

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»

Завідувач кафедри

_____ Євгенія СУЛЕМА

«___» _____ 2025 р.

Дипломний проєкт

на здобуття ступеня бакалавра

**за освітньо-професійною програмою «Інженерія програмного
забезпечення мультимедійних та інформаційно-пошукових систем»**

спеціальності 121 Інженерія програмного забезпечення

**на тему: «Програмна система управління випуском музики для
незалежних лейблів»**

Виконав:

студент IV курсу, групи КП-12

Чердаклієв Кирило Євгенович _____

Керівник:

асистент кафедри ПЗКС, д-р філософії,

Юсин Яків Олексійович _____

Консультант з нормоконтролю:

доцент кафедри ПЗКС, к.т.н., доцент,

Онай Микола Володимирович _____

Рецензент:

доцент кафедри ПСТ ФІТ «КНУ

ім. Тараса Шевченка», д.т.н., с.н.с.,

Порєв Геннадій Володимирович _____

Засвідчую, що у цьому дипломному
проєкті немає запозичень з праць інших
авторів без відповідних посилань.

Студент _____

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення
мультимедійних та інформаційно-пошукових систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Євгенія СУЛЕМА

«___» _____ 2024 р.

ЗАВДАННЯ

на дипломний проєкт студенту

Чердаклієву Кирилу Євгеновичу

1. Тема проєкту «Програмна система управління випуском музики для незалежних лейблів», керівник проєкту Юсин Яків Олексійович, асистент кафедри ПЗКС, доктор філософії, затверджені наказом по університету №1808-С. від «29» травня 2025 р.
2. Термін подання студентом проєкту «13» червня 2025 р.
3. Вихідні дані до проєкту: див. Технічне завдання.
4. Зміст пояснювальної записки:
 - огляд та аналіз існуючих рішень;
 - обґрунтування вибору засобів реалізації;
 - розроблення застосунку
 - аналіз розробленого вебзастосунку.
5. Перелік обов'язкового графічного матеріалу:
 - схема бази даних (креслення);
 - функціональність вебзастосунку (креслення);
 - схема оброблення HTTP запитів (плакат);

— карта вебзастосунку (плакат);

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «31» жовтня 2024 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	18.11.2024	
2.	Розроблення та узгодження технічного завдання	28.11.2024	
3.	Розроблення структури вебзастосунку	13.12.2024	
4.	Підготовка матеріалів першого розділу дипломного проєкту	26.12.2024	
5.	Розроблення дизайну сторінок та графічних елементів	07.02.2025	
6.	Підготовка матеріалів другого розділу дипломного проєкту	17.02.2025	
7.	Програмна реалізація вебзастосунку	4.03.2025	
8.	Тестування вебзастосунку	13.03.2025	
9.	Підготовка матеріалів третього розділу дипломного проєкту	29.03.2025	
10.	Підготовка матеріалів четвертого розділу дипломного проєкту	12.04.2025	
11.	Підготовка графічної частини дипломного проєкту	21.04.2025	
12.	Оформлення документації дипломного проєкту	29.05.2025	

Студент

Кирило ЧЕРДАКЛІЄВ

Керівник проєкту

Яків ЮСИН

АНОТАЦІЯ

У дипломному проєкті розглянуто процес проєктування та розроблення вебзастосунку «ReleaseM Manager», призначеного для управління музичними релізами незалежних лейблів. Основною метою проєкту є централізоване керування творчим контентом, зокрема – створення, модерація та підготовка релізів до публікації, а також надання статистичних та аналітичних даних.

У процесі роботи було проаналізовано наявні програмні рішення, виявлено їхні недоліки й окреслено потреби незалежних виконавців і менеджерів. На підставі цього обґрунтовано вибір технологій: фреймворку Django для серверної частини, PostgreSQL як системи керування базами даних, а також JavaScript та AJAX для забезпечення інтерактивності інтерфейсу.

Розроблений MVP включає реалізацію ролей користувачів (артист, менеджер лейблу), можливість створення та перевірки релізів, завантаження метаданих, аудіо- та візуального контенту, генерацію звітів, планування промокампаній через чеклісти та бюджети, а також перегляд статистики прослуховувань. Система спрямована на оптимізацію внутрішніх процесів незалежних лейблів, покращення комунікації між артистом та менеджером і забезпечення прозорості підготовки релізу до публікації.

ABSTRACT

This diploma project explores the process of designing and developing the web application «ReleaseM Manager», intended for managing music releases by independent labels. The main goal of the project is to provide centralized control over creative content – including the creation, moderation, and preparation of releases for publication – as well as access to statistical and analytical data.

During the development process, existing software solutions were analyzed, their shortcomings identified, and the needs of independent artists and label managers outlined. Based on this analysis, the following technologies were selected: the Django framework for the backend, PostgreSQL as the database management system, and JavaScript with AJAX to ensure interface interactivity.

The developed MVP includes the implementation of user roles (artist, label manager), functionality for creating and reviewing releases, uploading metadata, audio and visual content, generating reports, planning promotional campaigns through checklists and budgets, and viewing playback statistics. The system is aimed at optimizing the internal processes of independent labels, improving communication between artists and managers, and ensuring transparency in release preparation.

ДП.045440-01-90 Програмна система управління випуском музики для незалежних лейблів. Відомість проєкту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проєкту		
ДП.045440-02-91	Програмна система	5	
	управління випуском		
	музики для незалежних		
	лейблів. Технічне		
	завдання		
ДП.045440-03-81	Програмна система	64	
	управління випуском		
	музики для незалежних		
	лейблів. Пояснювальна		
	записка		
ДП.045440-04-51	Програмна система	4	
	управління випуском		
	музики для незалежних		
	лейблів. Програма та		
	методика тестування		
ДП.045440-05-34	Програмна система	18	
	управління випуском		
	музики для незалежних		
	лейблів. Керівництво		
	користувача		
ДП.045440-06-99	Програмна система	1	
	управління випуском		
	музики для незалежних		
	лейблів. Схема бази		
	даних. ER-діаграма		

[illegible]

Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Євгенія СУЛЕМА

“ ____ ” _____ 2024 р.

ПРОГРАМНА СИСТЕМА УПРАВЛІННЯ ВИПУСКОМ МУЗИКИ
ДЛЯ НЕЗАЛЕЖНИХ ЛЕЙБЛІВ

Технічне завдання

ДП.045440-02-91

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Яків ЮСИН

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Кирило ЧЕРДАКЛІЄВ

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ	3
2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ.....	3
3. ПРИЗНАЧЕННЯ РОЗРОБКИ	3
4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ	3
5. ВИМОГИ ДО ПРОЄКТНОЇ ДОКУМЕНТАЦІЇ.....	4
6. ЕТАПИ ПРОЄКТУВАННЯ.....	4
7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ.....	5

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: програмна система управління випуском музики для незалежних лейблів.

Галузь застосування: культура та мистецтво.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проєктування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка «ReleaseM Manager» призначена для створення зручної програмної системи, що забезпечує управління випуском музики для незалежних лейблів. Система дає змогу централізовано організовувати процес створення, модерації, планування релізів та аналізу результатів діяльності, спрощуючи взаємодію між артистами та менеджерами.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Вебзастосунок повинен забезпечувати такі основні функції:

- 1) можливість реєстрації менеджера лейблу через публічну форму;
- 2) можливість авторизації для всіх ролей (менеджер, артист);
- 3) можливість створення артистів менеджером лейблу;
- 4) можливість створення релізів артистами із заповненням усіх метаданих;
- 5) можливість додавання одного або декількох треків до релізу;

- 6) можливість перевірки релізу менеджером: перегляд, прослуховування, затвердження або відхилення;
- 7) відображення статусу релізу та причин відхилення в особистому кабінеті артиста;
- 8) можливість додавання бюджету та формування списку витрат до релізу;
- 9) можливість створення чекліста завдань до кожного релізу;
- 10) перегляд статистики релізів у вигляді графіків і діаграм;
- 11) можливість додавання маркетингового звіту менеджером;
- 12) можливість прикріплення посилання на креативи артистом;
- 13) автоматичне сповіщення артиста про зміну статусу релізу;
- 14) можливість редагування відхиленого релізу та повторного подання на перевірку.

5. ВИМОГИ ДО ПРОЄКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проєкту повинна бути розроблена наступна документація:

- 1) пояснювальна записка;
- 2) програма та методика тестування;
- 3) керівництво користувача;
- 4) креслення:
 - «Функціональність вебзастосунку. UML-діаграма прецедентів»;
 - «Схема бази даних системи. ER-діаграма».

6. ЕТАПИ ПРОЄКТУВАННЯ

Вивчення літератури за тематикою роботи.....18.11.2024
Розроблення та узгодження технічного завдання.....28.11.2024

Розроблення структури вебзастосунку.....	13.12.2024
Розроблення дизайну сторінок та графічних елементів.....	07.02.2025
Програмна реалізація вебзастосунку.....	04.03.2025
Тестування вебзастосунку.....	13.03.2025
Підготовка матеріалів текстової частини проєкту.....	12.04.2025
Підготовка матеріалів графічної частини проєкту.....	21.04.2025
Оформлення технічної документації проєкту.....	29.05.2025

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до «Програми та методики тестування».

Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Євгенія СУЛЕМА

“___” _____ 2025 р.

ПРОГРАМНА СИСТЕМА УПРАВЛІННЯ ВИПУСКОМ МУЗИКИ
ДЛЯ НЕЗАЛЕЖНИХ ЛЕЙБЛІВ

Пояснювальна записка

ДП.045440-03-81

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Яків ЮСИН

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Кирило ЧЕРДАКЛІЄВ

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ	3
ВСТУП	4
1. ОГЛЯД ТА АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ	5
1.1. LabelGrid	5
1.2. Reprtoir	11
1.3. Висновки	18
2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ	20
2.1. Вибір технологій для розроблення серверної частини	20
2.2. Обґрунтування вибору системи керування базами даних	23
2.3. Вибір технологій для розроблення клієнтської частини	25
2.4. Висновки	28
3. РОЗРОБЛЕННЯ ЗАСТОСУНКУ	30
3.1. Загальний опис програми	30
3.2. Функціональні вимоги до вебзастосунку	32
3.3. Особливості реалізації серверної частини	36
3.4. Архітектура бази даних	42
3.5. Особливості реалізації клієнтської частини застосунку	49
3.6. Висновки	50
4. АНАЛІЗ РОЗРОБЛЕНОГО ВЕБЗАСТОСУНКУ	52
4.1. Аналіз реалізованого вебзастосунку	52
4.2. Тестування реалізованого вебзастосунку	53
4.3. Рекомендації для майбутнього вдосконалення застосунку	57
4.4. Висновки	59
ВИСНОВКИ	60
СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ	62
ДОДАТКИ	64

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

Лейбл – це організація або компанія, яка займається випуском, просуванням і розповсюдженням музики артистів. Лейбл може бути великим (*мейджор*) або незалежним (*інді*). Він може забезпечувати фінансування запису, маркетинг, дистрибуцію, а також юридичну та PR-підтримку.

Реліз – це офіційний вихід музичного продукту (синглу, EP, альбому) у публічний доступ через цифрові платформи (DSP) або інші канали. Реліз включає не лише саму музику, а й супровідні матеріали: обкладинку, метадані, опис, а іноді – відео чи промокампанію. Він може бути як цифровим, так і фізичним (вініл, CD).

DSP (англ. Digital Service Provider) – це цифрова платформа, яка надає доступ до музики через стрімінг або завантаження. Серед найвідоміших DSP – Spotify, Apple Music, YouTube Music, Deezer, Amazon Music тощо. Вони дозволяють слухачам відтворювати музику онлайн, а артистам – розповсюджувати свої треки через агрегаторів або дистриб'юторів.

Дистриб'ютор (музичний дистриб'ютор) – це посередник між артистом або лейблом і цифровими сервісами (DSP), який забезпечує доставку музики на стрімінгові платформи (Spotify, Apple Music, YouTube Music тощо). Дистриб'ютор також може надавати додаткові послуги – звітність про прослуховування, монетизацію, збір роялті, контент-ID, іноді – маркетингову підтримку.

SaaS-платформа (англ. Software as a Service) – це модель надання програмного забезпечення як онлайн-сервісу через інтернет. Користувачі отримують доступ до функціональності програми через браузер без потреби встановлювати її на свій пристрій. Усі оновлення, зберігання даних і обробка відбуваються на стороні постачальника сервісу. У музичній індустрії SaaS-платформи можуть надавати послуги з управління релізами, аналітики, створення ліцензій, моніторингу роялті тощо.

ВСТУП

У контексті стрімкого розвитку музичної індустрії дедалі більшої популярності набуває незалежне музичне продюсування, в межах якого артисти та незалежні лейбли прагнуть реалізовувати власні творчі продукти без участі великих компаній. Такий підхід дає змогу зберігати контроль над авторським баченням, проте водночас створює низку організаційних та технічних викликів. Одним із ключових аспектів успішної діяльності незалежних лейблів є ефективне управління процесом випуску музики – від обробки метаданих до планування промокампаній та аналізу результатів після релізу.

Попри наявність різноманітних платформ для дистрибуції контенту, часто бракує комплексного інструменту, що дозволив би централізовано керувати релізами, координувати дії артистів та менеджерів, зберігати усі матеріали у впорядкованому вигляді, а також автоматизувати рутинні процеси, такі як формування звітів чи генерування контрактів. Це створює попит на створення спеціалізованого вебзастосунку, що відповідає потребам малих команд, які діють у межах обмежених ресурсів.

Метою цього дипломного проєкту є розробка інформаційної системи «ReleaseM Manager», що дозволить незалежним лейблам та артистам ефективно управляти повним життєвим циклом музичного релізу. Застосунок передбачає функціональність для створення релізів із прикріпленням медіаматеріалів та метаданих, планування і контроль виконання підготовчих дій, взаємодію між артистами, менеджерами та дистриб'юторами, а також надання базової аналітики. Система має забезпечити простоту у використанні, адаптивність до потреб користувачів і можливість подальшого масштабування.

Запропонований підхід сприятиме професіоналізації незалежного сектору музичної індустрії, оптимізації внутрішніх процесів лейблів та підвищенню якості управління творчими проєктами.

1. ОГЛЯД ТА АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1. LabelGrid

LabelGrid – це професійна платформа для цифрової музичної дистрибуції, орієнтована на незалежні лейбли, продюсерів і виконавців і має доволі велику кількість функцій. Вона дозволяє централізовано керувати релізами, артистами, метаданими, договорами, статистикою, а також полегшує підготовку контенту до розповсюдження через цифрові музичні сервіси (DSP).

Інтерфейс LabelGrid побудований у вигляді панелі адміністратора з модулями для кожного з напрямків (релізи, артисти, звіти, дистрибуція). Має зручне розділення за ролями: артист бачить лише свою статистику, менеджер – загальну.

Вебсервіс LabelGrid не тільки дозволяє повністю керувати життєвим циклом випуску релізу, а ще й дозволяє розповсюджувати музичні релізи на величезну кількість цифрових музичних сервісів таких як Spotify, Apple Music, Deezer, Bandcamp, SoundCloud, TikTok та інші, тобто виступає ще й в якості дистриб'ютора. Також є можливість підключити іншого дистриб'ютора, з яким співпрацює та має контракт музичний лейбл [1].

Функціональні можливості даного продукту досить широкі [2]:

- завантаження релізів та керування метаданими;
- автоматичне формування ZIP-архівів із релізами для дистрибуції;
- дистрибуція контенту;
- доступ до статистики прослуховувань;
- інструменти для контролю якості контенту;
- інструменти для просування;
- фінансовий облік та звітність;
- прийом, прослуховування та оцінка демо-матеріалів від потенційних артистів або партнерів;

- можливість створення власного брендованого рішення на базі LabelGrid або інтеграція функціональних можливостей платформи у власні системи через API.

Однією з основних функціональних можливостей LabelGrid є створення та управління музичними релізами. Ця функція є фундаментом для всіх інших процесів, пов'язаних із дистрибуцією, просуванням та аналітикою. Для створення нового релізу користувач обирає тип релізу (сингл, EP або альбом), після чого відкривається форма з розширеними полями (рис. 1.1). Необхідно заповнити такі основні поля:

- назва релізу;
- назва артиста;
- тип релізу (Single, EP, Album);
- обкладинка (у відповідному форматі та роздільній здатності);
- жанр та піджанр;
- мова композицій;
- дата релізу;
- ISRC та UPC-коди;
- авторські та видавничі дані.

Рис. 1.1. Загальна форма для додавання релізу

Після заповнення всіх основних форм, додається сам трек, якщо це сингл, або декілька треків, якщо це альбом або міні-альбом. Потрібно завантажити аудіо-файл у форматі WAV 44,1 кГц, 16 біт стерео [3]. Також додати текст, згідно вимог музичних майданчиків. До кожного релізу можна прикріпити: промоматеріали, прес-пакети та/або документи для підписання (договір, маркетингова стратегія).

