



**ECOLE MAROCAINE DES
SCIENCES DE L'INGENIEUR**
Membre de
HONORIS UNITED UNIVERSITIES



Rapport de projet

3ème année Ingénierie Informatique et Réseaux

Sous le thème

TuneX

Web App de Streaming Musical

Programmation Python et Framework

Élèves :

BENOUARI Badr-Eddine

IBNOUCHEIKH Ilyass

Encadrant :

Pr. K. NAFIL

Année Universitaire 2024-2025

Résumé

TuneX est une plateforme web de streaming musical développée sous Django, conçue pour offrir une expérience d'écoute fluide, personnalisée et collaborative. Le projet s'appuie sur une architecture en cascade pour structurer les phases de spécification, conception, développement, tests et déploiement.

TuneX propose :

- Un moteur de streaming en continu avec contrôle de lecture (play, pause, suivant) ;
- Un système de création, modification de playlists personnalisées ;
- Une interface responsive et ergonomique, accessible sur desktop et mobile.

La modélisation UML, comprenant les diagrammes de cas d'utilisation, de séquence et de classes, a permis de formaliser le fonctionnement du système avant son implémentation. Les tests unitaires et fonctionnels garantissent la robustesse et la performance de l'application. TuneX se positionne comme une alternative moderne aux plateformes existantes, avec des perspectives d'évolution vers un mode hors-ligne, une application mobile native et l'intégration de techniques d'intelligence artificielle.

Mots clés : streaming musical, Django, UML, responsive, playlist.

Abstract

TuneX is a Django-based web music streaming platform designed to provide a seamless, personalized, and collaborative listening experience. The project follows a waterfall methodology, structuring phases from requirements specification and design to development, testing, and deployment.

Key features include :

- Continuous streaming with playback controls (play, pause, skip) ;
- Creation, editing, and sharing of customized playlists ;
- A responsive, user-friendly interface for both desktop and mobile devices.

UML modeling—comprising use case, sequence, and class diagrams—was employed to formalize system behavior prior to implementation. Comprehensive unit and functional tests ensure application robustness and performance. TuneX positions itself as a modern alternative to existing services, with future enhancements planned such as offline mode, native mobile applications, and AI-powered recommendation algorithms.

Keywords : music streaming, Django, UML, responsive design, playlists.

Table des matières

1	Contexte général	8
1.1	Introduction	8
1.2	Présentation du projet	8
1.2.1	Étude de l'existant	8
1.2.2	Critique de l'existant	8
1.2.3	Solution proposée	9
1.3	Organisation du projet	9
1.3.1	Processus de développement	9
1.3.2	Planification opérationnelle	10
1.4	Conclusion	11
2	Analyse et Conception	12
2.1	Introduction	12
2.2	Description des besoins fonctionnels	12
2.2.1	Périmètre de projet	12
2.2.2	Description de la structure générale du site	12
2.2.3	Fonctionnalités de projet	13
2.3	Description des besoins ergonomiques	13
2.3.1	Définir les règles ergonomiques	13
2.3.2	Formaliser une charte ergonomique	13
2.4	Description des besoins graphiques	13
2.5	Descriptions des besoins techniques	13
2.5.1	Outils de développement	13
2.6	Modélisation UML	14
2.6.1	Vue Fonctionnelle (diagrammes de cas d'utilisation)	15
2.6.2	Vue Logique (diagramme de classes)	16
2.6.3	Diagramme de sequence (Search Music)	17
2.6.4	Diagramme de sequence (Login)	17
2.6.5	Diagramme de sequence (Add Song To Playlist)	18
2.7	Implémentation	19
2.7.1	Maquettes	19
2.8	Conclusion	20
3	Interfaces de l'application	21

3.1	Introduction	21
3.2	Interfaces	21
3.3	Conclusion	26

Table des figures

1.1	Modèle en cascade (waterfall)	9
1.2	Le diagramme de Gantt	10
2.1	Diagramme des cas d'utilisation	15
2.2	Diagramme des classes	16
2.3	Diagramme de sequence (Search Music)	17
2.4	Diagramme de sequence (Login)	17
2.5	Diagramme de sequence (Add Song To Playlist)	18
2.6	Maquette illustrant l'agencement des sections : header, recommandations, tendances, footer	19
3.1	Page de connexion (Login)	21
3.2	Page d'inscription (Register)	22
3.3	Page d'accueil	22
3.4	Page de recherche	23
3.5	Espace utilisateur	23
3.6	Page de création	24
3.7	Page des musiques aimées	24
3.8	Contenu d'une playlist	25
3.9	Lecteur audio	25

Liste des abréviations

- UML : Unified Modeling Language
- API : Application Programming Interface
- HTML5 : HyperText Markup Language version 5
- CSS : Cascading Style Sheets

Introduction Générale

Le secteur du streaming musical est aujourd'hui dominé par quelques acteurs majeurs, tels que Spotify, Deezer ou Apple Music. Ces plateformes proposent des catalogues vastes et des fonctionnalités avancées, mais elles ne répondent pas toujours aux attentes de tous les utilisateurs. En particulier, la personnalisation des recommandations peut manquer de précision, et l'ergonomie de l'interface peut parfois sembler complexe pour les néophytes ou les publics moins technophiles. Par ailleurs, les délais de chargement et la qualité sonore, bien qu'en constante amélioration, constituent encore des points d'attention pour offrir une expérience véritablement fluide et immersive.

Face à ces constats, le projet TuneX se donne pour ambition de développer une plateforme de streaming musical web, intuitive et performante. Conçue avec le framework Django, TuneX se veut à la fois accessible et riche en fonctionnalités. Les utilisateurs pourront non seulement écouter leurs morceaux préférés en streaming, mais aussi créer et organiser des playlists de manière simple. L'objectif est de proposer une alternative moderne, à la fois légère et évolutive, qui place l'utilisateur au centre de l'expérience musicale.

Le choix de modèle en cascade a permis de structurer rigoureusement le développement en phases successives : de la définition précise des besoins à la conception, puis au codage et enfin aux tests et à la mise en production. Cette approche garantit une traçabilité optimale et une validation progressive à chaque étape du projet.

Ce rapport est organisé en trois chapitres clairs :

- **Contexte général** : Nous détaillerons également l'organisation du projet, la planification et le processus en cascade retenu pour mener à bien cette réalisation.
- **Analyse et Conception** : Ce chapitre décrira en détail les besoins fonctionnels et non fonctionnels de la plateforme, la modélisation UML du système (cas d'utilisation, diagrammes de classes et de séquence) et le choix technique des outils et technologies.
- **Interfaces de l'application** : Nous y présenterons l'ensemble des écrans développés : page d'accueil, espace d'authentification, espace utilisateur, et lecteur audio, accompagnés d'une description fonctionnelle et ergonomique de chacune de ces interfaces.

Chapitre 1

Contexte général

1.1 Introduction

Ce premier chapitre expose le cadre dans lequel s'inscrit le projet TuneX. Nous y décrivons le marché actuel du streaming musical, les principales lacunes des solutions existantes, ainsi que la proposition de valeur de TuneX. Enfin, nous présentons l'organisation et la méthode de travail retenues pour mener à bien ce projet.

1.2 Présentation du projet

1.2.1 Étude de l'existant

Le marché du streaming musical est largement dominé par des plateformes comme Spotify, Deezer et Apple Music. Bien qu'elles proposent des catalogues vastes et des fonctionnalités variées, plusieurs aspects demeurent perfectibles :

- Personnalisation limitée des recommandations,
- Flexibilité restreinte dans la création de playlists,
- Interfaces parfois complexes pour les nouveaux utilisateurs,
- Accès payant à certaines fonctionnalités clés.

1.2.2 Critique de l'existant

Ces services rencontrent quatre principaux points de frustration pour l'utilisateur :

- Une ergonomie qui peut dérouter les néophytes,
- Des recommandations génériques,
- Des fonctionnalités premium trop restrictives pour les comptes gratuits.

1.2.3 Solution proposée

Afin de répondre à ces besoins, TuneX a été conçu comme :

- Une application web développée avec le framework Django,
- Un lecteur en continu garantissant une qualité sonore optimale,
- Un système de playlists personnalisées simple à créer et modifier,
- Une expérience utilisateur fluide et réactive, accessible sur tous types d'écrans.

