**- { "message": "Rejoint le groupe avec succès" }**

**Quitter un groupe**

*Méthode :* **POST**

*Route :* **/groups/leave**

*Description :* **Quitter un groupe**

*Authentification requise :* **✅**

*Exemple de réponse :*

**- 200 OK**

**- { "message": "A quitté le groupe avec succès" }**

**Lister tous les groupes**

*Méthode :* **GET**

*Route :* **/groups**

*Description :* **Lister tous les groupes**

*Authentification requise :* **❌**

*Exemple de réponse :*

**- 200 OK**

**- [ { "groupName": "groupe1", "userCount": 2 } ]**

**Obtenir les détails d'un groupe**

*Méthode :* **GET**

*Route :* **/groups/{groupName}**

*Description :* **Obtenir les détails d'un groupe**

*Authentification requise :* **✅**

*Exemple de réponse :*

**- 200 OK**

**- { "groupName": "groupe1", "members": [hugo, …] }**

**Génère l'URL de connexion Spotify**

*Méthode :* **GET**

***Route :* /spotify/login**

*Description :* **Génère l'URL de connexion Spotify**

*Authentification requise* **: ✅**

***Exemple de réponse :***

**- 200 OK**

**- { "url":**

**"https://accounts.spotify.com/authorize?client\_id=5996e16cdba64f768b013901df287254&response\_type=code&redirect\_uri=http%3A%2F%2Flocalhost%3A3000%2Fapi%2Fspotify%2Fcallback&scope=user-read-email%20user-read-private%20user-read-recently-played%20user-top-read%20user-read-playback-state%20user-library-read%20user-modify-playback-state"**

**}**

**Callback Spotify pour échanger le code contre token**

*Méthode :* **GET**

*Route :* **/spotify/callback**

*Description :* **Callback Spotify pour échanger le code contre token**

*Authentification requise :* **✅**

*Exemple de réponse :*

**- 200 OK**

**- { "access\_token": "BQCypKjnlRR4TzsP4vMZy-MjjFpT3spi5VfHVWMPjIeMtU6SJt7Cdur5f08swNXbg0VpGjQYwIfkD1\_4KgIvdnJMyZ7V9eEoM-M7-FRQNGBZIUDfUl65fA-JFVR1xjL38y3RMGLOH-YP5kUGxB8vzXTRj3x5i7uK5RuMFWkrBRTG4PvJkEzz0k7yL-5-7JKpYQAvoq4YLxMXnKwxi8lnRLDokfDc3PDtlaltjeSBPO8q6G2LUy7TUX9VRiXURBY4VWrzzKRlE9QCYx\_6o4JzaDc",**

**"refresh\_token": "AQBRsbfChyw241Kf\_k8cwZXQzftchMeak5Dv9\_xMHOTYFE2AVUcjJEy-k40GOko6KWMpUmMtOUQmRn6Q1JwHPvySgmGNTwI-FjGZtFsiDct3QJ3CAM9wcoG3etzDyB\_hU-s"**

**}**

**Analyse de personnalité de soi-même**

*Méthode :* **GET**

*Route :* **/spotify/personality/me**

*Description :* **Analyse de personnalité de soi-même**

*Authentification requise :* **✅**

*Exemple de réponse :*

**- 200 OK**

**- { "averagePopularity": 54.2, "averageDurationMs":54000 }**

**Analyse de personnalité Spotify d’un utilisateur spécifique**

*Méthode :* **GET**

*Route :* **/spotify/personality/{username}**

*Description :* **Analyse de personnalité Spotify d’un utilisateur spécifique**

*Authentification requise :* **✅**

*Exemple de réponse :*

**- 200 OK**

**- { "averagePopularity": 54.2, "averageDurationMs": 54000 }**

**Synchronise la musique actuelle de l’admin sur les membres du groupe**

*Méthode :* **POST**

*Route :* **/spotify/sync**

*Description :* **Synchronise la musique actuelle de l’admin sur les membres du groupe**

*Authentification requise :* **✅**

*Exemple de réponse :*

**- 200 OK**

**- { "message": "Synchronisation réussie" }**

**Conclusion**

Tout au long du projet SpotYnov, notre équipe a su s’organiser simplement. Nous nous sommes réparti les rôles et structuré les tâches à effectuer. À chaque blocage, nous avons su résoudre, que ce soit en revoyant la logique, en retravaillant nos appels API à Spotify ou en nous aidant de la doc.

Ce projet nous a surtout permis de monter en compétence sur NestJS, un framework que nous ne connaissions pas au départ (pour la plupart), mais qui s’est révélé puissant et plutôt intuitif.

Nous avons également pu un peu plus nous familiariser avec Docker, en utilisant les conteneurs pour gérer proprement nos environnements backend et frontend, ce qui a été un vrai gain de temps, surtout concernant travail d’équipe...

Même si l’interface utilisateur est restée en second plan, nous avons atteint notre objectif principal : livrer une API complète, robuste, bien documentée, et répondant parfaitement aux besoins fonctionnels définis au départ.