Ці додаткові матеріали автоматично включаються до ZIP-архіву метаданих, що згодом передається дистриб'ютору для модерації та розповсюдження.

Після заповнення усіх полів і завантаження файлів LabelGrid автоматично проводить валідацію метаданих. У випадку помилок (наприклад, відсутність ISRC або помилковий формат аудіо) користувачу виводяться відповідні повідомлення. Після успішної перевірки реліз можна передавати на дистрибуцію (рис. 1.2).

Рис. 1.2. Приклад підтвердження відправлення на дистрибуцію

Ще однією важливою функціональною можливістю платформи LabelGrid є доступ до статистики прослуховувань, завантажень і доходів по кожному релізу. Це надає як артисту, так і менеджеру або дистриб'ютору інструменти для відстеження ефективності релізу в цифровому середовищі.

Аналітика надається у зручному форматі з поділом за платформами, країнами, типами використання та часовими періодами.

Платформа дозволяє не лише керувати процесом дистрибуції, а й надає ефективні засоби для просування контенту на різних цифрових і соціальних майданчиках. Особливо актуальними ці функції є для незалежних лейблів, які прагнуть максимізувати охоплення своєї аудиторії без залучення зовнішніх маркетингових агентств. Розглянемо основні.

Плагін LabelGrid Tools для WordPress – дозволяє інтегрувати каталог релізів із сайтом лейблу чи артиста, автоматично створюючи сторінки релізів із повною інформацією: обкладинка, опис, посилання на DSP, відео, соцмережі тощо. Розглянемо основні можливості [4] даного плагіну:

- Інтеграція з маркетинговими пікселями Facebook Ads, Google Ads, TikTok Ads – дозволяє відстежувати конверсії з рекламних кампаній та вимірювати ефективність кожного джерела трафіку.
- Функція Spotify Pre-Save – користувачі можуть зберегти трек ще до дати релізу, що дозволяє артисту отримати більше прослуховувань у перші години після публікації.
- Гейтовані дії (Gate features) – наприклад, завантажити трек можуть тільки ті користувачі, які підписалися на сторінку артиста або залишили свій email. Це допомагає лейблам збирати контактні бази для email-маркетингу.

Ще однією функціональною можливістю вебплатформи LabelGrid, яка є корисною для незалежних лейблів, є прийом, прослуховування та оцінка демо-матеріалів (рис. 1.3), що надсилаються потенційними артистами, продюсерами або партнерами. Це дозволяє централізовано керувати потоком нових треків, які надходять на розгляд, і формувати власну базу для підписання контрактів або спільної роботи.

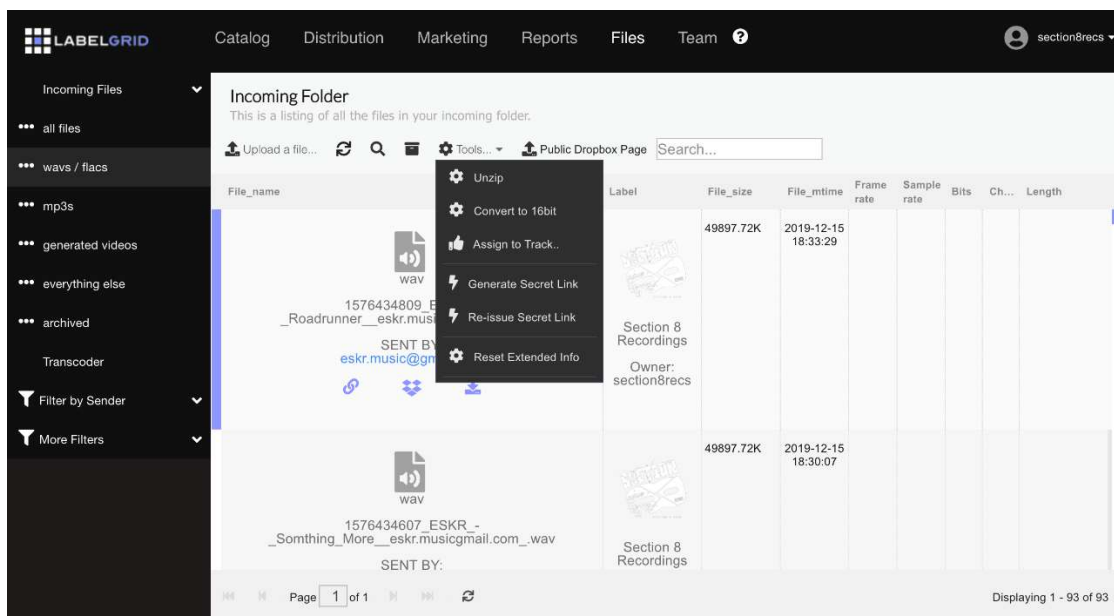


Рис. 1.3. Вигляд списку приймання демо-матеріалу

Демо надходять у вигляді посилань або прикріплених файлів (в залежності від налаштувань). Також доступна система внутрішніх коментарів та нотаток до кожного демо. За необхідності, демо можна прикріпити до профілю потенційного артиста або перенести в етап укладання договору.

У системі є можливість [5] зберігати всі подані заявки в єдиному інтерфейсі для перегляду, сортування та комунікації, проставляти статуси (прийнято, переглянуто, відхилено, очікує відповіді), призначити менеджера або A&R, який відповідатиме за перегляд нових заявок.

Окрім розглянутих вище основних функціональностей, вебсервіс LabelGrid також пропонує низку додаткових можливостей, які суттєво розширюють інструментарій як для артистів, так і для лейбл-менеджерів. До таких функцій належать: ролі користувачів у системі, модуль модерації релізів, контроль якості метаданих, інтеграція з зовнішніми API, звіти про ротацію на радіо або публікації в ЗМІ тощо.

Варто зазначити, що LabelGrid не є безплатним вебсервісом. Як і більшість професійних платформ для керування музичними релізами та дистрибуції, він працює за моделлю передплати (рис. 1.4), а також надає

окремі платні модулі (рис. 1.5), які можуть бути підключені за потреби. Сервіс також надає пробний 7-денний період користування.

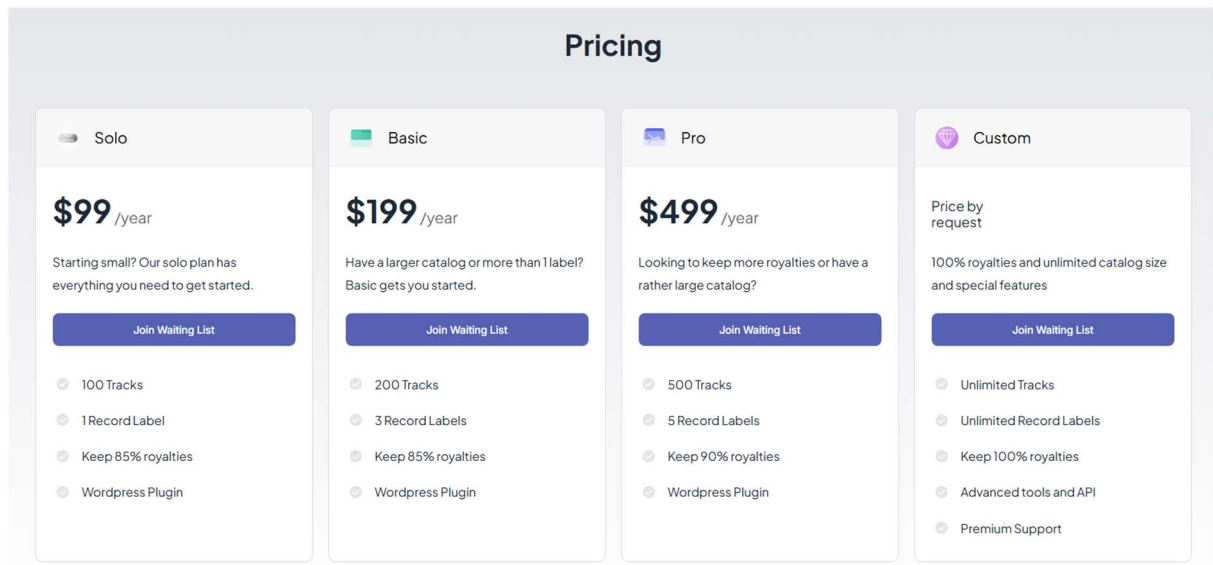


Рис. 1.4. Тарифні плани на підписки

До окремих платних модулів належать:

- LabelGrid Gate – для роботи з маркетингом, розповсюдженням SmartLinks і гейтованим доступом.
- White-Label – для створення кастомізованої дистрибуційної платформи з власним брендом, URL та інтерфейсом.
- Demo Submission Tools – для розміщення приймання демо на сайті (входить до LabelTools).

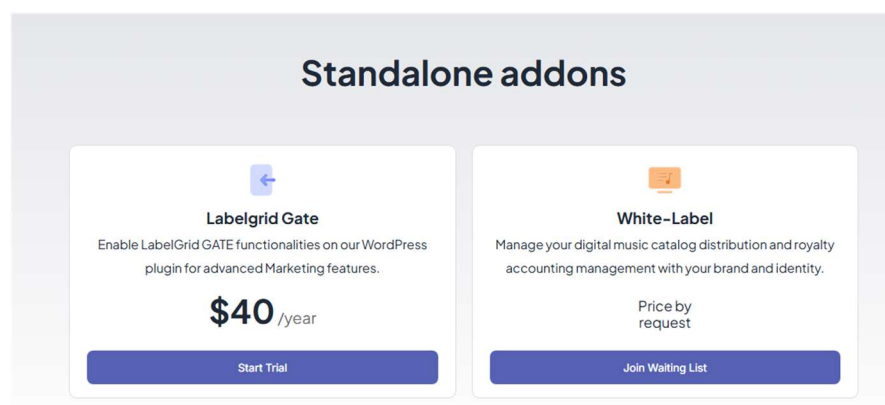


Рис. 1.5. Тарифні плани на окремі платні модулі

Платформа LabelGrid має низку переваг, які роблять її потужним інструментом для незалежних музичних лейблів та виконавців. Насамперед варто відзначити широкий функціональний спектр. Сервіс дозволяє повністю охопити процес підготовки та розповсюдження музичного релізу: від завантаження треків і метаданих до створення маркетингових кампаній, генерації договорів і перегляду статистики прослуховувань.

Важливою перевагою є підтримка інтеграції з WordPress за допомогою плагіну LabelGrid Tools, що дозволяє автоматично створювати сторінки релізів, синхронізувати контент і реалізовувати маркетингові функції. Серед інших сильних сторін – прийом та управління демо-записами, гнучка система ролей користувачів, White-Label налаштування для створення брендovаних платформ, а також потужна система контролю якості метаданих, яка допомагає уникнути помилок на етапі дистрибуції.

Попри численні переваги, платформа має й певні недоліки. По-перше, високий поріг входу для новачків. Інтерфейс платформи орієнтований на досвідчених користувачів – менеджерів лейблів, A&R-фахівців, продюсерів. Для артиста без технічного або музичного бекграунду перші кроки в LabelGrid можуть здатися складними. По-друге, обмежена звітність по ротаціях на радіо та ЗМІ. Хоча в LabelGrid можна вручну ввести маркетинговий звіт про ротації або згадки у медіа, автоматизованої інтеграції з PR-агентствами, радіостанціями або агрегаторами новин немає. Усе це заповнюється вручну. Нарешті, хоча LabelGrid підтримує дистрибуцію на більшість ключових цифрових музичних сервісів, деякі платформи потребують окремого посередника або API-інтеграції, що не завжди зручно.

1.2. Reprtoir

Reprtoir – це SaaS-платформа, розроблена для незалежних музичних компаній, зокрема лейблів, видавництв та дистриб'юторів. Її головна мета – надати централізоване середовище для управління музичним каталогом,

контрактами, роялті, статистикою, маркетингом і комунікацією з артистами. На відміну від платформ, які лише забезпечують дистрибуцію контенту, Reprtoir позиціонує себе як повноцінна CRM-система для музичного бізнесу, де поєднані бізнес-аналітика, документообіг, каталогізація та управління правами.

Reprtoir має дуже широкі функціональні можливості [6]:

- Управління музичним каталогом.
- Платформа для укладання та зберігання контрактів; створення шаблонів договорів і можливість електронного підписання без потреби сторонніх сервісів. Контракти зберігаються в особистій бібліотеці та можуть бути автоматично заповнені з даних каталогу.
- Розрахунок та облік роялті.
- Вбудовані аналітичні інструменти: статистика DSP, продажі, тренди.
- Music Share. Можливість приватно ділитись музичними творами, створювати приватні плейлисти для поширення.
- Audio AI. Вбудоване автоматичне додавання тегів (настрій, жанр) до релізів, пошук подібності та опис релізів за допомогою ШІ.
- Міграція музичного каталогу з інших систем. API-інтерфейс для масового імпорту даних, зокрема релізів, артистів, фінансової інформації та статистики з інших систем (наприклад, Fuga, Symphonic, або власна БД).
- Редагування великої кількості об'єктів (треки, релізи, договори) одночасно.
- Інструмент для приймання демозаписів
- Система інтелектуального фільтрування, яка автоматично створює динамічні категорії в каталозі, засновані на метаданих, тегах і фільтрах.
- Робочі області. Модуль для командної роботи над релізами та проєктами. Кожен користувач має персоналізовану панель, завдання

та файли. Це дозволяє створювати відокремлені робочі простори для кожного артиста, релізу або кампанії.

Однією з основних функцій платформи Reprtoir є управління музичним каталогом [7]. Саме ця складова лежить в основі всієї архітектури сервісу й забезпечує структуру для подальшої роботи з метаданими, контрактами, фінансовими розрахунками, аналітикою та комунікацією з артистами. Кожен трек або реліз у системі представлений як окремий об'єкт із гнучкою структурою даних, що дає змогу працювати з великими обсягами контенту професійно й системно. Приклад використання музичного каталогу показаний на рис. 1.6.

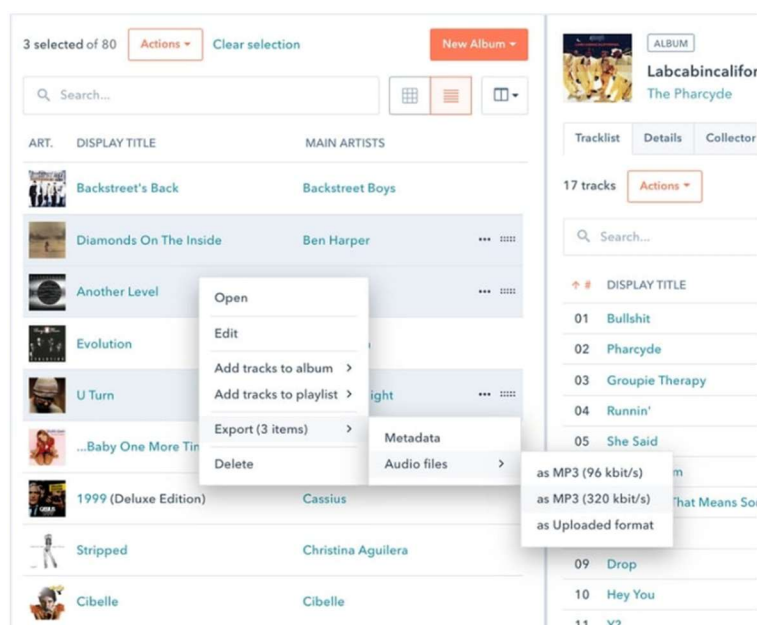


Рис. 1.6. Приклад використання музичного каталогу

Додавання нового треку або релізу відбувається через спеціалізовану форму, яка включає всі необхідні поля для подальшої дистрибуції або обліку: назва, виконавець, ISRC-код, авторські дані, тривалість, формат, а також підтримка мультимовності. Усі метадані можна налаштувати відповідно до внутрішніх вимог лейблу, а також доповнити власними полями, які не передбачені стандартною схемою. Це особливо важливо для

лейблів, які працюють з різними правовими юрисдикціями або мають власні внутрішні політики опису контенту.

Суттєвою перевагою Reprtoir є підтримка інструментів масового редагування. Якщо потрібно оновити інформацію одразу для кількох десятків релізів – наприклад, змінити видавця або статус прав – це можна зробити через інтерфейс без потреби повторно відкривати кожен об’єкт вручну. Також система дозволяє імпортувати великі обсяги каталогу з інших платформ (наприклад, Fuga, Label Engine) за допомогою API або інструментів міграції.

Однією з найбільш складних і водночас критично важливих складових у роботі музичного лейблу є облік роялті. Reprtoir пропонує повноцінний вбудований модуль для розрахунку [8], ведення статистики прослуховувань (рис. 1.7) та виплати роялті, який розроблений спеціально з урахуванням потреб музичної індустрії та змін у цифровій дистрибуції.