1.3 Organisation du projet

1.3.1 Processus de développement

Nous avons opté pour le modèle en cascade, décomposant le projet en phases successives – spécification, conception, développement, tests et déploiement – afin d’assurer une validation rigoureuse à chaque étape.

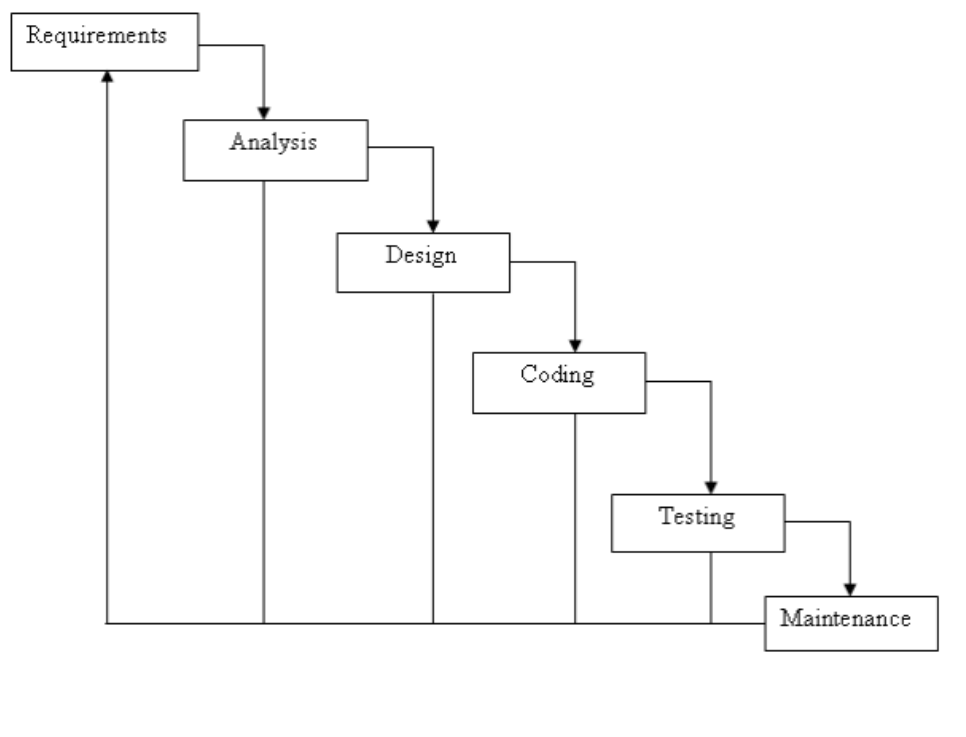


FIGURE 1.1 – Modèle en cascade (waterfall)

1.3.2 Planification opérationnelle

Diagramme de Gantt

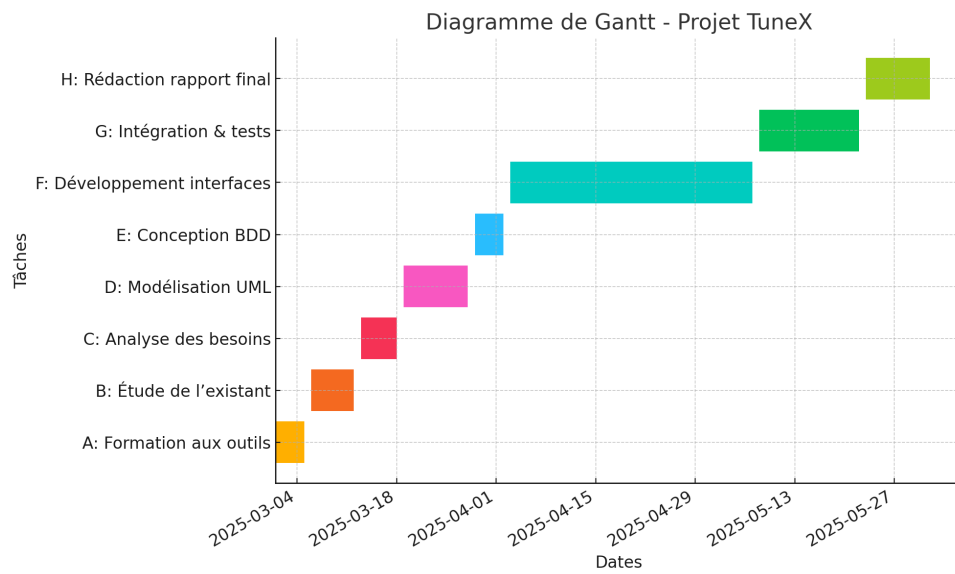


FIGURE 1.2 – Le diagramme de Gantt

1.4 Conclusion

Ce chapitre a dressé un état des lieux du marché du streaming musical, identifié les insuffisances des plateformes actuelles et présenté la solution TuneX. Il a également détaillé l'organisation du projet et la méthode « Modèle en cascade » choisie pour structurer son développement.

Il s'est structuré en deux grandes parties :

- **Étude du marché et analyse de l'existant :**

- Présentation des acteurs dominants (Spotify, Deezer, Apple Music),
- Identification des limites : recommandations peu personnalisées, interfaces complexes, fonctionnalités premium.

- **Solution TuneX et organisation du projet :**

- Description de TuneX : streaming fluide, gestion intuitive des playlists,
- Présentation de modèle en cascade pour structurer le développement,
- Planification opérationnelle détaillée des tâches (formation, étude, analyse, modélisation, développement, tests, rédaction).

L'objectif de ce chapitre était de poser le contexte du projet et de justifier la solution proposée, tout en déployant la feuille de route méthodologique. Cette base solide permet d'enchaîner, dans le chapitre suivant, sur l'analyse fonctionnelle et la conception détaillée de TuneX, en s'appuyant sur les besoins et la planification définis ici.

Chapitre 2

Analyse et Conception

2.1 Introduction

Ce chapitre expose en détail les besoins du projet TuneX, tant sur le plan fonctionnel que technique, ainsi que les exigences ergonomiques et graphiques. Il présente également la modélisation UML qui structure la conception du système.

2.2 Description des besoins fonctionnels

2.2.1 Périmètre de projet

Ce travail a pour objectif de :

- Permettre aux utilisateurs d'écouter de la musique en streaming de manière fluide et continue,
- Offrir un système de création, gestion de playlists personnalisées.

2.2.2 Description de la structure générale du site

- **Pages publiques** : page d'accueil, musique tendance, sélections recommandées,
- **Espace d'authentification** : accès sécurisé par e-mail et mot de passe. Chaque utilisateur peut consulter et modifier à tout moment ses informations personnelles,
- **Espace utilisateur** : gestion des playlists personnelles, moteur de recherche, historique d'écoute,

2.2.3 Fonctionnalités de projet

- **Utilisateur** : Écouter de la musique, créer/modifier playlists.

2.3 Description des besoins ergonomiques

2.3.1 Définir les règles ergonomiques

- Cohérence de la navigation (barre fixe, menus explicites),
- Visibilité des actions principales (lecture, ajout à playlist, J'aime),
- Chargement rapide des pages (< 2 secondes).

2.3.2 Formaliser une charte ergonomique

- Titres et boutons de taille lisible (≥ 16 px),
- Zones cliquables suffisamment larges ($\geq 44 \times 44$ px),
- Indicateur de progression (lecture en cours, chargement).

2.4 Description des besoins graphiques

- **Palette de couleurs** : tons clairs (blanc, gris clair) et accents bleu profond pour les boutons et liens.
- **Typographie** : police sans-serif moderne, hiérarchisation claire des titres et paragraphes.
- **Iconographie** : icônes vectorielles simples et compréhensibles (play, pause, cœur, plus).
- **Illustrations** : affichage optimisé des pochettes d'album et visuels d'artistes.

2.5 Descriptions des besoins techniques

2.5.1 Outils de développement

Une introduction générale aux technologies choisies :

- **Backend** : Django (Python) pour la gestion des utilisateurs, des playlists personnelles et l'interfaçage sécurisé avec l'API Deezer via des appels HTTP. Django garantit également

l'authentification des utilisateurs, la sécurité des échanges et l'encadrement des interactions entre l'utilisateur et les services externes.

- **Frontend** : HTML5, CSS3, et Javascript pour un design responsive et modulable,
- **Base de données** : MySQL, stockage des informations utilisateurs et des playlists personnalisées.
- **Services externe** : API Deezer, récupération des morceaux, informations d'artistes et streaming audio.
- **Serveur et déploiement** : PythonAnywhere, plateforme d'hébergement de l'application web, assurant la gestion des déploiements, des environnements virtuels et des configurations serveur.