Рис. 1.7. Приклад статистики прослуховувань

У традиційній моделі роялті-менеджменту лейбли зазвичай використовують стороннє програмне забезпечення або навіть електронні таблиці, що є неефективним, трудомістким і схильним до помилок. Reprtoir вирішує цю проблему за допомогою інтерфейсу, який дозволяє інтегрувати джерела доходів, автоматично розподіляти прибуток відповідно до

договорів (рис. 1.8), а також формувати звіти для артистів, авторів і видавців.

RIGHTHOLDER	ROLE	DIG %
× Universal Music Group	Producer	80
× India Arie Simpson	Artist	12
× Mark Batson Music	Musician	3
× New Star Management	Co-producer	5
		100

Add a rightholder

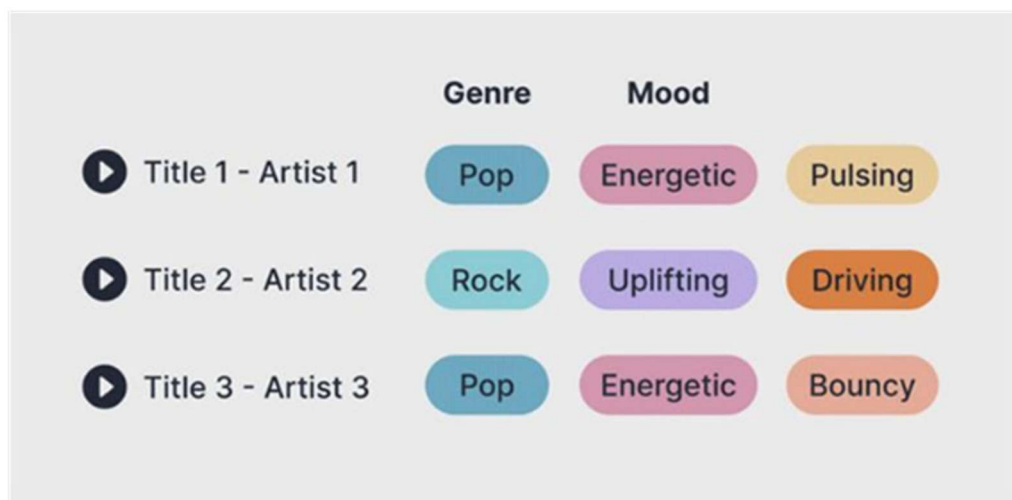
Rightholder *	Role *
Rightholder	Role

Рис. 1.8. Приклад відсоткового розподілу прибутку

Суттєвою перевагою є те, що всі розрахунки фіксуються у вигляді прозорих звітів, які можуть бути експортовані в PDF або Excel. Артисти отримують доступ до власних панелей, де можуть бачити історію роялті, періоди нарахувань, джерела доходу і, за потреби, подавати запити на виплату. Це зменшує адміністративне навантаження на лейбл і водночас підвищує рівень прозорості у відносинах з підписантами. Reprtoir також дозволяє враховувати аванси, мінімальні виплати, бонуси, пріоритетні права, а також різні територіальні обмеження або платформи.

Також цікаво, те що, Reprtoir активно впроваджує сучасні технології у свою інфраструктуру, зокрема – модулі, що базуються на штучному інтелекті. Суть роботи модуля [9] полягає у тому, що після завантаження треку в систему, алгоритм автоматично аналізує його звуковий профіль. На

основі акустичних та ритмічних параметрів. III визначає жанр, піджанр, настрій композиції, динаміку та енергетичний рівень (рис. 1.9). Ці дані одразу додаються до полів метаданих, які можуть бути використані для формування пошукових фільтрів Smart Folders або промо-опису треку.



The image shows a user interface for classifying music tracks. It features a table with three rows of tracks, each with a play button icon, a title, and two columns of tags: 'Genre' and 'Mood'. The 'Genre' column has three tags: 'Pop', 'Rock', and 'Pop'. The 'Mood' column has three tags: 'Energetic', 'Uplifting', and 'Energetic'. Each tag is represented by a colored pill-shaped button.

	Genre	Mood
▶ Title 1 - Artist 1	Pop	Energetic
▶ Title 2 - Artist 2	Rock	Uplifting
▶ Title 3 - Artist 3	Pop	Energetic

Рис. 1.9. Приклад класифікації за критеріями

Крім жанрової класифікації, Reprtoir також реалізував можливість пошуку схожих треків за допомогою вбудованої системи подібності. Це допомагає менеджерам лейблу автоматично формувати рекомендаційні добірки, створювати тематичні плейлисти або здійснювати аналіз каталогів потенційних партнерів.

Платформа Reprtoir працює за моделлю підписки з можливістю гнучкого вибору тарифного плану залежно від потреб лейблу, обсягу каталогу та кількості користувачів (рис. 1.10). Усього передбачено п'ять основних пакетів: Free, Starter, Professional, Business, Enterprise, кожен з яких орієнтований на окремий сегмент користувачів – від індивідуальних артистів до великих компаній з багатотисячними каталогами та потребами у глибокому індивідуальному налаштуванні. Користувачі можуть змінювати тариф у будь-який момент, що забезпечує додаткову гнучкість у масштабуванні діяльності.

Free	Starter	Professional	Business	Enterprise
From \$0 per month or \$0 per year	From \$49 per month or \$539 per year	From \$129 per month or \$1,419 per year	From \$299 per month or \$3,289 per year	From \$999 per month or \$10,989 per year
Built for Creators who can leverage Reprtoire's free CMS and CRM tools, starting with Playlists and Inbox.	Designed specifically for indies who require CMS, CRM, and metadata enrichment capabilities.	Crafted to meet the needs of small businesses that require advanced royalty management features.	Designed to accommodate medium-sized organizations that require increased capabilities.	Built for enterprises managing extensive catalogs and generating substantial royalty income.
Try for free	Try for free	Try for free	Try for free	Try for free
What's included	What's included	What's included	What's included	What's included
1 user	1 user	2 users	3 users	5 users
1,000 assets	5,000 assets	10,000 assets	25,000 assets	100,000 assets
50 GB of storage	100 GB of storage	200 GB of storage	500 GB of storage	2 TB of storage
		\$20,000 of royalty statement processing per month — or \$240,000 per year	\$50,000 of royalty statement processing per month — or \$600,000 per year	\$200,000 of royalty statement processing per month — or \$2,400,000 per year

Рис. 1.10. Тарифний план Reprtoire

Окрім описаних основних функціональних можливостей, Reprtoire, має ще багато додаткових, які були перелічені у списку вище. Підведемо висновки у вигляді переваг та недоліків системи:

Переваги:

- Професійний рівень організації музичного каталогу, договорів, роялті та аналітики в одному інтерфейсі.
- Автоматизоване формування звітів про доходи, облік роялті, а також підтримка контрактів з електронним підписом.
- Інтеграція з AI-модулем для класифікації треків, тегування і пошуку подібного контенту.
- Гнучкість у налаштуванні: API, підтримка кастомних полів, динамічні фільтри (Smart Folders), масове редагування.
- Наявність безкоштовного тарифу для ознайомлення та старту роботи з платформою.

Недоліки:

- Висока вартість підписки на професійні та корпоративні плани.

- Немає прямої інтеграції з дистрибуційними платформами (тільки керування релізами, без автоматичного завантаження).
- Інтерфейс та логіка роботи системи можуть бути складними для користувачів без попереднього досвіду користування системою, високий поріг входу.

1.3. Висновки

У цьому розділі було проведено детальний аналіз двох сучасних вебплатформ, орієнтованих на музичний бізнес. LabelGrid більше фокусується на повному циклі дистрибуції, надаючи можливість автоматично формувати архіви релізів і передавати їх на музичні платформи, а також містить інструменти маркетингу та модерації релізів. Окремою перевагою LabelGrid є наявність плагіну для WordPress, що дозволяє швидко адаптувати інтерфейс до власного сайту лейблу. Серед недоліків слід відзначити закриту структуру, обмежену кастомізацію, а також відсутність повного контролю над дистрибуційним потоком у деяких тарифах.

Reprtoir, у свою чергу, позиціонується як CRM-платформа для незалежного музичного бізнесу, де на першому плані стоїть не дистрибуція, а централізоване управління каталогом, контрактами, роялті, статистикою і комунікацією з артистами. Вона більше орієнтована на внутрішню організацію процесів та інтеграцію бізнес-логіки у роботу лейблу. Водночас, система має високий поріг входу та суттєво вищу вартість підписки на професійні плани, що може бути обмеженням для невеликих лейблів.

Проведений аналіз дозволив виявити ключові функціональні особливості, переваги та обмеження двох платформ, що активно використовуються в індустрії. Це, у свою чергу, сформувало основу для визначення унікальних конкурентних переваг і функцій майбутнього вебзастосунку, що розробляється в межах дипломного проєкту. Подальші

етапи дослідження будуть присвячені проектуванню власної системи, яка враховуватиме як сильні сторони наявних рішень, так і їхні слабкі місця.

2. ОБГРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1. Вибір технологій для розроблення серверної частини

2.1.1. Python

Python – це мова високого рівня із простою та зрозумілою синтаксичною структурою, яка активно використовується у веброзробці, штучному інтелекті, обробці даних, автоматизації та інших галузях. Python підтримує структурне, функціональне, процедурне та об'єктно-орієнтоване програмування, а також має потужну стандартну бібліотеку [10]. За даними рейтингу ТЮВЕ, Python вже кілька років поспіль займає перше місце серед найпопулярніших мов програмування у світі [11]. Основними причинами такого домінування є простота, читабельність, активна спільнота та величезна кількість готових бібліотек.

Python має такі переваги при розробленні серверної частини програмного забезпечення:

- Повторне використання коду – завдяки модульності та підтримці класів, Python дозволяє повторно використовувати код у різних частинах проєкту, що пришвидшує розробку.
- Покращена читабельність та простота коду – синтаксис Python вважається одним із найзрозуміліших серед мов програмування, що знижує поріг входу та спрощує супровід.
- Швидка розробка – завдяки великій кількості готових модулів та бібліотек (наприклад, requests, os, json, sqlite3, datetime), Python дає змогу розробити функціональний прототип у найкоротші строки.
- Кросплатформність – Python-програми можуть виконуватися на різних ОС без змін у коді (Windows, Linux, macOS).
- Активна спільнота та документація – наявність великої кількості матеріалів, форумів і репозиторіїв значно полегшує вирішення технічних проблем.

- Інтеграція з іншими технологіями – Python чудово взаємодіє з базами даних (PostgreSQL, MySQL, MongoDB), API, фреймворками (Django, Flask), а також з інструментами візуалізації та аналітики.

Python є динамічно типізованою мовою, що означає автоматичне визначення типу змінної під час виконання. Хоча це може бути джерелом потенційних помилок, воно значно підвищує гнучкість коду та зменшує його об'єм. Завдяки цьому Python ідеально підходить для прототипування та швидкої реалізації складних логік.

Таким чином, мова програмування Python може забезпечити високу продуктивність розроблення, гнучкість і простоту реалізації бізнес-логіки серверної частини застосунку. Її універсальність і широка екосистема стане ключовими факторами ефективної побудови функціональності ReleaseM Manager.

2.1.2. Фреймворк Django

У якості вебфреймворку було обрано Django – потужний інструмент з відкритим вихідним кодом, який реалізує архітектуру MVC (Model-View-Controller) у варіації MTV (Model-Template-View). Django забезпечує модульну структуру розроблення та містить у собі широкий набір вбудованих функцій: ORM (об'єктно-реляційне відображення), маршрутизацію URL, захист від CSRF-атак, систему авторизації та автентифікації, формування HTML-шаблонів тощо [12]. Django розроблений з урахуванням принципу *DRY (Don't Repeat Yourself)* та фокусується на швидкості розроблення, чіткості коду та вбудованій безпеці. Він є одним з найпопулярніших Python-фреймворків у світі та використовується в таких проєктах, як Instagram, Spotify, Mozilla, Pinterest та Disqus [13]

Серед головних переваг Django для реалізації серверної частини:

- Вбудована ORM-система, яка дозволяє працювати з базами даних, не використовуючи вручну SQL-запити. Це пришвидшує розробку,

зменшує кількість помилок і забезпечує міграції схеми бази даних.

- Адміністративна панель, яка генерується автоматично і дозволяє зручно керувати об'єктами системи.
- Хороша документація та активна спільнота, що значно спрощує процес навчання та розв'язання проблем під час розробки.
- URL-маршрутизація – гнучка система маршрутизації дозволяє визначати шляхи до ресурсів на основі шаблонів URL.
- Модульність і масштабованість – фреймворк структурований у вигляді застосунків, які легко підключати або переносити між проєктами. Django чудово підходить як для малих MVP-прототипів, так і для великих масштабованих проєктів.
- Безпека «з коробки» – Django забезпечує захист від основних вебзагроз: CSRF (міжсайтових підробок запитів), XSS (міжсайтового скриптингу), SQL-ін'єкцій, clickjacking та інших. Усі шаблони автоматично екрануються, а форми мають CSRF-токени.

У межах дипломного проєкту Django буде використано для реалізації всієї серверної логіки:

- Аутентифікація користувачів – у системі буде реалізовано підтримку трьох ролей (адміністратор, лейбл-менеджер, артист), кожна з яких матиме свій набір прав доступу та окремі панелі управління.
- Створення форм – за допомогою ModelForm буде реалізовано форми для створення та редагування релізів, треків, бюджетів, а також форми для додавання креативів і чеклістів.
- Обробка POST/GET-запитів – сервер оброблятиме запити від користувачів та повертатиме відповідь у форматі HTML або JSON, залежно від контексту (наприклад, під час додавання треків, оновлення статусів релізів, збереження бюджетів тощо).
- Динамічна взаємодія з інтерфейсом користувача – у поєднанні з JavaScript буде реалізовано обробку чеклістів, динамічне оновлення

статистичних графіків, а також завантаження релізів на модерацію без перезавантаження сторінки.

- Механізми перегляду і фільтрації даних – лейбл-менеджери матимуть змогу переглядати список своїх артистів, фільтрувати релізи за статусом, типом та датою, а також переглядати повну інформацію про кожен реліз.
- Модерація релізів – буде реалізовано повний цикл перевірки: після подання релізу артистом менеджер перевірятиме його метадані та треки, а потім схвалюватиме або відхилятиме реліз із вказанням причини. Лише після схвалення реліз отримає статус «approved».

Таким чином, фреймворк Django стане оптимальним вибором для реалізації серверної частини вебзастосунку. Його модульна архітектура, вбудовані інструменти для роботи з базами даних, високий рівень безпеки та зручна інтеграція з JavaScript забезпечать швидке, гнучке та безпечне впровадження функціональності відповідно до вимог проєкту.

2.2. Обґрунтування вибору системи керування базами даних

У межах проєкту ReleaseM Manager дані мають чітко визначену структуру що обумовлює використання реляційної моделі даних. З огляду на вимоги до цілісності даних, підтримки транзакцій, зручної інтеграції з Django та відкритої ліцензії, було обрано PostgreSQL як основну СКБД.

2.2.1. PostgreSQL

PostgreSQL – це потужна об’єктно-реляційна система керування базами даних з відкритим кодом, яка підтримує розширення, транзакційність, збережені процедури, складні запити, геопросторові дані, багаторівнені індекси та високий рівень відповідності стандарту SQL. Вона була створена в Університеті Каліфорнії в Берклі і активно розвивається завдяки міжнародній спільноті розробників [14].

До основних переваг PostgreSQL належать:

- Дотримання стандартів – повна підтримка SQL:2016, а також додаткові можливості, такі як віконні функції, CTE, часткові індекси та вирази в індексах.
- Об'єктно-реляційна модель – підтримка складних типів даних, масивів, JSON, XML, а також створення власних типів та функцій.
- Масштабованість – PostgreSQL підтримує вертикальне масштабування та має інструменти для горизонтального масштабування за допомогою реплікації, шардингу або сторонніх рішень.
- Розширюваність – завдяки модульній архітектурі легко додаються сторонні розширення, зокрема PostGIS для геоданих, pg_trgm для пошуку подібності рядків, uuid-ossr для генерації унікальних ідентифікаторів тощо.
- Транзакції та цілісність – підтримка ACID-транзакцій, зовнішніх ключів, каскадних оновлень/видалень і обмежень (constraints), що критично важливо для збереження достовірності даних.
- Безпека – підтримка автентифікації через паролі, SSL, сертифікати; налаштування рівнів доступу для різних ролей; аудит та журналювання транзакцій.
- Висока продуктивність – завдяки індексам (B-tree, GIN, GiST, Hash), оптимізатору запитів, кешуванню та підтримці паралельного виконання складних запитів.

PostgreSQL активно використовується в таких компаніях, як Instagram, Spotify, Reddit, TripAdvisor та інших. Вона також повністю інтегрується з Django ORM, що значно спрощує обробку моделей даних та дозволяє генерувати запити до БД автоматично на основі Python-класів.

У межах розроблення застосунку ReleaseM Manager PostgreSQL дозволить реалізувати такі функціональні можливості:

- зберігання інформації про користувачів, їхні ролі та автентифікацію;

- побудову зв'язків між об'єктами, зокрема артистами, релізами, треками та статистикою;
- забезпечення транзакційної цілісності під час створення, редагування або модерації релізів;
- ефективну обробку запитів, що потребують фільтрації, агрегації та аналітики по треках.

Завдяки відкритій ліцензії PostgreSQL не вимагатиме придбання платної версії, що зробить її оптимальним рішенням для розроблення дипломного проєкту.

2.3. Вибір технологій для розроблення клієнтської частини

Клієнтська частина вебзастосунку є ключовим елементом у забезпеченні зручної, інтуїтивної та привабливої взаємодії користувача із системою. Основною метою є забезпечення динамічного інтерфейсу для артистів і менеджерів лейблу, який дозволяє ефективно додавати релізи, працювати з чеклістами, переглядати статистику та завантажувати контент. З огляду на це, було обрано набір технологій, що відповідають сучасним вимогам до UX/UI. У межах реалізації клієнтської частини було обрано наступні технології.

2.3.1. HTML та CSS

HTML є основою розмітки вебсторінок, яка визначає структуру елементів користувацького інтерфейсу. CSS відповідає за стилізацію цих елементів: кольори, розміри, позиціонування, анімації та адаптивність інтерфейсу під різні пристрої.

Основні переваги використання HTML/CSS:

- Відокремлення структури та стилів – сприяє кращому розподілу обов'язків у розробці.
- Адаптивна верстка – забезпечена за допомогою медіазапитів та сіткових систем (flexbox, grid).

- Швидке завантаження сторінок – відсутність зайвих компонентів зменшує час рендерингу.
- Можливість гнучкого налаштування – повна свобода у стилізації без залежності від сторонніх бібліотек.
- Сумісність з усіма браузерами – HTML/CSS є стандартами, які підтримуються усіма сучасними веббrowsers.

2.3.2. JavaScript

JavaScript – це мова програмування високого рівня, що виконується на стороні клієнта і використовується для створення інтерактивних елементів на вебсторінках. Стандартизована мовою ECMAScript, вона підтримує мультипарадигмальний стиль програмування, включно з імперативним, функціональним та об’єктно-орієнтованим [15]. Завдяки інтеграції з HTML і CSS, JavaScript дозволяє реалізувати повноцінну взаємодію користувача з вебінтерфейсом – від обробки кліків і анімацій до асинхронного обміну даними з сервером.

У рамках розроблення клієнтської частини застосунку ReleaseM Manager мова JavaScript відіграватиме ключову роль у забезпеченні динамічної поведінки інтерфейсу та покращенні взаємодії з користувачем. Зокрема, буде реалізовано такі функціональні можливості:

- Двоетапне додавання релізів – за допомогою JavaScript буде організовано перемикання між етапами створення релізу: спочатку – введення основних метаданих (назва, тип, дата тощо), а потім – додавання треків. Це відбуватиметься без перезавантаження сторінки завдяки взаємодії з DOM-структурою.
- Динамічне додавання треків – у випадку створення альбому або EP користувач зможе додавати довільну кількість треків. JavaScript буде відповідати за клонування шаблонів форм і надання можливості видаляти їх без втрати інших даних.

- Асинхронна робота з сервером (AJAX) – через використання `fetch()` будуть реалізовані асинхронні запити до серверної частини. Наприклад, форма додавання завдань у чекліст надсилатиметься без оновлення сторінки, а відповідь оброблятиметься у форматі JSON для миттєвого оновлення інтерфейсу.
- Оновлення статистичних графіків – при перемиканні часових діапазонів (наприклад, останні 7 днів або за весь час) JavaScript динамічно оновлюватиме дані в `Chart.js` без потреби в перезавантаженні сторінки.
- Захист від втрати даних – буде реалізовано механізм попередження користувача про незбережені зміни у формах, якщо той намагатиметься випадково залишити сторінку. Це реалізується за допомогою обробника подій `beforeunload`.

Ключові переваги JavaScript:

- Інтерактивність – дозволяє реалізовувати складну поведінку інтерфейсу без звернення до сервера (навігація, перевірка форм, відкриття/закриття блоків).
- Широка підтримка бібліотек – наприклад, `Chart.js`, `Axios`, `jQuery`.
- Асинхронність – за допомогою `fetch()` або `XMLHttpRequest`, можна обробляти запити до сервера у фоновому режимі.
- Підтримка браузерами – JavaScript підтримується всіма сучасними браузерами без необхідності встановлення додаткових компонентів.
- Можливість інтеграції з HTML/CSS без компіляції – спрощує процес тестування та відладки.

2.3.3. *Chart.js*

Для візуалізації статистики прослуховувань та активності релізів було обрано бібліотеку `Chart.js` – популярний інструмент для побудови адаптивних графіків з підтримкою анімацій і інтерактивних компонентів. Вона дозволяє ефективно представити динаміку змін у вигляді графіків.

Chart.js дозволяє виводити такі типи графіків:

- лінійні графіки (Line Chart) для відображення прослуховувань за днями;
- гістограми (Bar Chart) – кількість прослуховувань по треках альбому;
- кругові діаграми (Pie/Donut Chart) – частка аудиторії з різних платформ (Spotify, Apple Music тощо).

Переваги Chart.js:

- простота інтеграції – бібліотека легко підключається через CDN або npm;
- висока гнучкість – підтримка користувацького налаштування, подій наведення та кліку;
- адаптивність – графіки масштабуються під розміри екрана;
- активна підтримка – бібліотека постійно оновлюється і має гарну документацію.

2.4. Висновки

Таким чином, обґрунтування вибору засобів реалізації базуватиметься на аналізі вимог до функціональності, продуктивності, безпеки та масштабованості програмного забезпечення ReleaseM Manager. Під час реалізації серверної частини буде використано мову програмування Python, яка забезпечить швидку розробку логіки взаємодії користувачів із системою, а також зручну й ефективну роботу з даними. Фреймворк Django відіграватиме ключову роль завдяки модульній архітектурі, вбудованій ORM, механізмам безпеки та адміністративній панелі, що дозволить суттєво оптимізувати процес створення, модерації та керування даними.

У якості системи керування базами даних буде обрано PostgreSQL, яка забезпечить надійне зберігання структурованої інформації, підтримку транзакцій, складні зв'язки між об'єктами та високу відповідність сучасним вимогам до реляційних СКБД. Її тісна інтеграція з Django дозволить

реалізувати повний спектр операцій обробки, агрегації й аналітики даних у межах застосунку.

Клієнтська частина буде реалізована на основі HTML, CSS та JavaScript, що забезпечить повний контроль над зовнішнім виглядом інтерфейсу та дозволить створити адаптивний і зручний UX. За допомогою JavaScript буде реалізовано динамічну взаємодію з формами, двоетапне додавання релізів, інтеграцію з серверною частиною через AJAX та обробку статистичних даних у реальному часі. Для візуалізації графіків прослуховувань буде використано бібліотеку Chart.js, що підвищить інформативність інтерфейсу та зручність аналізу даних. Загалом, обраний технологічний стек повністю відповідає поставленим завданням, дозволяє забезпечити ефективну взаємодію між користувачами системи, надійне зберігання даних та реалізацію складної бізнес-логіки без втрати швидкодії та зручності користування. Інструменти були підібрані з урахуванням як вимог дипломного проєкту, так і перспектив подальшого розширення функціональних можливостей системи.

3. РОЗРОБЛЕННЯ ЗАСТОСУНКУ

3.1. Загальний опис програми

Вебзастосунок ReleaseM Manager є внутрішньою інформаційною системою, призначеною для управління музичними релізами незалежних лейблів та взаємодії між артистами й менеджерами. Основна мета системи – забезпечити централізовану платформу, яка дозволить ефективно планувати, створювати, модерувати та готувати до дистрибуції музичні релізи, а також надавати повноцінний супровід після їх публікації.

Вебзастосунок побудований за клієнт-серверною архітектурою: клієнтська частина реалізує інтерфейс користувача та взаємодіє із сервером через асинхронні запити, тоді як серверна – відповідає за бізнес-логіку, обробку запитів, валідацію даних, взаємодію з базою даних та забезпечення авторизації користувачів.

Система передбачає наявність трьох основних ролей:

- Адміністратор – має повний доступ до всієї системи, керує користувачами та виконує технічне обслуговування;
- Менеджер лейблу – реєструється на платформі та створює облікові записи артистів, переглядає всі релізи, додає до них маркетингові описи, бюджети та чеклісти;
- Артист – має доступ до особистого кабінету, в якому може створювати нові релізи (сингли або альбоми), додавати треки з аудіофайлами та текстами, переглядати стан релізів, отримувати фідбек, статистику прослуховувань та повідомлення від менеджера.

Структура навігації вебзастосунку виглядає наступним чином:

- Головна сторінка – публічний лендинг, що містить опис системи, інформацію про ролі користувачів та кнопки для входу/реєстрації.
- Особистий кабінет артиста – включає секції майбутніх релізів, архіву випущених релізів, повідомлень, статистики та панелі створення релізу.

- Особистий кабінет лейблу – відображає список артистів, релізи, які відправлені на модерацію, майбутні релізи, а також інструменти для редагування метаданих.
- Сторінка майбутнього релізу – відображає повну інформацію про реліз, зокрема обкладинку, жанр, дату випуску, трек-лист, маркетинговий опис, чекліст завдань, бюджет просування, посилання на креативи, графіки прослуховувань тощо.
- Форма створення релізу – реалізована у вигляді покрокового процесу: спочатку заповнюється метаінформація релізу (тип, жанр, дата тощо), далі додаються треки (залежно від типу: один для синглу, декілька для альбому), після чого відображається попередній перегляд і виконується фінальне підтвердження.
- Статистика по релізу – відображає статистику по релізу у вигляді графіків прослуховувань та слухачів за весь час або за 7 днів.
- Статистика по артисту – відображає повну статистику по артисту та по всім його релізам.

Процес створення релізу включає внутрішню модерацію: після подання артистом, реліз переходить у статус «на перевірці», менеджер може переглянути деталі, прослухати треки, завантажити ZIP-архів з метаданими або відхилити реліз із поясненням причин. Після схвалення реліз отримує статус «до публікації» та зберігається у відповідній секції.

Усі дії менеджера зберігаються в системі, що дозволяє відстежувати історію змін та рішень по кожному релізу. Такий підхід забезпечує прозору комунікацію між артистом і менеджером.

Таким чином, застосунок покликаний автоматизувати рутинні етапи підготовки музичного матеріалу до публікації та зменшити кількість помилок у метаданих.

На рисунку 3.1 зображено функціональність вебзастосунку.

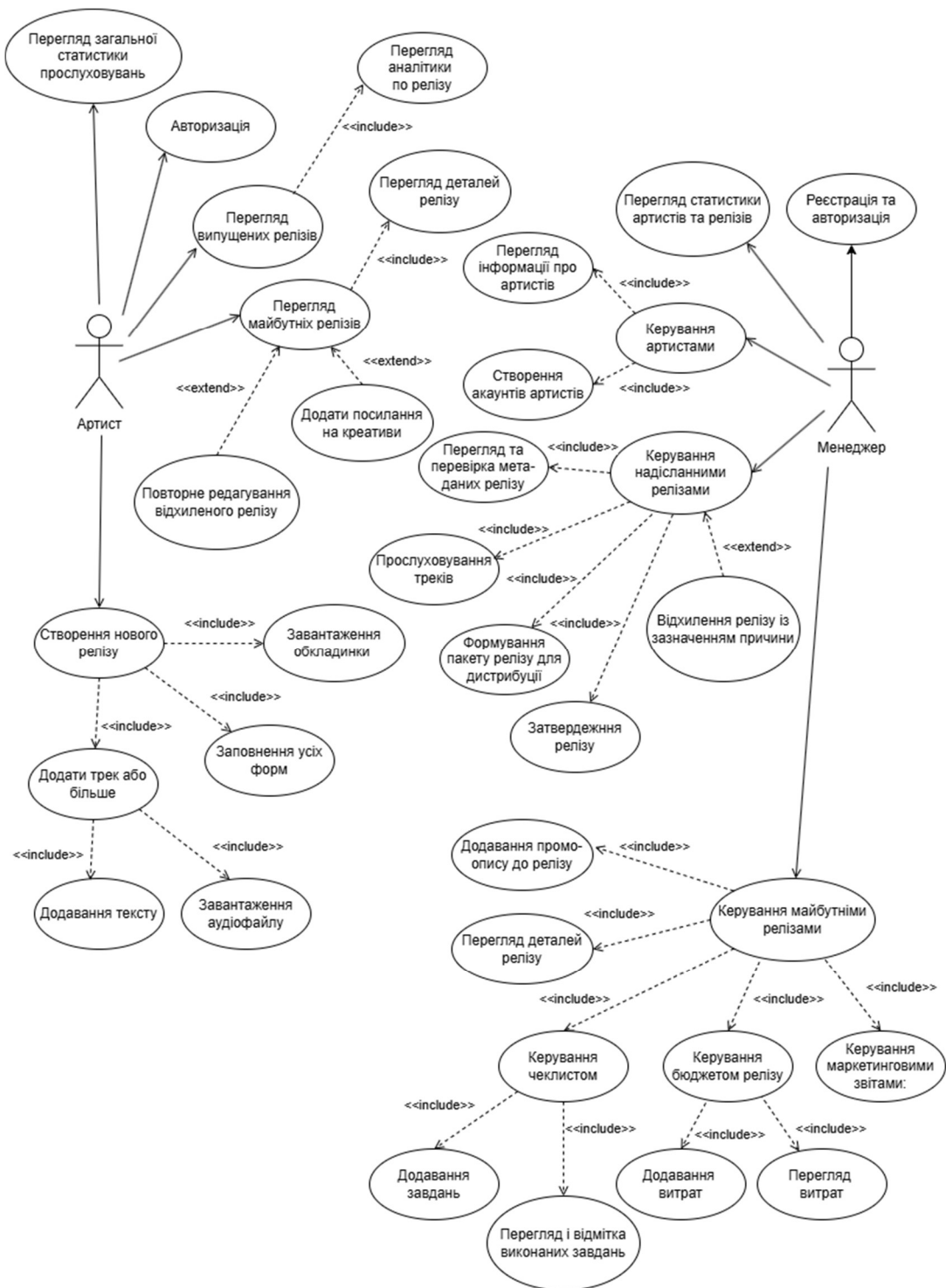


Рис. 3.1. Функціональність вебзастосунку. UML-діаграма прецендентів

3.2. Функціональні вимоги до вебзастосунку

У процесі проєктування системи ReleaseM Manager було визначено набір функціональних вимог, які відображають основні сценарії взаємодії користувачів із застосунком. Зважаючи на специфіку сервісу, в системі передбачено три ролі користувачів: менеджер лейблу та артист. Кожна роль

має доступ до власного набору функцій, які реалізовані в межах відповідного інтерфейсу.

Формування функціональних вимог дозволило визначити обов'язкові компоненти застосунку, а також розмежувати зони відповідальності між учасниками процесу випуску релізів. Перелік функціональних вимог представлено в таблиці 3.1.