2.6 Modélisation UML

Une introduction à l'utilisation d'UML pour structurer la conception :

- **Objectif** : visualiser et formaliser l'architecture du système avant le développement,
- **Avantages** :
 - Descriptions graphiques claires,
 - Vues adaptées à chaque phase (fonctionnelle, dynamique, logique),
 - Mise en évidence rapide des incohérences ou lacunes,
- **Inconvénients** :
 - Peut nécessiter des ajustements en cours de projet,
 - Apprentissage et rigueur requis pour une bonne exploitation.

2.6.1 Vue Fonctionnelle (diagrammes de cas d'utilisation)

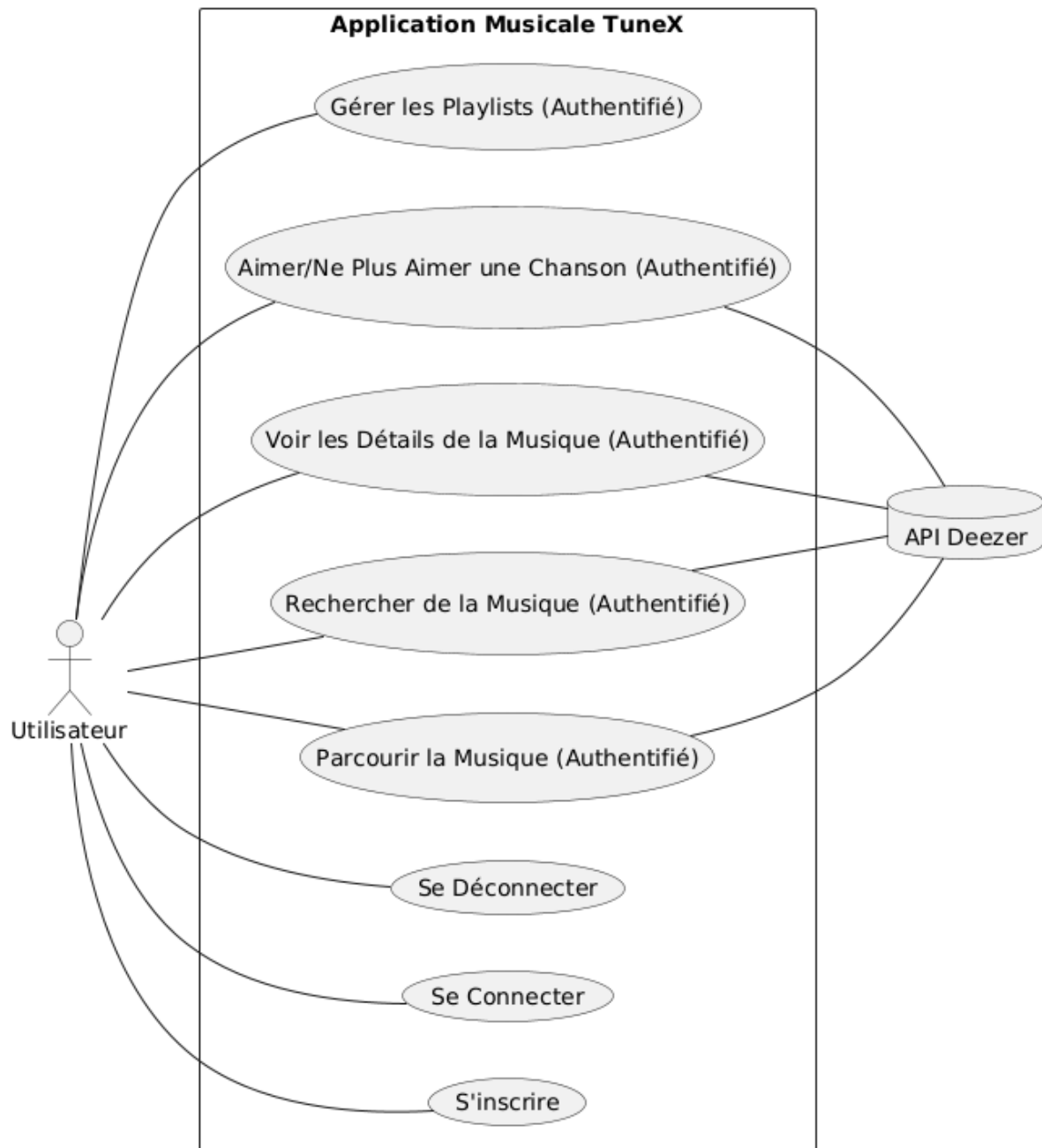