Таблиця 3.1

Функціональні вимоги до вебзастосунку

Код вимоги	Назва вимоги	Опис вимоги	Роль користувача
F-1	Реєстрація лейблу	Менеджер лейблу має можливість зареєструватися на платформі, вказавши назву лейблу (яка також виступає логіном), електронну адресу та пароль. У разі помилки – система виводить повідомлення про некоректні дані.	Менеджер лейблу
F-2	Авторизація	Вхід у систему для всіх ролей здійснюється за допомогою логіна та паролю.	Всі
F-3	Створення артиста	Користувач має змогу додати нового артиста до системи, вказавши сценічне ім'я, повне ім'я, електронну адресу та завантажити фотографію. Пароль створюється автоматично і відправляється на вказану пошту	Менеджер лейблу

F-4	Додавання релізу	Користувач має мати можливість створити новий реліз, заповнюючи форму з такими метаданими: назва, тип (сингл або альбом), жанр, піджанр, обкладинка, мова, дата релізу.	Артист
F-5	Додавання треків	Користувач має мати можливість додавати треки до релізу: для синглу – один трек, для альбому – декілька. Для кожного треку заповнюється назва, додається текст пісні та завантажується аудіофайл.	Артист
F-6	Перевірка релізу	Після подання релізу артистом, менеджер лейблу має мати можливість • переглянути повну інформацію про реліз; • прослухати треки; • завантажити архів із метаданими (у форматі ZIP); • затвердити або відхилити реліз із зазначенням причини.	Менеджер лейблу
F-7	Перегляд статусу релізу	У кабінеті артиста має відображатися поточний статус кожного створеного релізу (на перевірці, відхилено, до публікації), а також причина відхилення, якщо реліз не було прийнято.	Артист

Продовження табл. 3.1

F-8	Додавання бюджету релізу	Користувач має мати можливість додавати витрати на реліз та описувати їх.	Менеджер лейблу
F-9	Формування завдань на реліз	Користувач має мати можливість додавати завдання до релізу.	Менеджер лейблу
F-10	Перегляд статистики релізу	На сторінці релізу мають відображатися графіки загальних прослуховувань та динаміки слухачів у вигляді діаграм.	Всі
F-11	Додавання маркетингового звіту	Менеджер має змогу заповнити текстовий звіт про маркетингову діяльність, які відображаються артисту.	Менеджер лейблу
F-12	Додавання посилання на креативи	Артист може додати URL-посилання, де зібрано креативні матеріали до релізу	Артист
F-13	Повідомлення про зміну статусу релізу	Після схвалення або відхилення релізу менеджером, артист отримує сповіщення у своєму кабінеті. У разі відхилення також вказується причина.	Артист
F-14	Редагування відхиленого релізу	Артист має змогу внести зміни до релізу, що був відхилений менеджером, і повторно надіслати його на перевірку.	Артист

3.3. Особливості реалізації серверної частини

Серверна частина вебзастосунку ReleaseM Manager реалізована з дотриманням принципів модульного моноліту. Такий підхід забезпечує логічне розділення функціональних складових системи при збереженні єдиного процесу розгортання. Модульний моноліт поєднує в собі переваги монолітної архітектури у вигляді простоти обслуговування та високої продуктивності із частковою ізолюваністю функціональних блоків, притаманною мікросервісам. Це дозволяє забезпечити зрозумілу внутрішню структуру, спростити масштабування та підтримку системи в майбутньому.

Модульна структура була сформована на основі аналізу функціональних вимог до застосунку. Кожен модуль відповідає за окремий напрямок роботи системи, при цьому чітко визначено межі відповідальності між ними. Для побудови серверної частини було використано фреймворк Django, що дозволив ефективно реалізувати структуру модулів через apps.

Основні модулі серверної частини.

- **UserModule** – відповідає за автентифікацію та авторизацію користувачів, управління обліковими записами, створення артистів менеджером лейблу.
- **ReleaseModule** – основний модуль, що забезпечує роботу з релізами. Містить механізми створення релізів, додавання треків, подання релізу на перевірку, а також відображення статусів для артистів. Тут також реалізовано завантаження метаданих та аудіофайлів.
- **ModerationModule** – забезпечує модерацію релізів з боку менеджера: перевірку метаданих, прослуховування треків, схвалення або відхилення релізу з поясненням причини. У межах модуля також реалізовано логіку надсилання повідомлень артисту про зміну статусу.
- **InteractionModule** – реалізує взаємодію між артистом і менеджером після підтвердження релізу. До функціональних можливостей входить додавання бюджету, створення списку завдань, написання

маркетингового опису, а також надання артистом посилання на креативи (наприклад, Google Drive).

- **AnalyticsModule** – містить механізми генерації та відображення статистичних даних про релізи: загальна кількість прослуховувань, динаміка слухачів, порівняльна статистика по треках альбому. Дані візуалізуються через бібліотеку Chart.js.

Кожен із цих модулів реалізований таким чином, щоб у разі необхідності масштабування вебзастосунку можна було додати нові функціональні можливості до наявних модулів. Такий підхід відповідає сучасним вимогам до архітектури вебзастосунків.

3.3.1. Особливості реалізації обробки HTTP запитів

Обробка HTTP-запитів є однією з ключових функцій серверної частини вебзастосунку. Кожна взаємодія між клієнтом і сервером відбувається у вигляді запиту (наприклад, GET, POST, PUT, DELETE), на який сервер формує відповідь. У межах Django ця логіка реалізована за допомогою системи маршрутів (URL dispatcher), представлень (views) та шаблонів. Django дозволяє будувати як синхронні, так і асинхронні (через AJAX) сценарії обробки запитів, що забезпечує високий рівень інтерактивності застосунку.

У вебзастосунку ReleaseM Manager кожен запит, надісланий користувачем, проходить через кілька основних етапів.

- **Маршрутизація:** URL-запит порівнюється з налаштованими маршрутами у файлах `urls.py`. Система визначає, якому обробнику (view) передати запит.
- **Обробка у представленнях:** відповідний клас або функція у `views.py` отримує дані, перевіряє права доступу, формує запит до бази даних через ORM та готує відповідь.
- **Формування відповіді:** залежно від сценарію, відповідь може бути HTML-сторінкою (у випадку класичних запитів) або JSON-даними

(для асинхронних AJAX-запитів), що оновлюють частини інтерфейсу без перезавантаження сторінки.

- Відправлення відповіді клієнту: Django автоматично обгортає відповідь у відповідний формат HTTP-відповіді, яку клієнтська частина обробляє й відображає користувачеві.

Особливістю реалізації є застосування асинхронних запитів у таких сценаріях, як оновлення статусу релізу, динамічне додавання треків, побудова графіків статистики, що суттєво покращує користувацький досвід. На рисунку 3.2 схематично представлено процес обробки HTTP запитів.

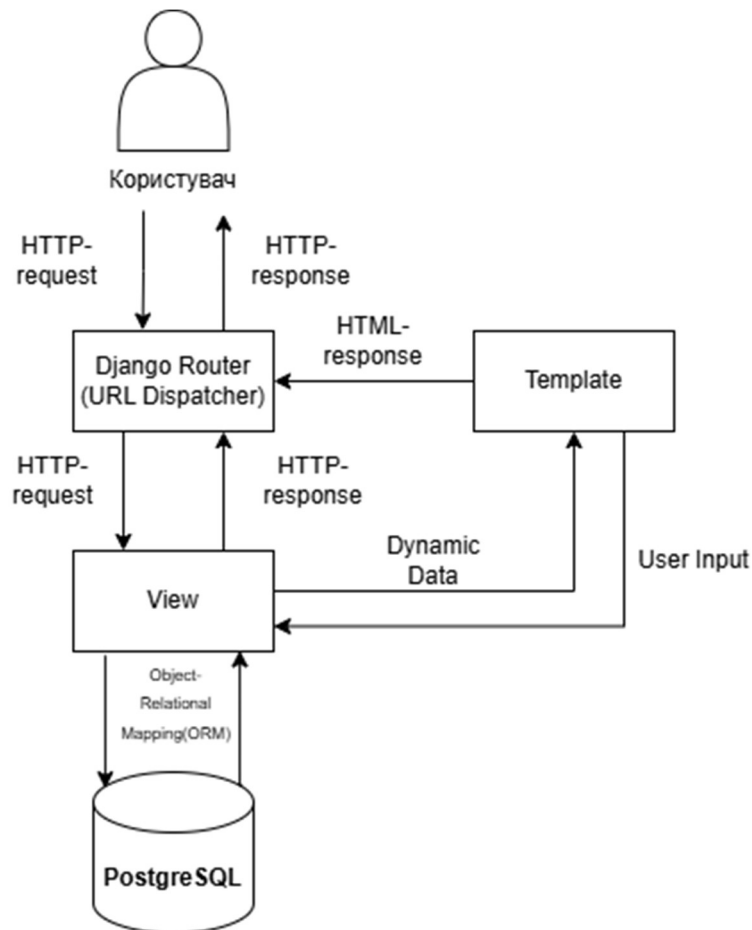


Рис. 3.2. Схема обробки HTTP запитів

Крім того, для реалізації повного циклу управління даними у вебзастосунку ReleaseM Manager» реалізовано такі ключові маршрути:

- /release/<release_id>/marketing-report – оновлення маркетингового

опису релізу;

- `/budget-item/<item_id>/delete` – видалення елемента бюджету;
- `/release/<release_id>/add-budget-item` – додавання витрати до бюджету релізу;
- `/release/<release_id>/checklist/add` – додавання нового елемента до чекліста релізу;
- `/label/download-release/<release_id>` – завантаження архіву з метаданими релізу.

Ці маршрути демонструють, як система використовує різні типи HTTP-запитів для виконання конкретних бізнес-операцій, забезпечуючи зручність керування та високий рівень інтерактивності інтерфейсу.

3.3.2. Особливості реалізації обробки файлів

У сучасних вебзастосунках робота з медіа та файлами є невіддільною частиною функціональності, особливо коли мова йде про музичну сферу. У застосунку ReleaseM Manager артисти завантажують аудіофайли, обкладинки релізів, а також прикріплюють зовнішні посилання на креативні матеріали. Менеджери лейблів, у свою чергу, завантажують ZIP-архіви з метаданими релізів для подальшої модерації та дистрибуції релізу.

Для обробки файлів у Django використовуються такі основні механізми:

- `MEDIA_ROOT` і `MEDIA_URL` – налаштування, що визначають місце збереження медіафайлів на сервері та доступ до них через HTTP.
- `FileField` / `ImageField` у моделях – спеціальні типи полів, що дозволяють зберігати посилання на файли, а також завантажувати їх через форми.
- Валідація файлів – обмеження на формат файлів (наприклад, лише `.mp3` або `.jpg`) та розмір файлу, щоб уникнути завантаження небезпечних або надмірно великих даних.

Користувач завантажує файл через HTML-форму, після чого дані обробляються в `views.py` і зберігаються у спеціально відведеній директорії. Після чого у базі зберігається не сам файл, а шлях до нього що дозволяє ефективно керувати даними. Публічні медіафайли стають доступними через посилання на `MEDIA_URL`. Для обробки великих креативних файлів для маркетингу релізу замість фізичного завантаження передбачено додавання URL-посилань, що зберігаються як текстові поля в базі даних.

3.3.3. Особливості реалізації генерації даних

Для забезпечення демонстраційних можливостей вебзастосунку, а також для тестування функціональних модулів і перевірки масштабованості системи, було реалізовано механізм автоматизованої генерації даних. Генерація даних дозволяє наповнити систему реалістичними, але штучно створеними записами, що дозволяє перевірити роботу графіків, функціональних елементів навіть без залучення реальних користувачів. Ця логіка реалізована за допомогою `management`-команд Django, які можна запускати через консоль. Зокрема, команда `generate_statistics` дозволяє заповнити таблицю статистики треків, створюючи дані прослуховувань і слухачів по кожному треку за період від дати релізу до поточного дня. Для більшої правдоподібності система враховує масштаб артиста (наприклад, відомі артисти мають вищі показники, ніж маловідомі) та варіацію платформ-джерел (Spotify, Apple Music, YouTube тощо). Цей підхід дозволяє: перевіряти адекватність відображення динамічних графіків, тестувати навантаження на систему та демонструвати готовність продукту потенційним користувачам або інвесторам.

Важливо зазначити, що в майбутньому передбачено інтеграцію з API музичних платформ (наприклад, Spotify, Apple Music, YouTube Music), щоб отримувати реальні дані прослуховувань та інших показників без потреби штучного наповнення. Це розширить аналітичні можливості застосунку та

значно збагатить інтерфейси статистики. Приклад генерації даних прослуховувань наведено в лістингу 3.1.

Лістинг 3.1. Генерація даних прослуховувань

```
class Command(BaseCommand):
    def handle(self, *args, **kwargs):
        today = date(2025, 4, 13)
        created = 0

        tracks = Track.objects.select_related('release', 'release__artist')
        \.filter(release__status='released')

        for track in tracks:
            artist_name = track.release.artist.name
            release_date = track.release.release_date
            if release_date > today:
                continue
            days_total = (today - release_date).days + 1

            for i in range(days_total):
                stat_date = release_date + timedelta(days=i)

                for source in random.sample(SOURCES, k=random.randint(2,
4)):
                    if artist_name in STABLE_BIG:
                        base = random.randint(500, 30000)
                        variation = int(base * 0.1)
                        plays = random.randint(base - variation, base +
variation)

                        elif artist_name in UNSTABLE_SMALL:
                            base = random.randint(300, 8000)
                            variation = int(base * 0.6)
                            plays = max(20, random.randint(base - variation,
base + variation))

                        else:
                            plays = random.randint(100, 5000)

                            min_listeners = max(5, plays // 6)
                            max_listeners = max(min_listeners + 1, plays // 4)
                            listeners = random.randint(min_listeners,
max_listeners)

                            stat, created_flag = TrackStat.objects.get_or_create(
                                track=track,
                                date=stat_date,
                                source=source,
                                defaults={
                                    'plays': plays,
                                    'listeners': listeners
                                }
                            )
                            if created_flag:
                                created += 1
```

3.4. Архітектура бази даних

У вебзастосунку ReleaseM Manager база даних є ключовим компонентом, що забезпечує зберігання всіх даних, пов'язаних із користувачами, релізами, треками, бюджетами, чеклістами, повідомленнями та статистикою. Схема бази даних побудована так, щоб відображати логіку застосунку і розділяти дані за сутностями. Кожна таблиця в системі представляє окрему логічну одиницю та має свої унікальні характеристики.

При проєктуванні структури бази даних враховано:

- чіткий поділ на сутності відповідно до бізнес-логіки;
- мінімізацію дублювання даних внаслідок винесення повторюваної інформації в окремі таблиці;
- забезпечення масштабованості – можливість розширення структури без необхідності переробляти основну схему.

Схема бази даних показана на ER-діаграмі (рис. 3.3).

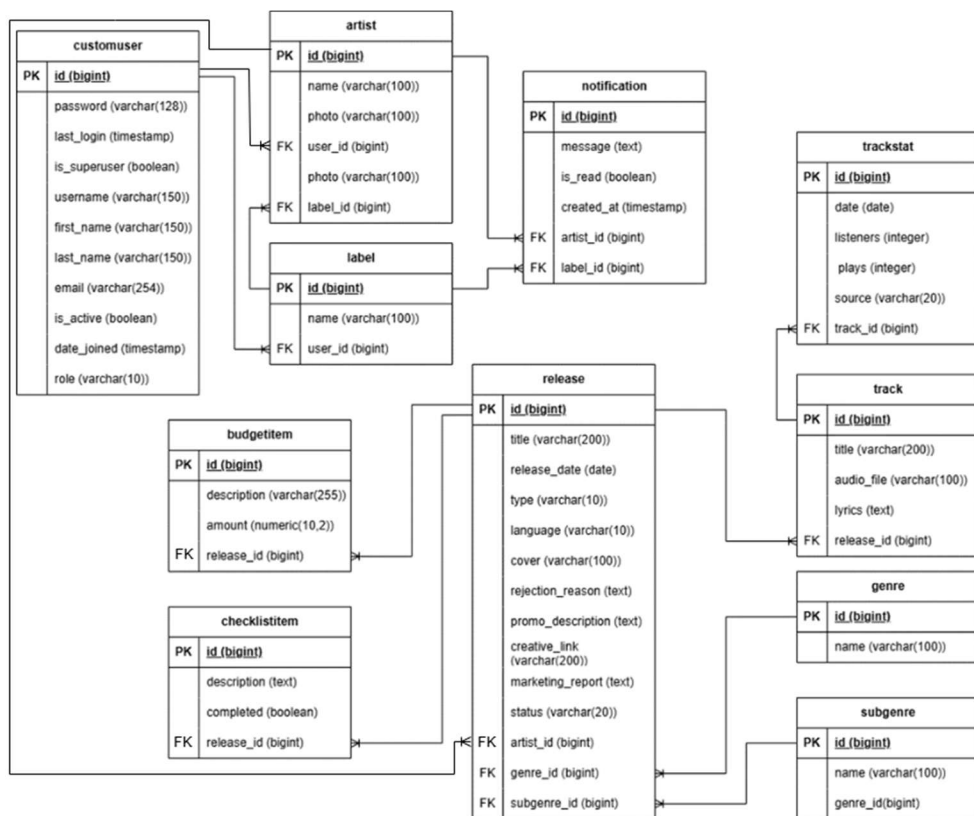


Рис. 3.3. Схема бази даних

3.4.1. Опис таблиць та зв'язків між ними

Customuser

Таблиця Customuser зберігає дані про зареєстрованих користувачів системи, включаючи артистів, менеджерів лейблів та адміністраторів. Опис полів таблиці Customuser наведено у табл. 3.2.

Таблиця 3.2

Поля таблиці Customuser

Назва поля	Тип даних	Опис поля
Id	bigint	Первинний ключ, ідентифікатор користувача
password	varchar(128)	Захешований пароль користувача
last_login	timestamp	Дата й час останнього входу користувача
is_superuser	boolean	Чи є користувач суперкористувачем (адміністратором)
username	varchar(150)	Унікальний логін користувача
first_name	varchar(150)	Ім'я користувача
last_name	varchar(150)	Прізвище користувача
email	varchar(254)	Електронна пошта користувача
is_active	boolean	Чи активовано обліковий запис
date_joined	timestamp	Дата приєднання користувача до системи
role	varchar(10)	Роль користувача (artist, label)

Label

Таблиця Label зберігає дані про музичні лейбли, які керують артистами в системі. Між Customuser та Label існує зв'язок один-до-

багатьох. Опис полів таблиці Label наведено у табл. 3.3.