FIGURE 2.1 – Diagramme des cas d'utilisation

2.6.2 Vue Logique (diagramme de classes)

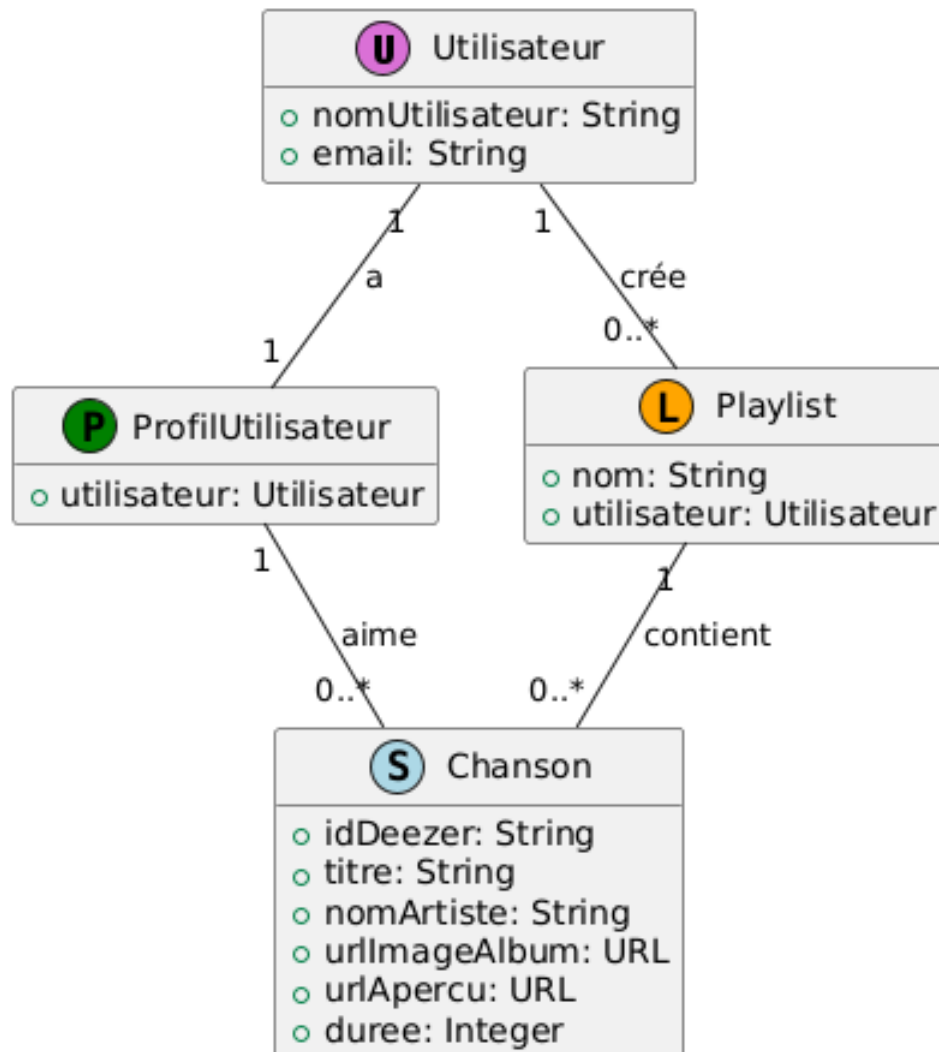


FIGURE 2.2 – Diagramme des classes

2.6.3 Diagramme de sequence (Search Music)

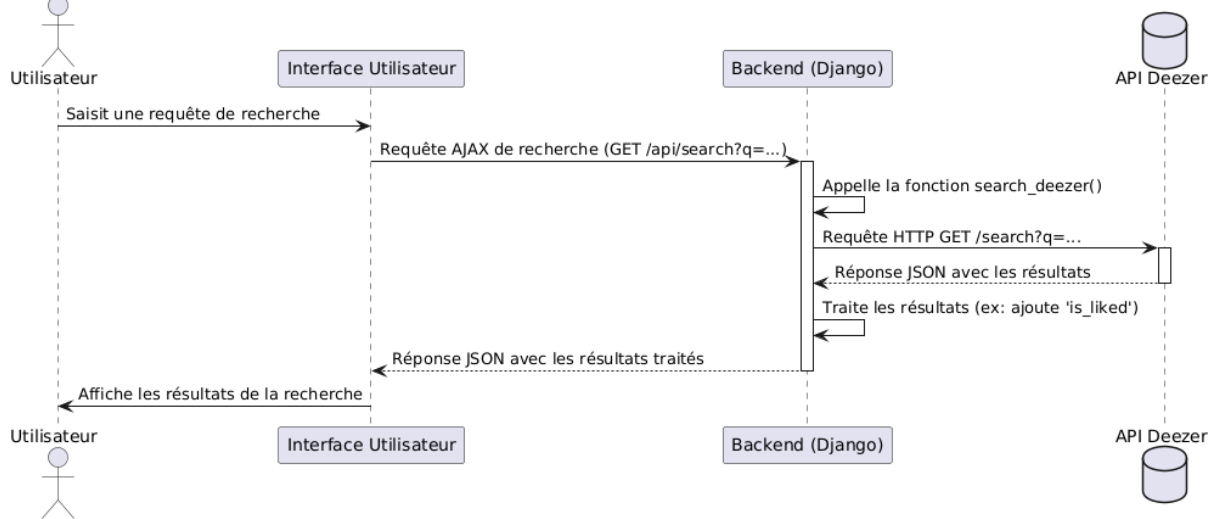


FIGURE 2.3 – Diagramme de sequence (Search Music)

2.6.4 Diagramme de sequence (Login)

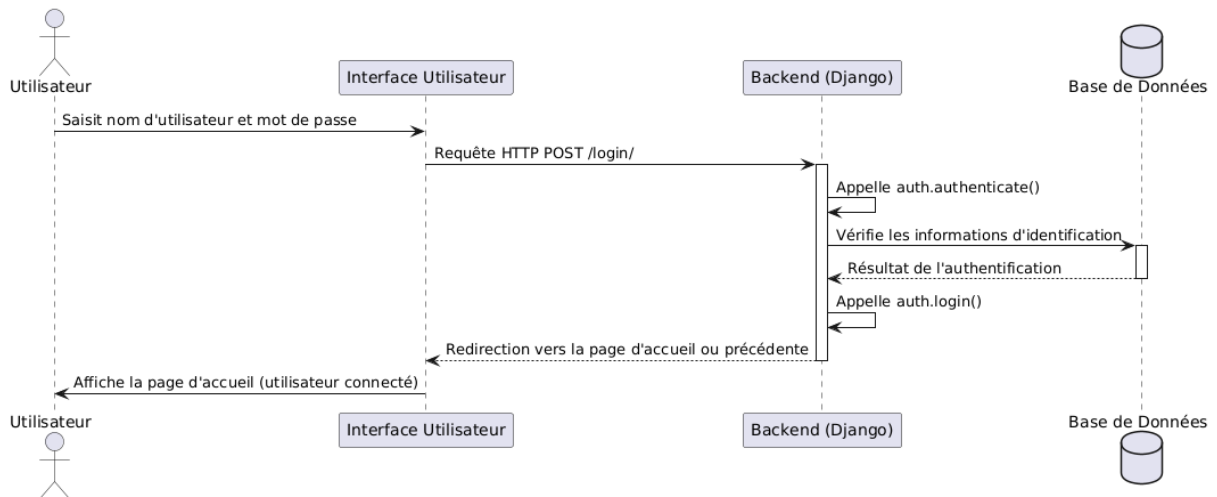


FIGURE 2.4 – Diagramme de sequence (Login)

2.6.5 Diagramme de sequence (Add Song To Playlist)

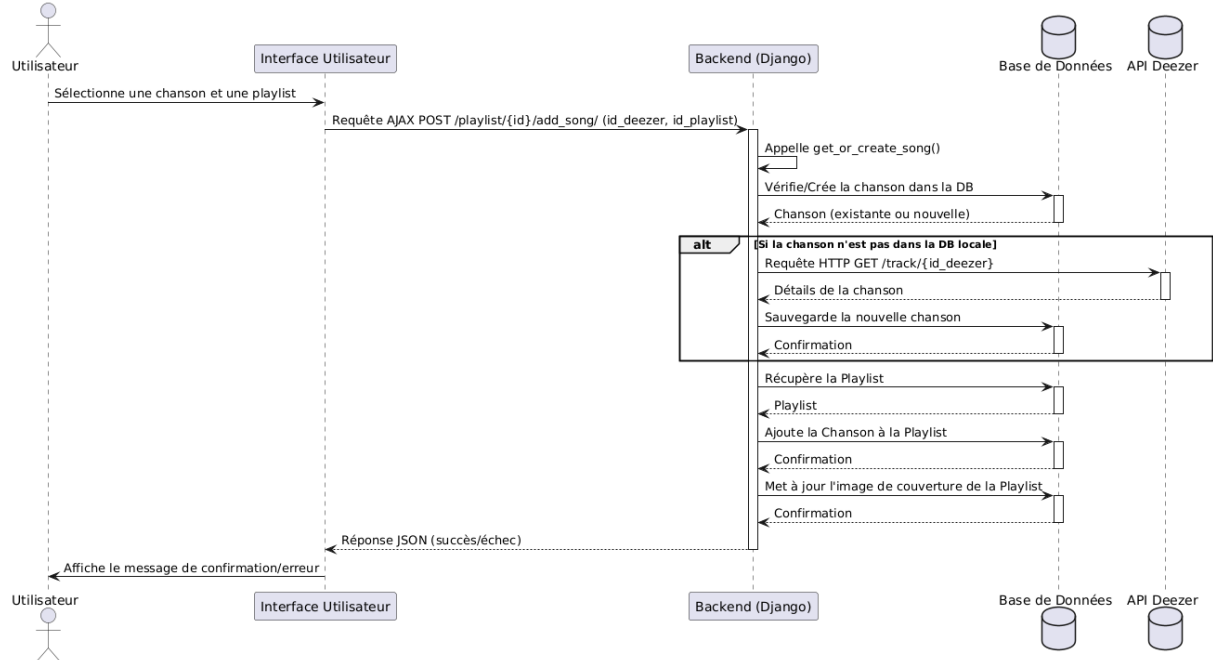


FIGURE 2.5 – Diagramme de sequence (Add Song To Playlist)