Таблиця 3.3

Поля таблиці Label

Назва поля	Тип даних	Опис поля
Id	bigint	Первинний ключ, ідентифікатор користувача
name	varchar(100)	Назва лейблу
user_id	bigint (FK)	Зовнішній ключ на customuser

Artist

Таблиця Artist зберігає інформацію про артистів, які підпорядковані лейблам. Між label та Artist – зв’язок один-до-багатьох. Опис полів таблиці Artist наведено у табл. 3.4.

Таблиця 3.4

Поля таблиці Artist

Назва поля	Тип даних	Опис поля
Id	bigint	Первинний ключ, ідентифікатор користувача
name	varchar(100)	Сценічне ім’я артиста
photo	varchar(100)	Шлях до фото артиста
user_id	bigint (FK)	Зовнішній ключ на customuser
label_id	bigint (FK)	Зовнішній ключ на лейбл артиста

Release

Таблиця Release зберігає дані про музичні релізи (альбоми, сингли), що створюються артистами. Між Artist та Release – зв’язок один-до-

багатьох. Між Genre/Subgenre та Release – зв’язок один-до-багатьох. Опис полів таблиці Release наведено у табл. 3.5.

Таблиця 3.5

Поля таблиці Release

Назва поля	Тип даних	Опис поля
id	bigint	Первинний ключ, ідентифікатор релізу
title	varchar(200)	Назва релізу
release_date	date	Дата випуску релізу
type	varchar(10)	Тип релізу (single/album)
language	varchar(10)	Мова релізу
cover	varchar(100)	Шлях до обкладинки релізу
rejection_reason	text	Причина відхилення (якщо є)
promo_description	text	Промоційний опис релізу
creative_link	varchar(200)	Посилання на креативи
marketing_report	text	Звіт про маркетингову діяльність
status	varchar(20)	Статус релізу
artist_id	bigint (FK)	Зовнішній ключ на artist
genre_id	bigint (FK)	Зовнішній ключ на genre
subgenre_id	bigint (FK)	Зовнішній ключ на subgenre

Track

Таблиця Track зберігає дані про окремі треки, що входять до релізів. Між Release та Track – зв’язок один-до-багатьох. Опис полів таблиці Track наведено у табл. 3.6.

Таблиця 3.6

Поля таблиці Track

Назва поля	Тип даних	Опис поля
id	bigint	Первинний ключ, ідентифікатор релізу
title	varchar(200)	Назва релізу
audio_file	varchar(100)	Шлях до аудіофайлу
lyrics	text	Текст пісні
release_id	bigint (FK)	Зовнішній ключ на release. Прив'язка треку до релізу

Trackstat

Таблиця Trackstat зберігає статистику по треках – кількість прослуховувань, слухачів тощо. Між Track та Trackstat – зв'язок один-до-багатьох. Опис полів таблиці Trackstat наведено у табл. 3.7.

Таблиця 3.7

Поля таблиці Trackstat

Назва поля	Тип даних	Опис поля
id	bigint	Первинний ключ, ідентифікатор запису
date	date	Дата запису статистики
listeners	integer	Кількість унікальних слухачів
plays	integer	Кількість програвань треку
source	varchar(20)	Джерело (платформа, сайт)
track_id	bigint (FK)	Зовнішній ключ на track

Genre

Таблиця Genre містить список доступних жанрів, що використовуються для класифікації релізів. Опис полів таблиці Genre наведено у табл. 3.8.

Таблиця 3.8

Поля таблиці Genre

Назва поля	Тип даних	Опис поля
id	bigint	Первинний ключ, ідентифікатор жанру
name	varchar(100)	Назва жанру

Subgenre

Таблиця Subgenre містить піджанри, пов'язані з основними жанрами. Між Genre та Subgenre – зв'язок один-до-багатьох. Опис полів таблиці Subgenre наведено у табл. 3.9.

Таблиця 3.9

Поля таблиці Subgenre

Назва поля	Тип даних	Опис поля
id	bigint	Первинний ключ, ідентифікатор піджанру
name	varchar(100)	Назва піджанру
genre_id	bigint (FK)	Зовнішній ключ на genre

Checklistitem

Таблиця Checklistitem зберігає завдання, що формуються менеджером лейблу для підготовки релізу. Між Release та Checklistitem – зв'язок один-до-багатьох. Опис полів таблиці Checklistitem наведено у табл. 3.10.

Таблиця 3.10

Поля таблиці Checklistitem

Назва поля	Тип даних	Опис поля
id	bigint	Первинний ключ, ідентифікатор завдання
description	text	Опис завдання
completed	boolean	Статус виконання (true/false)
release_id	bigint (FK)	Зовнішній ключ на release

Budgetitem

Таблиця Budgetitem зберігає дані про витрати, пов'язані з релізом. Між Release та Budgetitem – зв'язок один-до-багатьох. Опис полів таблиці Budgetitem наведено у табл. 3.11.

Таблиця 3.11

Поля таблиці Budgetitem

Назва поля	Тип даних	Опис поля
id	bigint	Первинний ключ, ідентифікатор витрати
description	varchar(255)	Опис витрати
amount	numeric(10,2)	Сума витрати
release_id	bigint (FK)	Зовнішній ключ на release

Notification

Таблиця Notification містить повідомлення для артистів та менеджерів лейблів. Має зовнішні ключі на Artist та Label. Опис полів таблиці Notification наведено у табл. 3.12.

Поля таблиці Notification

Назва поля	Тип даних	Опис поля
id	bigint	Первинний ключ, ідентифікатор повідомлення
message	text	Текст повідомлення
is_read	boolean	Чи прочитано повідомлення
created_at	timestamp	Дата та час створення повідомлення
artist_id	bigint (FK)	Зовнішній ключ на artist
label_id	bigint (FK)	Зовнішній ключ на label

3.5. Особливості реалізації клієнтської частини застосунку

Клієнтську частину вебзастосунку ReleaseM Manager реалізовано як класичний багатосторінковий застосунок із чітко визначеними маршрутами. Така структура дозволяє ефективно розділити функціональні зони для різних ролей користувачів – артистів і менеджерів лейблів – та забезпечує зрозумілу навігацію між основними екранами системи.

Інтерфейс побудовано з урахуванням принципів адаптивного дизайну. Головне меню та вкладки забезпечують швидкий перехід між ключовими розділами, а окремі сторінки відповідають за конкретні задачі користувача: створення релізу, модерацію, перегляд статистики, управління бюджетами тощо.

Основні маршрути клієнтської частини:

- / – маршрут для відображення головної сторінки застосунку;
- /artist/login – маршрут для авторизації артиста;
- /artist/dashboard – маршрут для кабінету артиста;
- /artist/logout – маршрут для виходу артиста з системи;
- /artist/add-release – маршрут для створення нового релізу артистом;

- /register/label – маршрут для реєстрації нового лейблу;
- /dashboard/label – маршрут для кабінету менеджера лейблу;
- /label/add-artist – маршрут для додавання нового артиста менеджером лейблу;
- /label/moderate-release/<release_id> – маршрут для перегляду модератором релізу з конкретним ідентифікатором;
- /artist/edit-release/<release_id> – маршрут для редагування відхиленого релізу артистом;
- /artist/notifications – маршрут для перегляду сповіщень артиста;
- /release/<release_id>/future – маршрут для перегляду сторінки майбутнього релізу;
- /artist/<artist_id>/stats – маршрут для перегляду статистики артиста;
- /release/<release_id> – маршрут для перегляду детальної сторінки релізу;

Клієнтська частина застосунку тісно інтегрована із серверною через HTTP-запити, що дозволяє реалізувати повноцінну динамічну взаємодію з базою даних: додавання, редагування та відображення даних у режимі реального часу.

3.6. Висновки

Вебзастосунок ReleaseM Manager побудовано на основі клієнт-серверної архітектури, що дає змогу чітко розділити обов'язки між двома частинами системи. Сервер відповідає за реалізацію бізнес-логіки, обробку HTTP-запитів, роботу з базою даних і управління інформацією, тоді як клієнтська частина забезпечує взаємодію з користувачем, відображення отриманих даних та формування запитів до сервера.

Серверна частина застосунку реалізована у вигляді модульного моноліту, що дало змогу впорядкувати функціональні можливості за окремими модулями, кожен з яких виконує свої чітко визначені завдання. Зокрема, було виокремлено такі ключові модулі: UserModule,

ReleaseModule, ModerationModule, InteractionModule та AnalyticsModule. Такий підхід спростив реалізацію основних функціональних можливостей функціональних можливостей і водночас забезпечив масштабованість та готовність системи до розширення – як шляхом додавання нових модулів, так і шляхом доповнення вже наявних.

Для забезпечення стабільної роботи системи було спроектовано структуру бази даних, що охоплює всі основні сутності (користувачі, лейбли, артисти, релізи, треки, бюджети, чеклісти, повідомлення, статистика) та формує цілісну модель з правильно налаштованими зв'язками. Це дозволяє виконувати всі необхідні CRUD-операції, а також будувати аналітичні звіти та переглядати статистику.

Клієнтська частина розроблена як багатосторінковий вебінтерфейс, де кожен маршрут відповідає за окрему функціональність (кабінет артиста, кабінет лейблу, модерація, управління релізами, перегляд звітів тощо). Навігація між розділами здійснюється через головне меню та вкладки, а обмін даними із сервером забезпечує динамічне оновлення інформації.

Таким чином, розроблена архітектура вебзастосунку ReleaseM Manager дає змогу ефективно управляти даними, пропонувати користувачам зручний інтерфейс і створює міцну основу для подальшого розвитку системи – зокрема для інтеграції API зовнішніх музичних майданчиків, автоматизації процесів дистрибуції та розширення можливостей аналітики.

4. АНАЛІЗ РОЗРОБЛЕНОГО ВЕБЗАСТОСУНКУ

4.1. Аналіз реалізованого вебзастосунку

У ході розробки програмного забезпечення було створено вебзастосунок «ReleaseM Manager», призначений для управління музичними релізами незалежних лейблів. Система надає широкий спектр інструментів для адміністрування артистів, підготовки релізів, а також для організації взаємодії між артистами та менеджерами лейблів. Основні функціональні можливості вебзастосунку охоплюють створення релізів, завантаження треків, управління метаданими, модерацію релізів менеджерами, аналіз статистики прослуховувань, формування чеклістів та планування маркетингового бюджету.

Доступ до головних можливостей системи надається після авторизації, що здійснюється за обліковими даними, створеними адміністратором або менеджером лейблу. Залежно від ролі користувача (менеджер лейблу, артист) після входу до системи відкривається персоналізований кабінет із відповідним набором можливостей.

Основна взаємодія з релізами та їх модерацією відбувається у відповідних кабінетах користувачів. Артисти можуть формувати релізи, додавати до них треки, вказувати жанр, мову, дату виходу, завантажувати обкладинки та аудіофайли. Менеджери лейблів здійснюють перевірку даних релізу, складають бюджет, формують перелік завдань, а також приймають рішення щодо затвердження або відхилення релізу з поясненням причин. Затверджені релізи переходять до секції майбутніх релізів, де можна відстежувати виконання запланованих завдань і переглядати аналітику.

Інтерфейс застосунку реалізовано з використанням AJAX-запитів, що дозволяє оновлювати дані на сторінці без повного її перезавантаження. Це дає змогу, наприклад, артистам додавати треки у реліз динамічно, а менеджерам – відмічати виконані пункти у чеклісті в режимі реального

часу. Розділ аналітики пропонує графічні візуалізації (лінійні графіки, гістограми), що дозволяє оперативно аналізувати результати релізів.

Серверна частина вебзастосунку виконує роль центрального вузла, що відповідає за обробку запитів користувачів, роботу з базою даних PostgreSQL та реалізацію логіки доменної області. Вона слугує посередником між інтерфейсом користувача та даними, забезпечуючи коректне виконання бізнес-процесів, контроль за правами доступу, обробку помилок, а також збереження і цілісність даних.

У підсумку, розроблений вебзастосунок відповідає заявленим вимогам, надаючи користувачам потужний інструмент для управління процесами релізу, а також створює основу для подальшого розширення функціональних можливостей та інтеграцій.

4.2. Тестування реалізованого вебзастосунку

Тестування вебзастосунку «ReleaseM Manager» проводилося з метою перевірки стабільності, правильності реалізації функцій, відповідності очікуваній поведінці та надійності роботи системи під різними сценаріями. Особлива увага приділялася коректності взаємодії між користувачами з різними ролями (артист, менеджер лейблу), а також точності збереження та обробки метаданих, релізів і статистики. У межах реалізації проєкту було прийнято рішення здійснювати ручне тестування. Воно проводилось поетапно на кожному етапі впровадження нових функціональних можливостей. Такий підхід дозволив оперативно виявляти і виправляти помилки без накопичення технічного боргу.

Процес тестування умовно поділявся на п'ять основних етапів:

- визначення об'єкта тестування;
- формування тестових сценаріїв;
- проведення тестування;
- фіксація та усунення дефектів;
- контрольне тестування після виправлень.

У результаті проведеного ручного тестування реалізованого вебзастосунку було перевірено основні функціональні можливості для всіх типів користувачів – артистів і менеджерів лейблів. Всі функціональні можливості працювали відповідно до вимог. На таблицях 4.1-4.3 продемонстровані сценарії тестування вебзастосунку.

Таблиця 4.1

Сценарії тестування авторизації та доступу

№	Назва сценарію	Передумова	Сценарій дій	Очікуваний результат
1	Авторизація артиста	Артиста створено в системі менеджером	1. Користувач переходить на сторінку входу. 2. Вводить сценічне ім'я, назву лейблу і пароль. 3. Натискає кнопку «Увійти».	Користувач переходить до свого кабінету артиста.
2	Авторизація менеджера лейблу	Менеджера зареєстровано через публічну форму.	1. Перехід на сторінку входу. 2. Введення логіна і пароля. 3. Вхід.	Користувач потрапляє до кабінету менеджера.
3	Невдала авторизація	Неправильні облікові дані	1. Введення невірної логіна або пароля. 2. Натискання кнопки входу.	Виводиться повідомлення про помилку авторизації.
4	Спроба доступу до чужого релізу	Артист намагається переглянути не свій реліз	Введення прямого URL до чужого релізу.	Виводиться повідомлення про відмову в доступі або перенаправлення.

Таблиця 4.2

Сценарії тестування функціональних можливостей для менеджера лейблу

№	Назва сценарію	Передумова	Сценарій дій	Очікуваний результат
1	Додавання артиста	Менеджер лейблу авторизований у системі	1. Користувач натискає кнопку «Додати артиста» в своєму персональному кабінеті 2. Заповнює дані (сценічне ім'я, email, ПІБ). 3. Натискає «Створити».	Новий артист з'являється у списку; створюється обліковий запис з автоматично згенерованим паролем, який надсилається на вказану пошту артиста.
2	Перегляд релізів на перевірку	У систему додано реліз артистом	1. Користувач переходить у розділ «Релізи на модерацію». 2. Обирає реліз для перегляду.	Відкривається сторінка з усіма метаданими релізу та файли прикріплені артистом.
3	Схвалення релізу	Реліз перебуває у статусі «очікує перевірки»	1. Менеджер відкриває реліз. 2. Натискає кнопку «Схвалити реліз».	Статус релізу змінюється на «схвалено», артист бачить оновлення і отримує повідомлення про це у своєму кабінеті.
4	Відхилення релізу	Реліз перевіряється користувачем	1. Менеджер натискає кнопку «Відхилити». 2. Вводить причину. 3. Підтверджує дію.	Реліз отримує статус «відхилено», артист бачить причину в інтерфейсі, а також отримує повідомлення.

Продовження табл. 4.2

5	Додавання чекліста	Реліз має статус «майбутній»	1. Менеджер відкриває реліз. 2. В полі «Чекліст» додає нове завдання. 3. Зберігає зміни.	Завдання з'являється у списку, інформація доступна для перегляду артистом..
6	Редагування бюджету	Реліз має статус «майбутній»	1. Менеджер відкриває реліз. 2. Додає суму витрати 3. Зберігає.	Бюджет зберігається, інформація доступна для перегляду артистом.
7	Завантаження метаданих	Реліз перебуває у статусі «очікує перевірки»	1. Менеджер відкриває реліз. 2. Натискає кнопку «Завантажити метадані». 3. Система формує архів.	ZIP-файл із метаданими релізу завантажується на комп'ютер менеджера.