2.7 Implémentation

2.7.1 Maquettes

Maquette illustrant l'agencement des sections

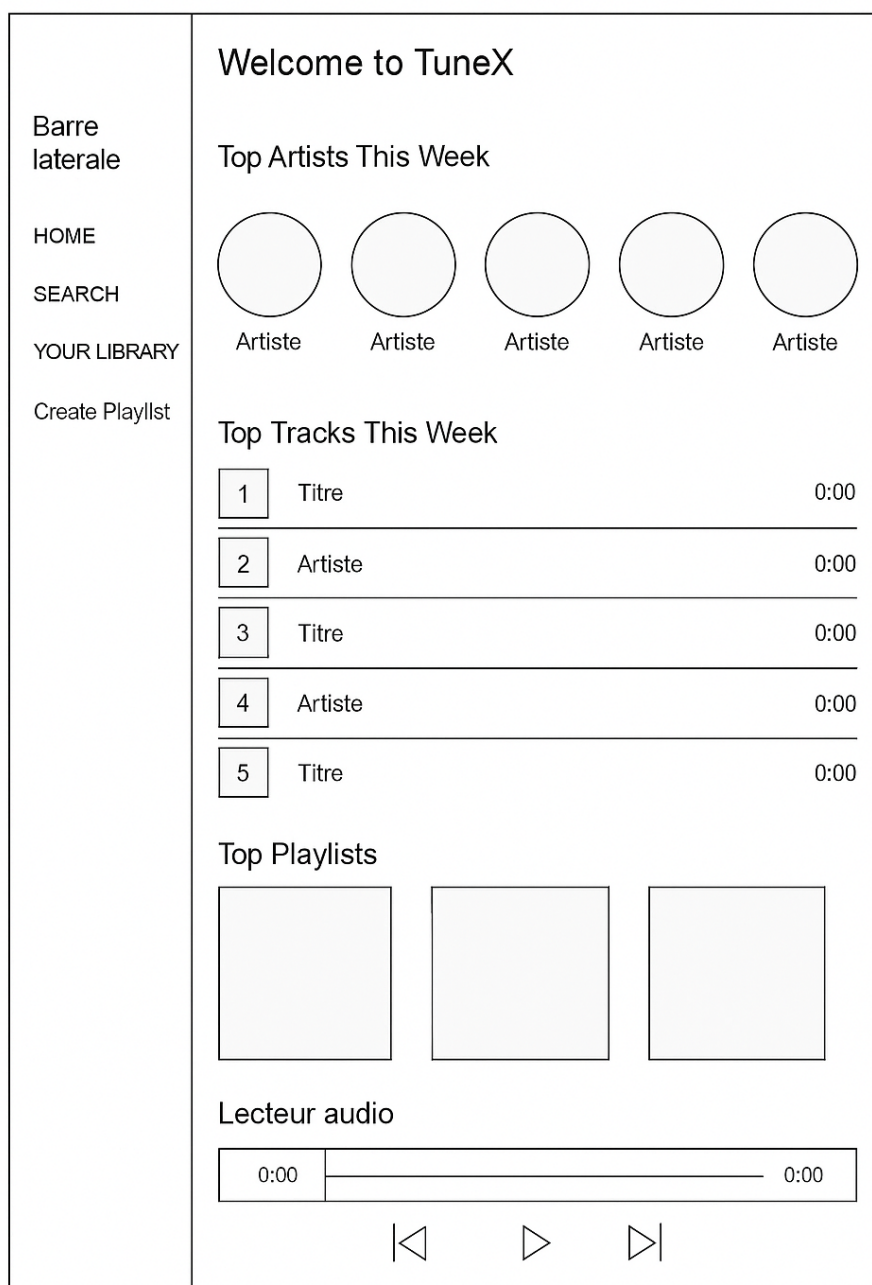


FIGURE 2.6 – Maquette illustrant l'agencement des sections : header, recommandations, tendances, footer

2.8 Conclusion

Nous avons détaillé dans ce chapitre les différentes dimensions de l'analyse et de la conception du projet TuneX :

- Les besoins fonctionnels, exprimés en périmètre, structure et fonctionnalités détaillées,
- Les exigences ergonomiques et la charte associée, garantissant une expérience utilisateur optimale,
- Les contraintes graphiques définissant la charte visuelle et l'iconographie,
- Les aspects techniques, justifiant le choix des technologies et outils,
- La modélisation UML, offrant une vision à la fois fonctionnelle, dynamique et logique du système.

Ces éléments constituent le socle sur lequel repose la phase de développement et de tests, afin d'assurer la cohérence et la qualité de la solution finale.

Chapitre 3

Interfaces de l'application

3.1 Introduction

Ce chapitre décrit les interfaces développées, leurs fonctionnalités et leur organisation.

3.2 Interfaces

- **Page de connexion (Login)** : page permettant à l'utilisateur de saisir ses identifiants pour accéder à l'application.

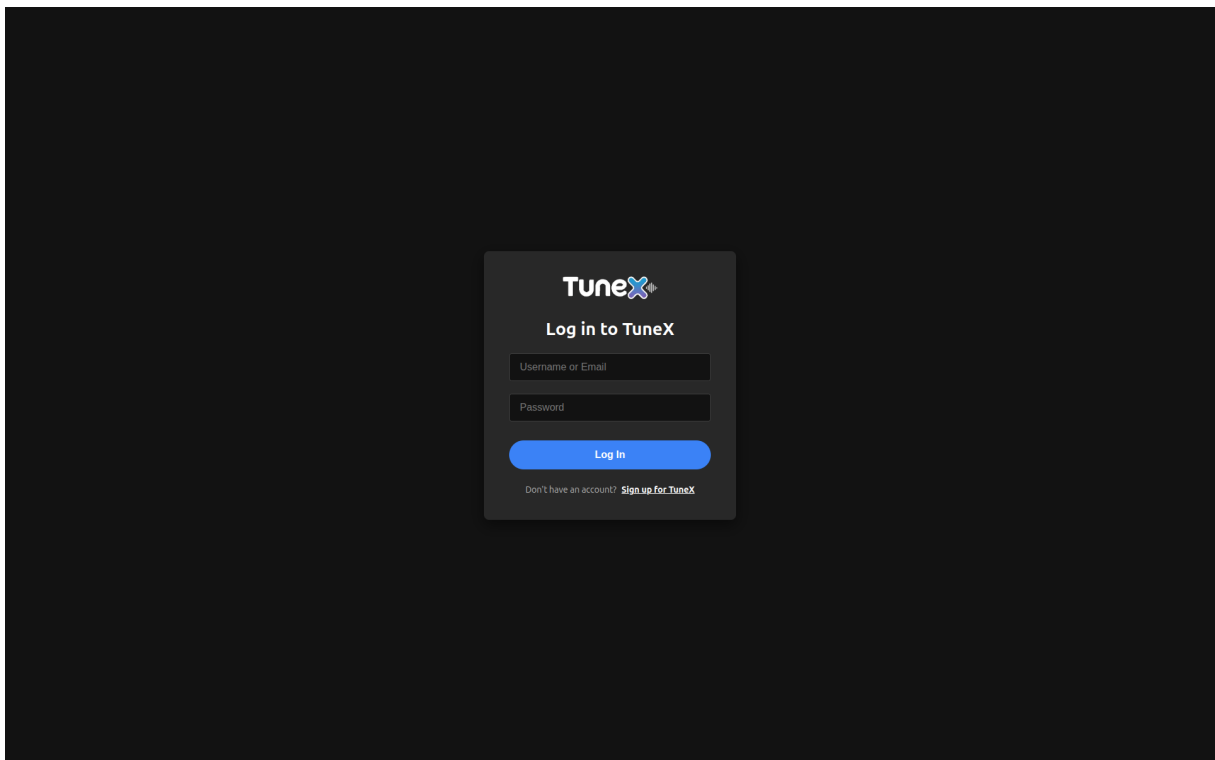


FIGURE 3.1 – Page de connexion (Login)

- **Page d'inscription (Register)** : page permettant à un nouvel utilisateur de créer un compte en fournissant ses informations personnelles et ses identifiants.

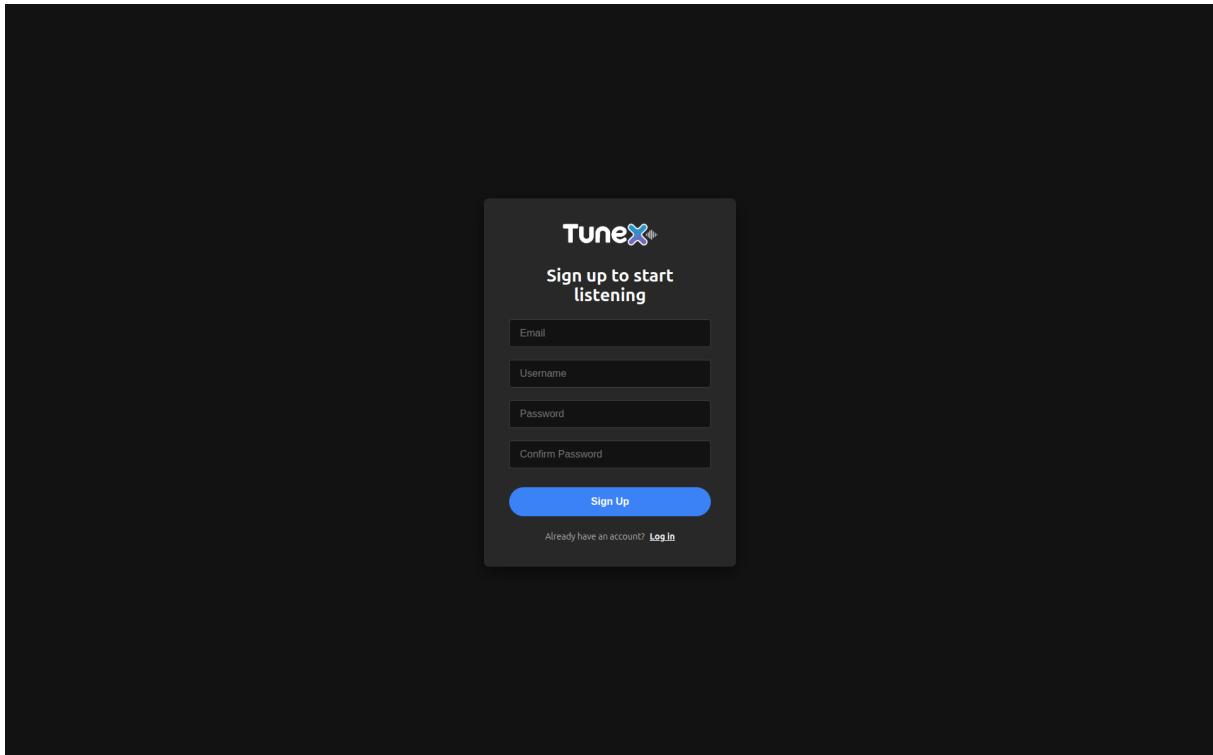


FIGURE 3.2 – Page d'inscription (Register)

- **Page d'accueil** : page principale affichant les artistes et les musiques tendances.

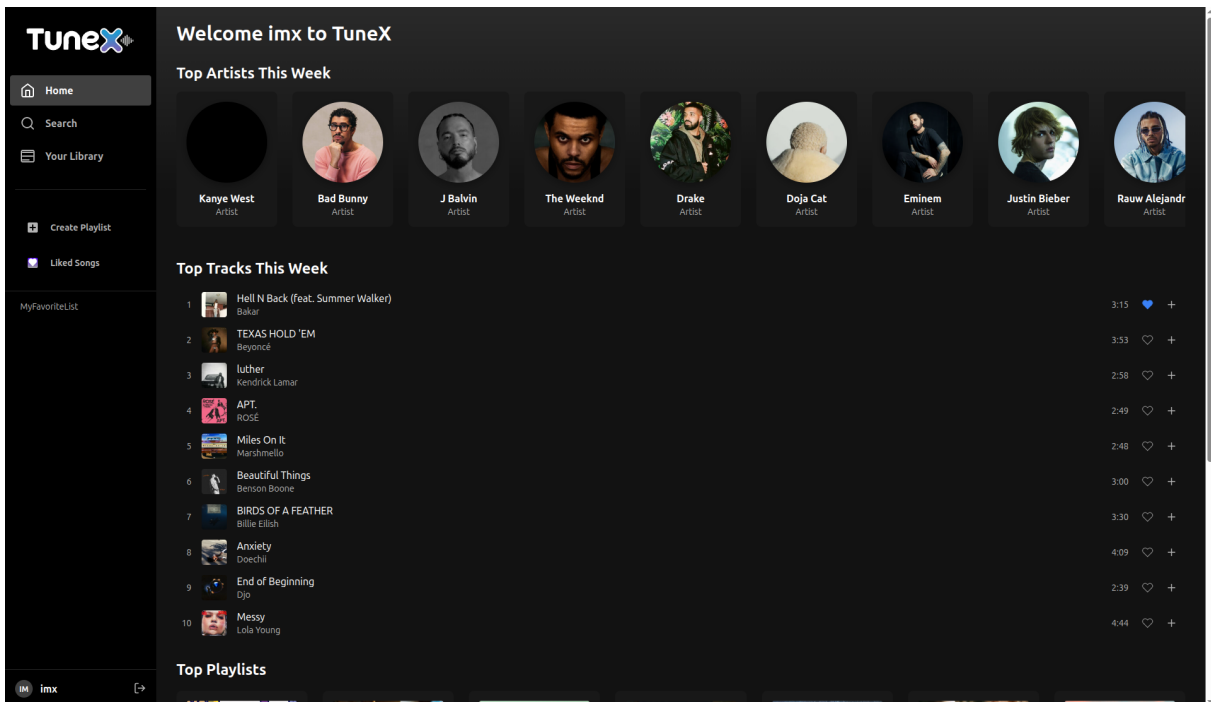


FIGURE 3.3 – Page d'accueil

- **Page de recherche** : interface permettant à l'utilisateur de rechercher des musiques, artistes, albums.

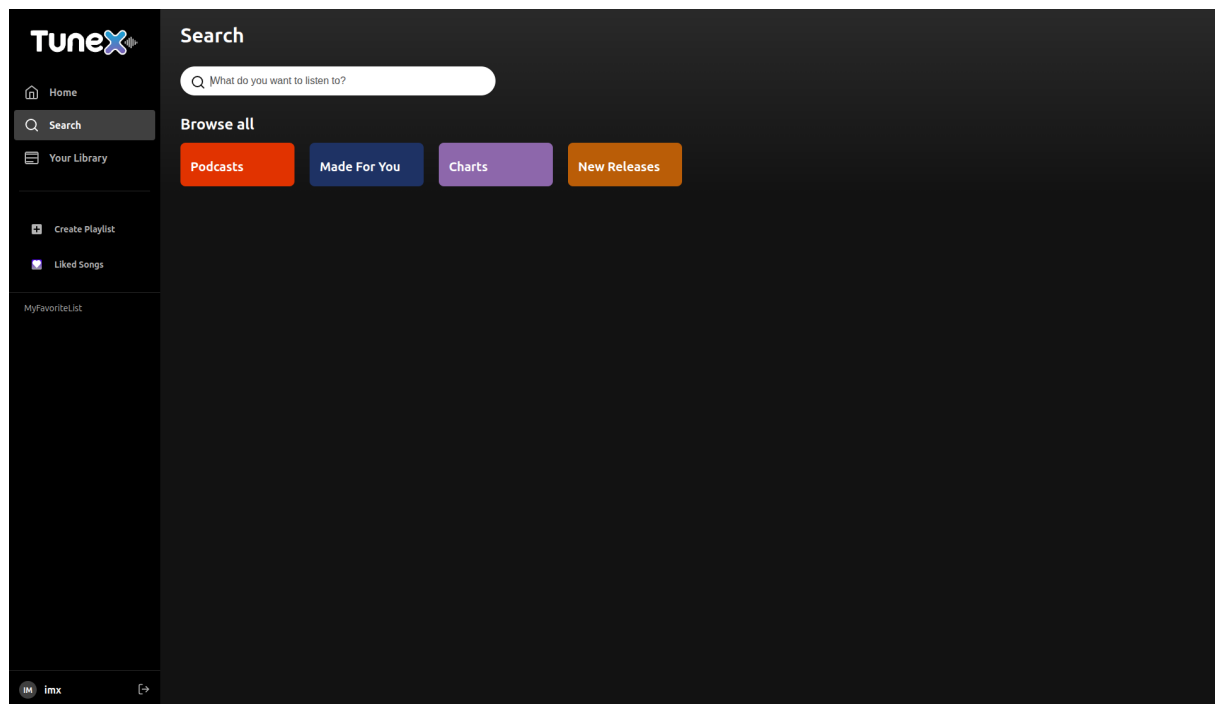


FIGURE 3.4 – Page de recherche

- **Espace utilisateur** : section dédiée à l'utilisateur où il peut gérer ses playlists.