Таблиця 4.3

Сценарії тестування функціональних можливостей артиста

№	Назва сценарію	Передумова	Сценарій дій	Очікуваний результат
1	Створення релізу	Артист авторизований	1. Артист натискає «Додати реліз». 2. Заповнює поля і завантажує обкладинку. 3. Додає один або декілька треків в залежності від типу релізу. 4. Зберігає.	Реліз створено, статус – «очікує модерації».

Продовження табл. 4.3

2	Повторна подача відхиленого релізу	Реліз має статус «відхилено»	1. Артист редагує дані. 2. Натискає «Надіслати повторно».	Реліз знову потрапляє на модерацію.
3	Додавання посилання на креативи	Реліз має статус «майбутній»	1. Артист відкриває реліз. 2. У відповідному полі вводить посилання на Google Диск. 3. Натискає «Зберегти».	Посилання зберігається, менеджер може його переглянути.
4	Перегляд статистики	Реліз опубліковано та має згенеровані метрики	1. Артист відкриває реліз. 2. Перемикає графіки за періодами.	Відображаються аналітичні графіки та показники (платформи, слухачі тощо).
5	Перегляд повідомлень	Артист має нові сповіщення у системі	1. Артист натискає на іконку дзвіночка у верхній панелі. 2. Відкривається список повідомлень.	Виводиться список повідомлень із індикатором непрочитаних; після перегляду вони стають прочитаними.

4.3. Рекомендації для майбутнього вдосконалення застосунку

Хоча розроблений вебзастосунок «ReleaseM Manager» уже виконує ключові функції з управління музичними релізами та взаємодії між артистами й менеджерами лейблів, у системі зберігається простір для подальшого розвитку та доповнення новими можливостями.

Щоб зробити платформу більш гнучкою, масштабованою та зручною для роботи в різних сценаріях, можна виокремити низку напрямів для її

майбутнього вдосконалення, зокрема тих, що сприятимуть розширенню функціональності:

- Інтеграція з платформами цифрової дистрибуції (Spotify, Apple Music, YouTube Music тощо) через API-партнерів або агрегаторів. Це дозволить автоматизувати публікацію релізів, зменшити кількість ручної роботи та підвищити швидкість виведення треків на ринок.
- Розширення можливостей взаємодії між артистами та менеджерами. Додавання внутрішніх коментарів до релізів, можливості обговорення у вигляді чату або нотаток покращить колабораційний процес у межах платформи.
- Автоматизоване створення контрактів між артистами та лейблом з використанням шаблонів у форматі PDF та можливістю їх підписання за допомогою електронного підпису.
- Мобільна версія вебзастосунку або створення PWA (Progressive Web App). Забезпечення повноцінної роботи на мобільних пристроях дозволить артистам оперативно керувати своїми релізами, переглядати статус модерації, отримувати повідомлення та завантажувати креативи без використання настільного браузера.
- Запровадження системи сповіщень (email або push) про зміни статусів релізу, повідомлення менеджера, кінцеві терміни у чеклісті або нові статистичні звіти.
- Можливість створення публічної сторінки артиста з переліком його релізів, описом, посиланнями на стримінгові платформи та контактами. Така сторінка може використовуватись для просування артиста поза межами платформи.
- Підтримка інтерфейсу кількома мовами дозволить масштабувати вебзастосунок на міжнародну аудиторію та сприятиме підвищенню користувацької лояльності.

Усі зазначені рекомендації можуть бути реалізовані у наступних ітераціях проєкту, що дозволить суттєво підвищити як функціональну повноту системи, так і її цінність для кінцевих користувачів.

4.4. Висновки

У цьому розділі було здійснено комплексний аналіз реалізованого вебзастосунку «ReleaseM Manager», що функціонує як внутрішня система управління музичними релізами для незалежних лейблів. Система реалізована у вигляді MVP – мінімально життєздатного продукту, який охоплює базовий набір функціональних можливостей, достатній для тестування концепції та роботи з реальними користувачами.

Було детально проаналізовано основні компоненти системи, описано її функціональну логіку та протестовано ключові сценарії використання, зокрема: авторизацію, створення й модерацію релізів, управління бюджетами, перегляд статистики, роботу з чеклістами та внутрішні повідомлення. Усі критичні функції успішно пройшли ручне тестування, що підтвердило відповідність їх реалізації заданим вимогам.

На основі отриманих результатів сформовано перелік перспектив для подальшого розвитку системи: інтеграція зі стримінговими платформами, створення мобільної версії, автоматизація юридичних процесів, підтримка кількох мов та інші покращення, що підвищать цінність продукту для користувачів.

Таким чином, розроблений вебзастосунок демонструє готовність до впровадження в рамках цільової аудиторії та закладає основу для масштабування функціональності у наступних версіях.

ВИСНОВКИ

Метою даного дипломного проєкту було створення вебзастосунку «ReleaseM Manager», який забезпечує централізоване управління процесом підготовки та випуску музичних релізів для незалежних лейблів та артистів.

У першому розділі було досліджено чинні програмні рішення, що надають сервіси для дистрибуції або менеджменту музичних релізів. Виявлено, що більшість з них не дозволяє ефективно координувати взаємодію між артистами, менеджерами та дистриб'юторами в одному середовищі, не пропонуючи гнучкого та адаптивного інструменту для роботи з усім життєвим циклом релізу. Це обґрунтовує доцільність створення власного рішення.

У другому розділі обґрунтовано вибір технологічного стека для реалізації серверної та клієнтської частин. Зокрема, обрано мову програмування Python та фреймворк Django як основу серверної логіки завдяки їхній продуктивності, безпечності та гнучкості. Для зберігання даних використано реляційну базу даних PostgreSQL. Також було застосовано JavaScript для реалізації інтерактивних елементів інтерфейсу користувача.

Було описано загальну структуру розробленого застосунку, ключові модулі та реалізовані функціональні можливості. Реалізація системи базувалась на архітектурі «клієнт-сервер» та шаблоні MVT (Model-View-Template), що забезпечує модульність та масштабованість проєкту.

Було проведено оцінку ефективності реалізованого MVP-варіанту системи. Застосунок продемонстрував здатність вирішувати поставлену проблему централізованого управління релізами незалежних артистів.

Таким чином, поставлену мету дипломної роботи досягнуто повністю: було створено MVP інформаційної системи для підтримки процесів створення, підготовки та випуску музичних релізів незалежними лейблами.

Застосунок «ReleaseM Manager» не лише закриває функціональні потреби обраної цільової аудиторії, а й створює підґрунтя для подальшого розвитку: інтеграції інструментів аналітики, автоматичного опису релізів, API-зв'язків з платформами дистрибуції тощо.

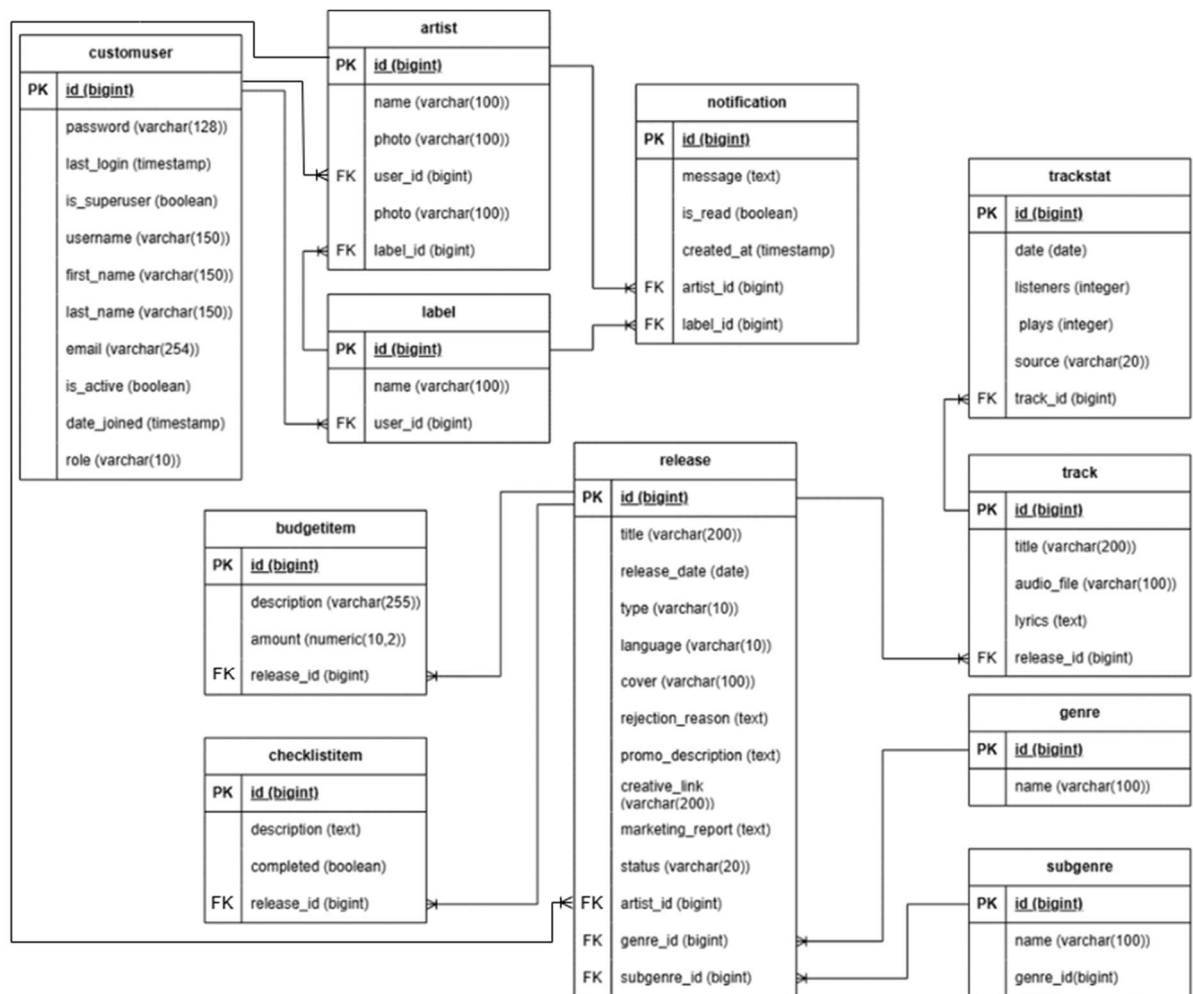
СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Сторінка тарифів LabelGrid [Електронний ресурс]. – Режим доступу: <https://labelgrid.com/pricing/> – Дата доступу: березень 2025.
2. Домашня сторінка LabelGrid [Електронний ресурс]. – Режим доступу: <https://labelgrid.com> – Дата доступу: березень 2025.
3. Документація LabelGrid [Електронний ресурс]. – Режим доступу: <https://labelgrid.atlassian.net/wiki/spaces/CSM/pages/5505025/Create+a+release> – Дата доступу: березень 2025.
4. Інструменти маркетингу та просування LabelGrid [Електронний ресурс]. – Режим доступу: <https://labelgrid.com/solutions/music-promotion-marketing-tools/> – Дата доступу: березень 2025.
5. Документація LabelGrid Demo Inbox [Електронний ресурс]. – Режим доступу: <https://labelgrid.atlassian.net/wiki/spaces/CSM/pages/16154666/Demo+Inbox#Configuring-Presets> – Дата доступу: березень 2025.
6. Домашня сторінка Reptoir [Електронний ресурс]. – Режим доступу: <https://www.reprtoir.com/> – Дата доступу: березень 2025.
7. Сторінка Release Builder Reptoir [Електронний ресурс]. – Режим доступу: <https://www.reprtoir.com/release-builder> – Дата доступу: березень 2025.
8. Сторінка Royalty Accounting Reptoir [Електронний ресурс]. – Режим доступу: <https://www.reprtoir.com/royalty-accounting> – Дата доступу: березень 2025.
9. Сторінка Reptoir Audio AI [Електронний ресурс]. – Режим доступу: <https://www.reprtoir.com/audio-ai> – Дата доступу: березень 2025.
10. Сторінка Python Official Website [Електронний ресурс]. – Режим доступу: <https://www.python.org/> – Дата доступу: квітень 2025.
11. Сторінка TIOBE Index for April 2025 [Електронний ресурс]. – Режим доступу: <https://www.tiobe.com/tiobe-index/> – Дата доступу: квітень 2025.

12. Сторінка Django documentation – The web framework for perfectionists with deadlines [Електронний ресурс]. – Режим доступу:<https://docs.djangoproject.com/en/5.2/> – Дата доступу: квітень 2025.
13. Сторінка 10 Popular Django Websites [Електронний ресурс]. – Режим доступу:<https://djangostars.com/blog/10-popular-sites-made-on-django/> – Дата доступу: квітень 2025.
14. Сторінка PostgreSQL Global Development Group [Електронний ресурс]. – Режим доступу:<https://www.postgresql.org/> – Дата доступу: квітень 2025.
15. Сторінка (MDN). JavaScript Guide [Електронний ресурс]. – Режим доступу:<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide> – Дата доступу: квітень 2025.

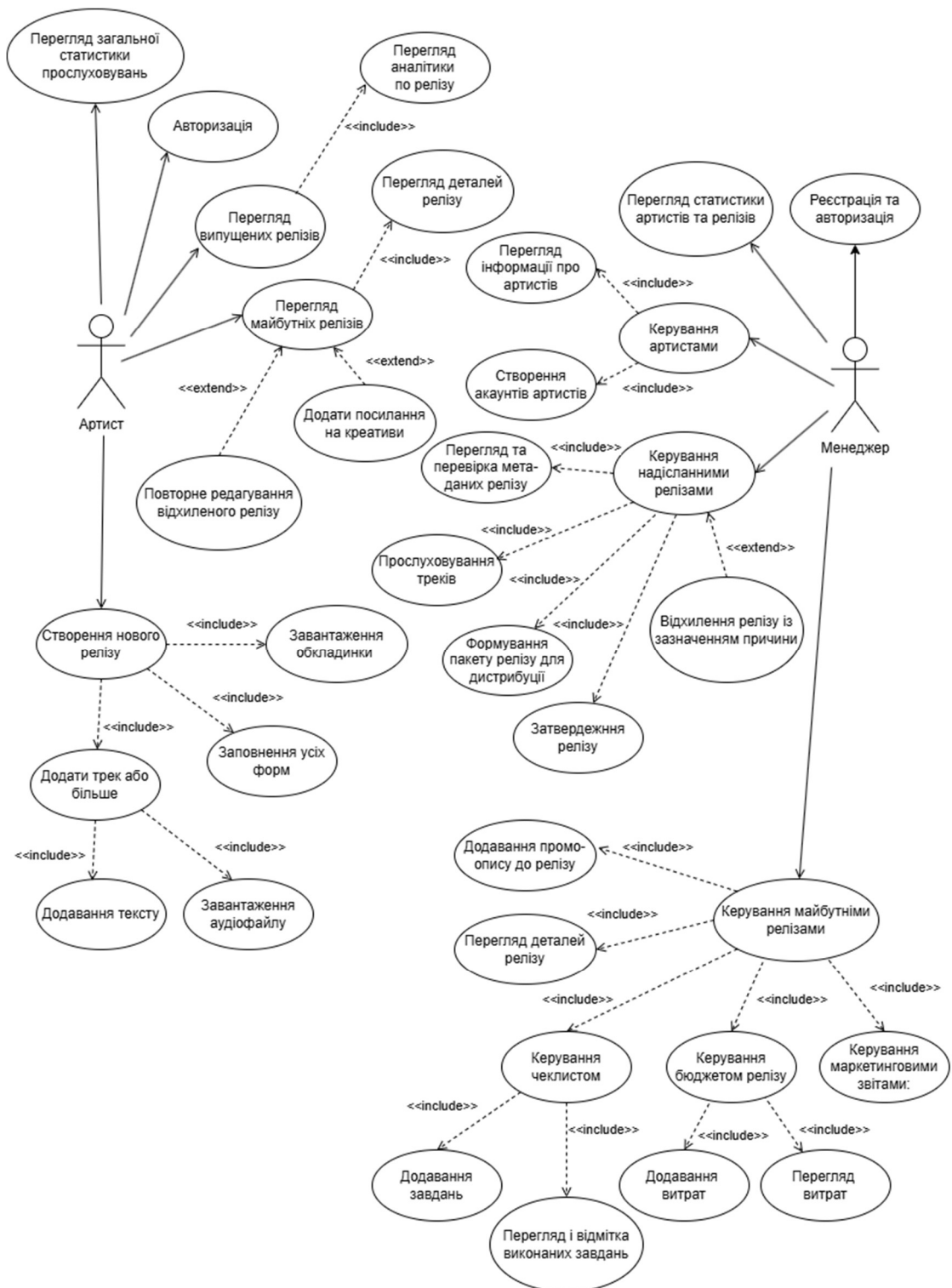
ДОДАТКИ

Додаток 1
Копії графічних матеріалів



ДП.045440-06-99

Програмна система управління
випуском музики для незалежних
лейблів. Схема бази даних системи.
ER-діаграма