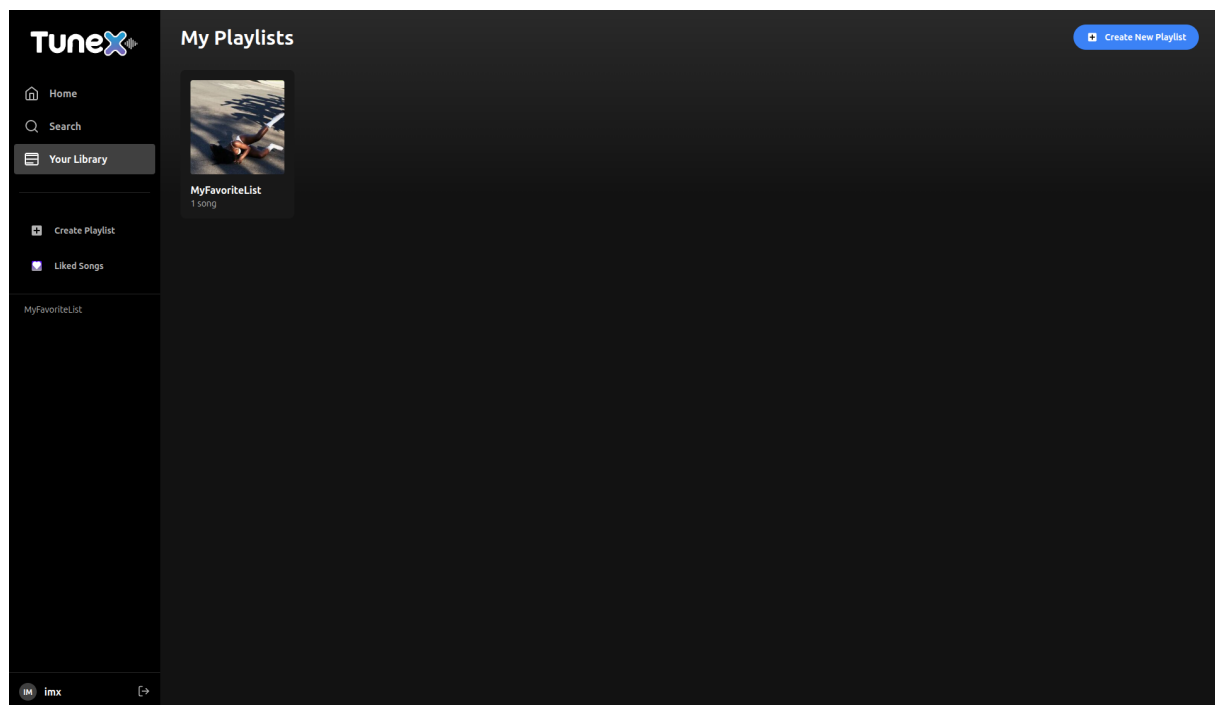


FIGURE 3.5 – Espace utilisateur

- **Page de création** : interface permettant à l'utilisateur d'ajouter une nouvelle playlist.

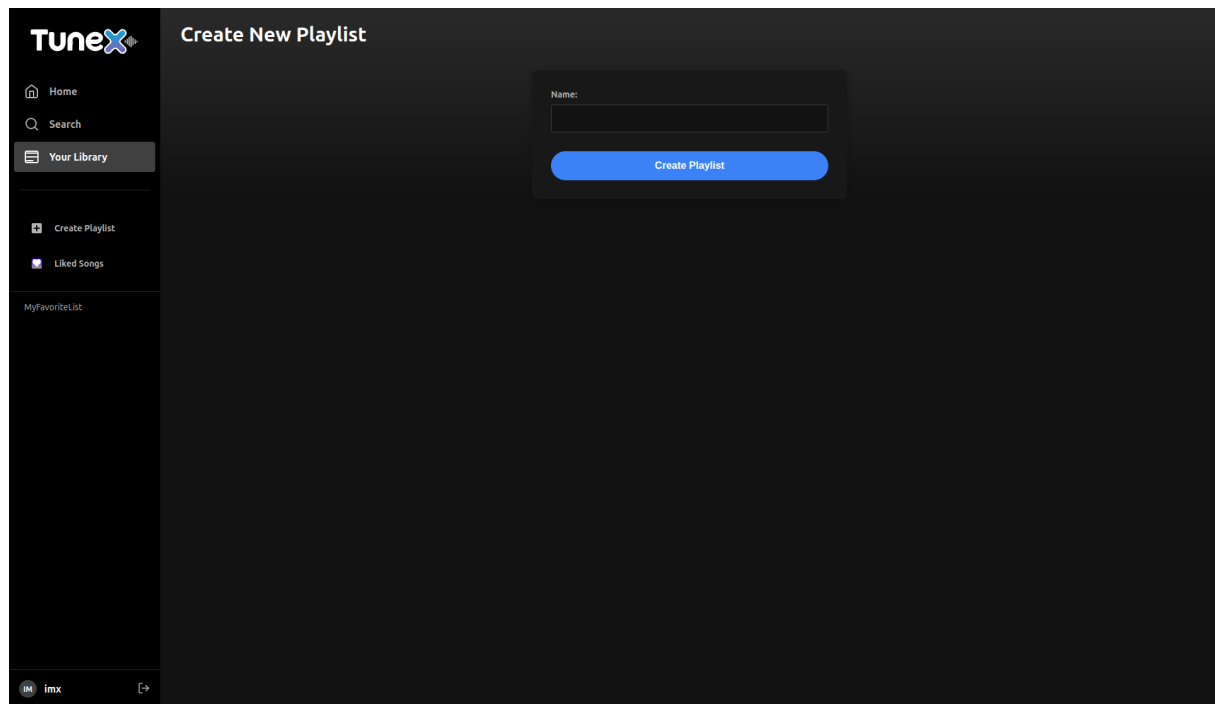


FIGURE 3.6 – Page de création

- **Page des musiques aimées** : liste des morceaux que l'utilisateur a marqués comme favoris ou aimés.

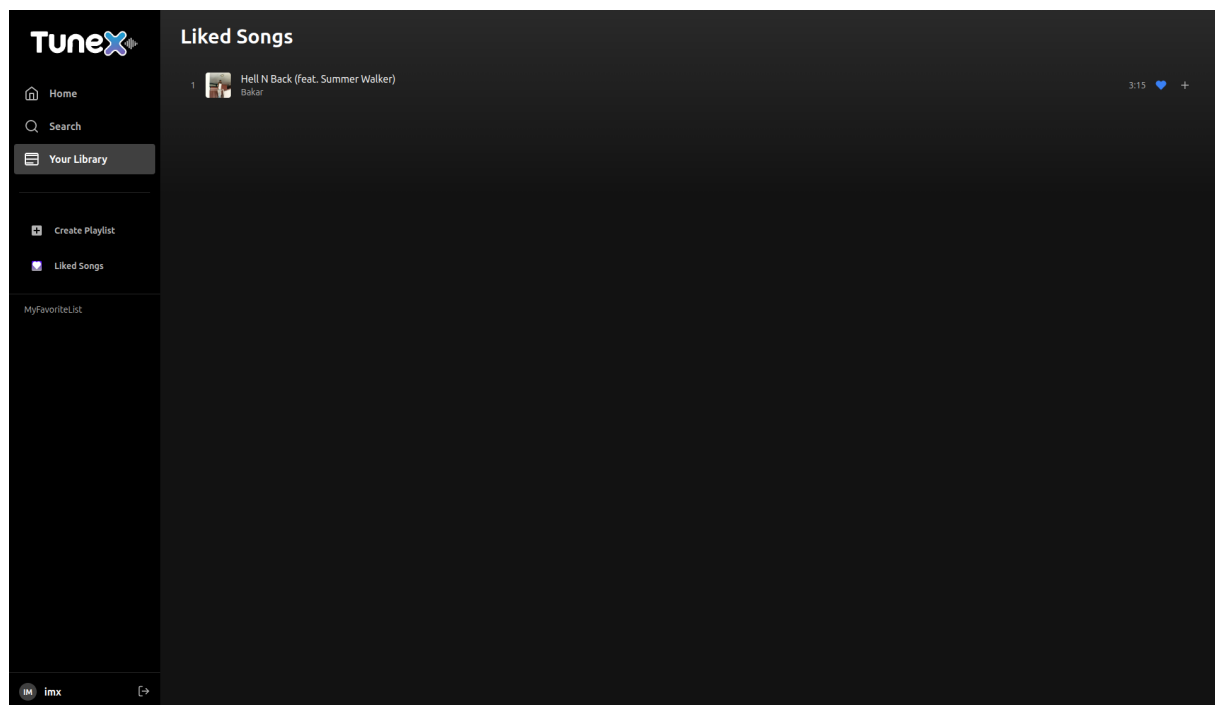


FIGURE 3.7 – Page des musiques aimées

- **Contenu d'une playlist** : affichage des musiques comprises dans une playlist sélectionnée.

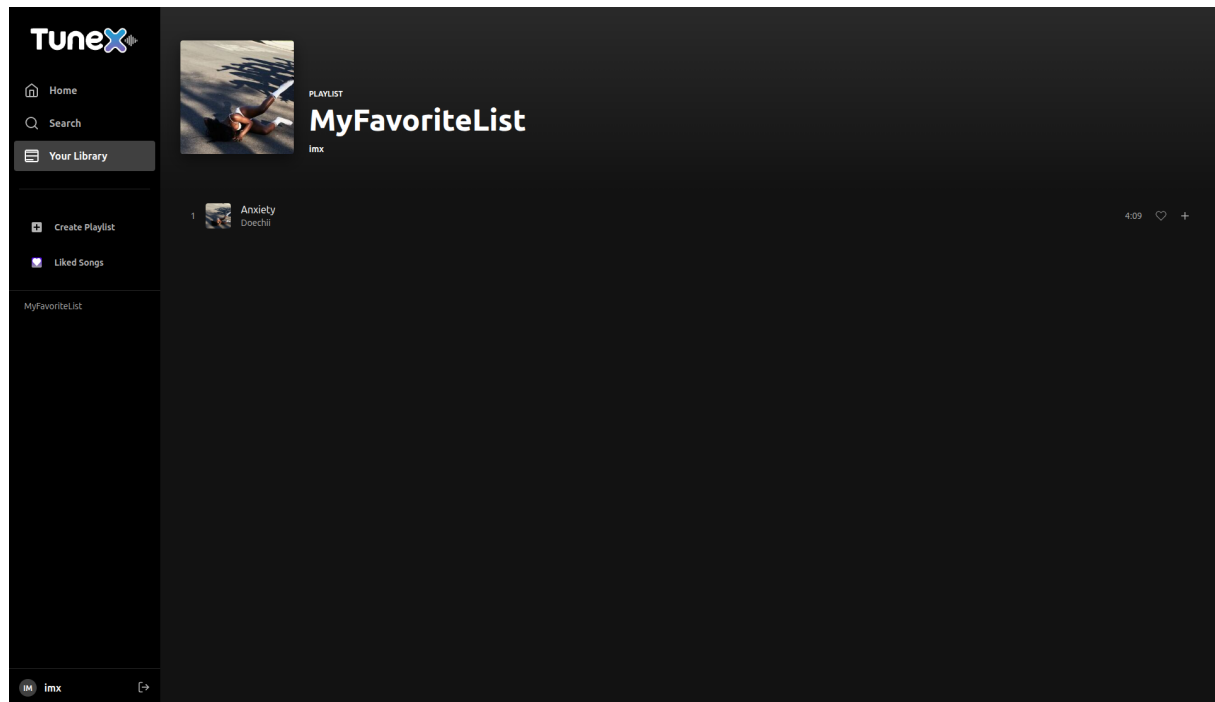


FIGURE 3.8 – Contenu d'une playlist

- **Lecteur audio** : barre inférieure affichant le morceau en cours de lecture et les contrôles (lecture, pause, suivant, précédent).

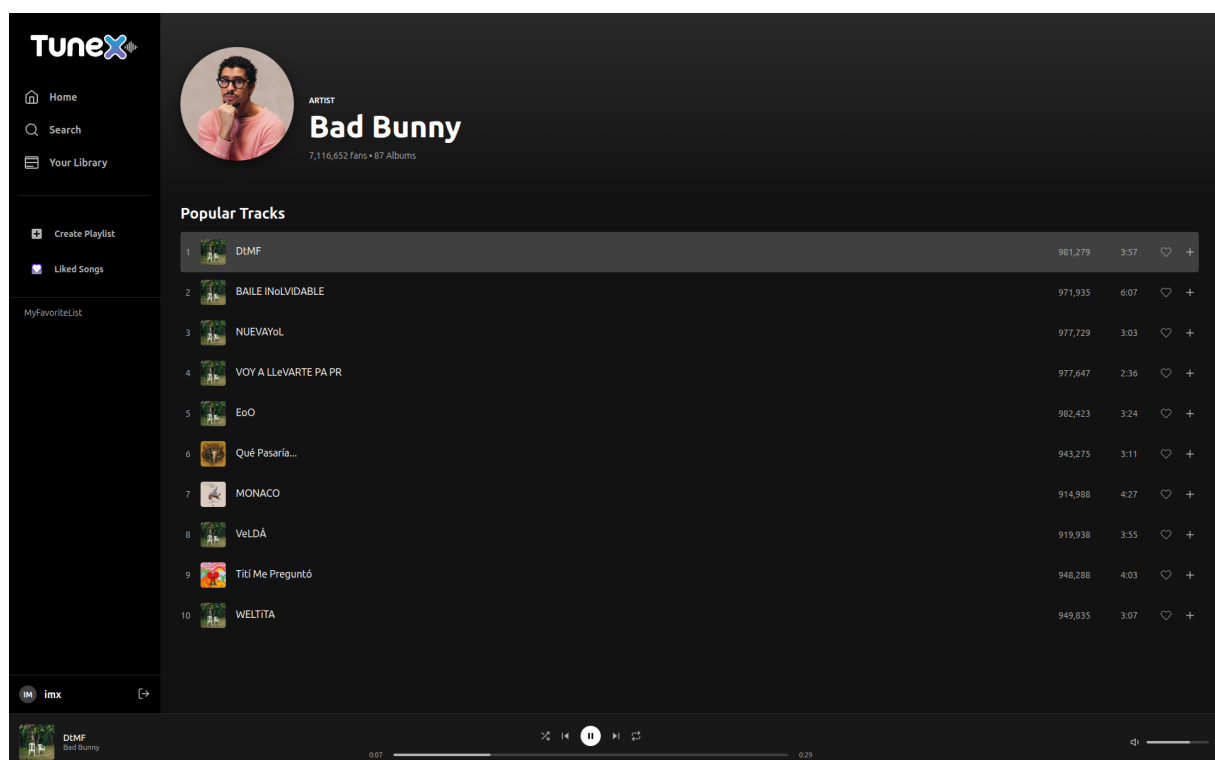


FIGURE 3.9 – Lecteur audio

3.3 Conclusion

Nous avons exploré dans ce chapitre les différentes interfaces et les fenêtres utilisées pour chaque utilisateur, ensuite, nous avons détaillé en particulier les fonctionnalités offertes pour chacune d'elles.

Conclusion Générale

Au terme de ce projet, TuneX se présente comme une plateforme de streaming musical web à la fois ergonomique, performante et évolutive. Son développement sous Django répond pleinement aux objectifs initiaux : offrir une expérience d'écoute fluide et simplifier la création de playlists.

Sur le plan de l'avancement, l'application est déjà fonctionnelle pour les cas d'usage majeurs :

- Authentification et gestion sécurisée des comptes.
- Lecture en continu des morceaux avec contrôle intégré (play, pause, skip).
- Création, modification de playlists.

Ce projet nous a également offert l'opportunité de :

- Approfondir notre maîtrise de Django, en structurant les vues, modèles et formulaires selon les meilleures pratiques,
- Mettre en œuvre la modélisation UML pour documenter l'architecture fonctionnelle, dynamique et logique du système,
- Déployer une application web responsive, avec PythonAnywhere,
- Appliquer les principes d'ergonomie et de design pour garantir une interface claire et accessible sur desktop et mobile.

Perspectives d'évolution

- **Mode hors-ligne** : permettre l'écoute de contenus téléchargés, afin d'assurer une continuité de service sans connexion Internet,
- **Application mobile native** : proposer une version iOS/Android pour étendre l'accessibilité et exploiter les notifications push,
- **Moteur de recommandations par IA** : enrichir l'algorithme actuel avec des techniques de machine learning (collaborative filtering, deep learning) pour affiner les suggestions musicales,
- **Fonctionnalités sociales** : ajout de flux d'activités, commentaires et interactions entre utilisateurs pour renforcer l'engagement,
- **Analyses et reporting** : tableau de bord d'analytics pour suivre l'usage, la popularité des morceaux et optimiser le catalogue.

RÉFÉRENCES

OUVRAGES :

- **GANTT** : *Organizing for Work*, de H. L. Gantt, publié en 1919 chez Harcourt, Brace and Howe.
- **UML** : *Unified Modeling Language (UML) Specification, Version 2.5.1*, par Object Management Group (OMG), publié en 2017.

WEB :

- **Django** : <https://docs.djangoproject.com/en/stable/> — Documentation officielle du framework Django.
- **MySQL** : <https://dev.mysql.com/doc/> — Documentation officielle de MySQL.
- **Deezer API** : <https://developers.deezer.com/api> — Documentation officielle de l'API Deezer.
- **GANTT** : <https://asq.org/quality-resources/gantt-chart> — Présentation et ressources sur les diagrammes de Gantt.
- **UML** : <https://www.omg.org/spec/UML/2.5.1/> — Spécification officielle du langage UML.