ДП.045440-07-99

Програмна система управління випуском музики для незалежних лейблів. Функціональність вебзастосунку. UML-діаграма прецендентів

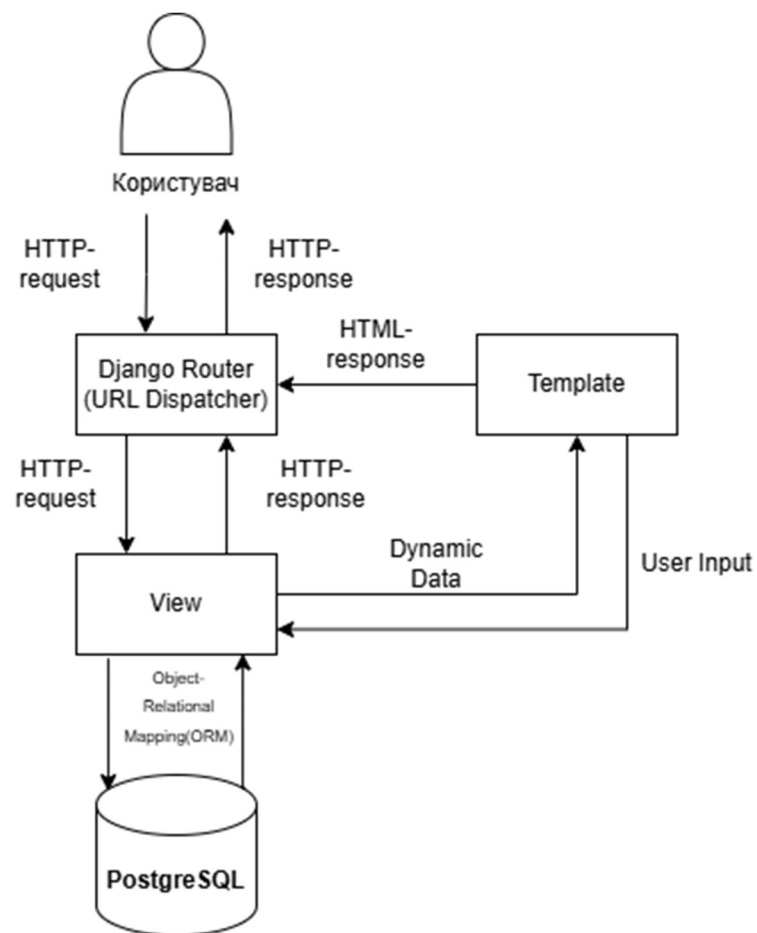
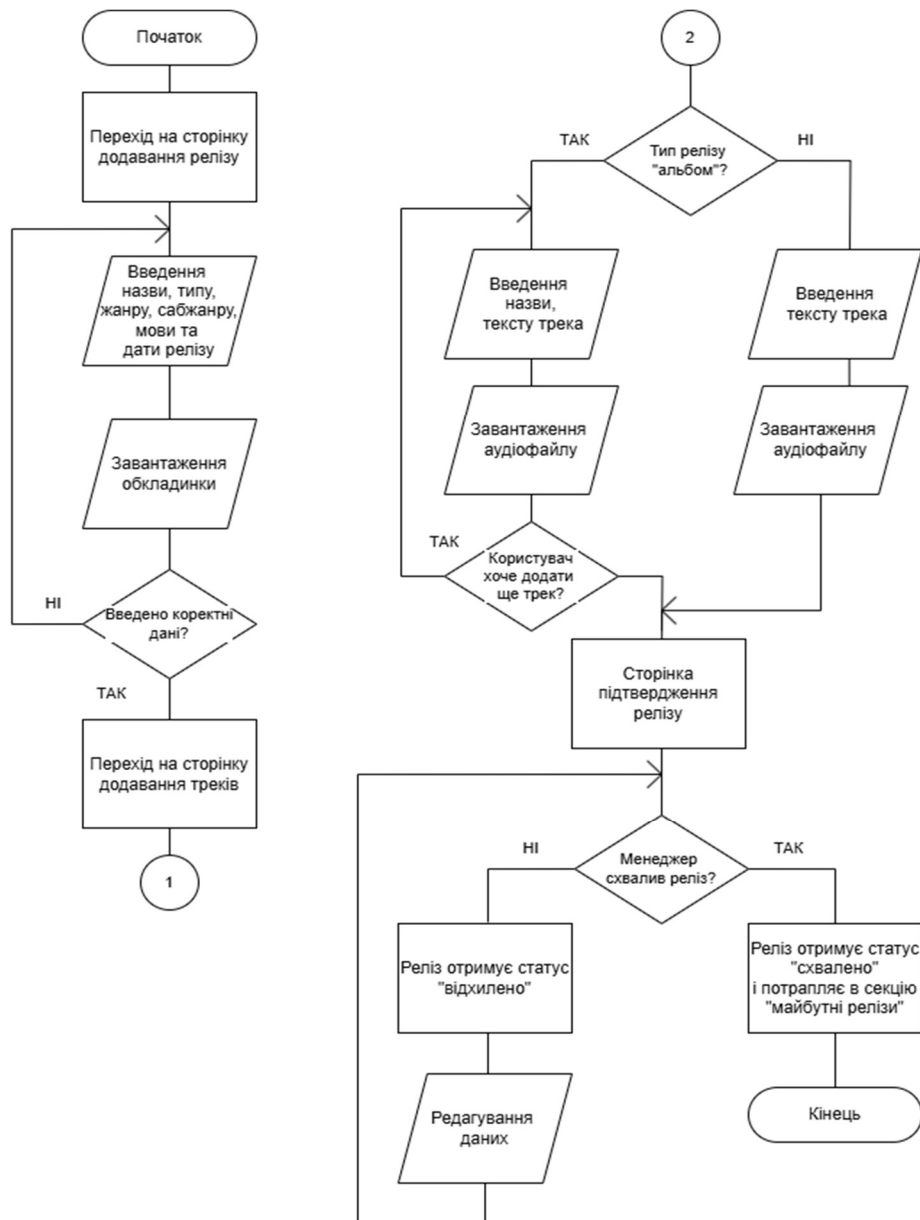


Схема оброблення
HTTP запитів
Чердаклієв К. Є., група КП-12



Алгоритм додавання релізу артистом
Чердаклієв К. Є., група КП-12

Додаток 2
Лістинг програми

Фрагмент коду маршрутизації у вебзастосунку

```
from django.contrib import admin

from django.urls import path

from core.views import add_artist, add_budget_item, add_checklist_item,
artist_login_view, artist_dashboard_view, artist_logout_view,
artist_statistics_view, create_release_view, delete_budget_item,
download_release_metadata, edit_rejected_release, future_release_detail,
register_label_view, label_dashboard, home_view, release_detail_view,
review_release_view, get_notifications, update_marketing_report

from django.conf import settings

from django.contrib.auth import views as auth_views

from django.conf.urls.static import static

urlpatterns = [

    path('admin/', admin.site.urls),

    path('', home_view, name='home'),

    path('artist/login/', artist_login_view, name='artist_login'),

    path('artist/dashboard/', artist_dashboard_view,
name='artist_dashboard'),

    path('artist/logout/', artist_logout_view, name='artist_logout'),

    path('artist/add-release/', create_release_view,
name='create_release'),

    path('register/label/', register_label_view, name='register_label'),

    path('dashboard/label/', label_dashboard, name='label_dashboard'),

    path('label/add-artist/', add_artist, name='add_artist'),

    path('label/moderate-release/<int:release_id>', review_release_view,
name='review_release'),

    path('label/download-release/<int:release_id>',
download_release_metadata, name='download_release_metadata'),

    path('artist/edit-release/<int:release_id>', edit_rejected_release,
name='edit_rejected_release'),

    path('artist/notifications/', get_notifications,
name='get_notifications'),

    path('release/<int:release_id>/future/', future_release_detail,
name='future_release_detail'),

    path('release/<int:release_id>/checklist/add/', add_checklist_item,
name='add_checklist_item'),

    path('release/<int:release_id>/add-budget-item/', add_budget_item,
name='add_budget_item'),

    path('budget-item/<int:item_id>/delete/', delete_budget_item,
name='delete_budget_item'),

    path('artist/<int:artist_id>/stats/', artist_statistics_view,
name='artist_statistics'),
```

```
    path('release/<int:release_id>/', release_detail_view,
name='release_detail'),

    path('release/<int:release_id>/marketing-report/',
update_marketing_report, name='update_marketing_report'),

    path(
        'accounts/login/',
        auth_views.LoginView.as_view(
            template_name='registration/login.html',
        ),
        name='login'
    ),

    path('accounts/logout/', auth_views.LogoutView.as_view(),
name='logout'),
]

if settings.DEBUG:

    urlpatterns += static(settings.MEDIA_URL,
document_root=settings.MEDIA_ROOT)
```

Фрагмент коду представлення створення релізу

```
@login_required
def create_release_view(request):
    if request.method == 'POST':
        release_form = ReleaseForm(request.POST, request.FILES)
        track_form = SingleTrackForm(request.POST, request.FILES)
        track_formset = TrackFormSet(request.POST, request.FILES)

        if release_form.is_valid():
            release_type = release_form.cleaned_data['type']

            if release_type == 'single' and track_form.is_valid():
                release = release_form.save(commit=False)
                release.artist = request.user.artist
                release.status = 'pending_review'
                release.save()

                track = track_form.save(commit=False)
                track.release = release
                track.title = release.title
                track.save()
                track_form.save_m2m()

                return redirect('artist_dashboard')

            elif release_type == 'album' and track_formset.is_valid():
                release = release_form.save(commit=False)
                release.artist = request.user.artist
                release.status = 'pending_review'
                release.save()

                tracks = track_formset.save(commit=False)

                for i, track in enumerate(tracks):
                    track.release = release
                    track.save()
                    form = track_formset.forms[i]
```

```
        form.save_m2m()

        return redirect('artist_dashboard')

    else:

        release_form = ReleaseForm()

        track_form = SingleTrackForm()

        track_formset = TrackFormSet(queryset=Track.objects.none(),
prefix='form')

    return render(request, 'add_release.html', {

        'release_form': release_form,

        'track_form': track_form,

        'track_formset': track_formset,

    })
```

Фрагмент коду динамічного додавання треків до релізу

```
document.addEventListener("DOMContentLoaded", function () {

    let formChanged = false;

    document.querySelectorAll("input, select, textarea").forEach((el) => {
        el.addEventListener("input", () => {
            formChanged = true;
        });
    });

    window.addEventListener("beforeunload", function (e) {
        if (formChanged) {
            e.preventDefault();
            e.returnValue = "";
        }
    });

    window.goToStep = function (step) {
        document.querySelectorAll(".step-indicator").forEach((el, i) => {
            el.classList.toggle("active", i + 1 === step);
        });

        document.querySelectorAll(".step").forEach((s) =>
s.classList.remove("active"));

        document.getElementById("step" + step).classList.add("active");

        if (step === 2) {
            document.getElementById("releaseTitleDisplay").innerText =
document.getElementById("id_title").value;
        }

        if (step === 3) {
            document.getElementById("confirmTitle").innerText =
document.getElementById("id_title").value;

            const genre = document.getElementById("id_genre");

            document.getElementById("confirmGenre").innerText =
genre.options[genre.selectedIndex]?.text;

            document.getElementById("confirmDate").innerText =
document.getElementById("id_release_date").value;

            const lang = document.getElementById("id_language");
```

```

        document.getElementById("confirmLang").innerText =
lang.options[lang.selectedIndex]?.text;

        const type = document.getElementById("id_type").value;

        const trackBlocks = document.querySelectorAll(".track-block");

        document.getElementById("confirmType").innerText =
            type === "single" ? "Сингл" : trackBlocks.length < 7 ? "Мініальбом"
: "Альбом";

        const coverInput = document.getElementById("id_cover");
        if (coverInput.files.length > 0) {
            const reader = new FileReader();
            reader.onload = (e) => {
                const img = document.getElementById("confirmCover");
                img.src = e.target.result;
                img.style.display = "block";
            };
            reader.readAsDataURL(coverInput.files[0]);
        }

        renderTrackPreview();
    }
};

window.onReleaseNext = function () {
    const requiredFields = ["id_title", "id_genre", "id_release_date",
"id_language", "id_type", "id_cover"];
    let isValid = true;
    requiredFields.forEach((id) => {
        const input = document.getElementById(id);
        if (!input || !input.value) {
            input.classList.add("input-error");
            isValid = false;
        } else {
            input.classList.remove("input-error");
        }
    });
});

if (!isValid) {
    alert("Будь ласка, заповніть усі обов'язкові поля!");
}

```

```

        return;
    }

    const type = document.getElementById("id_type").value;
    goToStep(2);

    document.getElementById("single-block").style.display = type ===
    "single" ? "block" : "none";

    document.getElementById("album-block").style.display = type ===
    "single" ? "none" : "block";
};

window.addTrack = function () {
    const container = document.getElementById("track-container");
    const totalFormsInput = document.getElementById("id_form-TOTAL_FORMS");
    const template = document.getElementById("empty-form-
    template").innerHTML;
    const trackIndex = parseInt(totalFormsInput.value);

    // Замінюємо __prefix__ на індекс
    const html = template.replace(/__prefix__/g, trackIndex);
    const temp = document.createElement("div");
    temp.innerHTML = html;

    const trackBlock = temp.firstElementChild;
    trackBlock.classList.add("track-block");
    trackBlock.setAttribute("data-track-index", trackIndex);

    // Додаємо кнопку ✕
    const removeBtn = document.createElement("button");
    removeBtn.type = "button";
    removeBtn.className = "remove-track";
    removeBtn.textContent = "✕";

    removeBtn.addEventListener("click", () => {
        trackBlock.remove();

        // оновлюємо TOTAL_FORMS після видалення
        totalFormsInput.value = document.querySelectorAll(".track-
        block").length;
    });
};

```

```

trackBlock.appendChild(removeBtn);
container.appendChild(trackBlock);

// Оновлюємо TOTAL_FORMS після додавання
totalFormsInput.value = trackIndex + 1;
};

function renderTrackPreview() {
    const container = document.getElementById('confirmTracks');
    container.innerHTML = "";

    const type = document.getElementById('id_type').value;

    // === Для синглу ===
    if (type === 'single') {
        const title = document.getElementById('id_title').value.trim();
        const lyrics = document.getElementById('id_lyrics').value.trim();
        const audioInput = document.getElementById('id_audio_file');
        const audioFile = audioInput?.files?.[0];

        const div = document.createElement("div");
        div.classList.add("track-preview");
        div.style.marginBottom = "20px";

        div.innerHTML = `
            <p><strong>${title || "Без назви"}</strong></p>
            ${audioFile ? `
                <audio controls style="width: 100%; margin-bottom: 5px;">
                    <source src="${URL.createObjectURL(audioFile)}"
type="audio/mpeg">
                Ваш браузер не підтримує аудіо.
            </audio>
            ` : '<p><i>Аудіо не вибрано</i></p>' }

            <button type="button" onclick="toggleLyrics(0)">Показати/сховати
текст</button>

            <div id="lyrics-0" style="display: none; margin-top: 10px;">

```

```

        <pre>${lyrics || 'Немає текстів'}</pre>
    </div>
    `;

    container.appendChild(div);
    return;
}

// === Для альбому / EP ===
const trackBlocks = document.querySelectorAll('.track-block');
if (trackBlocks.length === 0) {
    container.innerHTML = "<p><i>Треки не знайдено</i></p>";
    return;
}

trackBlocks.forEach((block, index) => {
    const titleInput = block.querySelector('input[name$="title"]');
    const lyricsInput = block.querySelector('textarea[name$="lyrics"]');
    const audioInput = block.querySelector('input[name$="audio_file"]');

    if (!titleInput || !lyricsInput) return;

    const title = titleInput.value.trim();
    const lyrics = lyricsInput.value.trim();
    const audioFile = audioInput?.files?.[0];

    const div = document.createElement("div");
    div.classList.add("track-preview");
    div.style.marginBottom = "20px";

    div.innerHTML = `
        <p><strong>${title || "Без назви"}</strong></p>
        ${audioFile ? `
            <audio controls style="width: 100%; margin-bottom: 5px;">
                <source src="${URL.createObjectURL(audioFile)}"
type="audio/mpeg">
                Ваш браузер не підтримує аудіо.
            </audio>
        ` : ''}
    `;

```

```

        ` : '<p><i>Аудіо не вибрано</i></p>'{
        <button type="button"
onclick="toggleLyrics(${index})">Показати/сховати текст</button>

        <div id="lyrics-${index}" style="display: none; margin-top: 10px;">
            <pre>${lyrics || 'Немає тексту'}</pre>
        </div>
        `;

        container.appendChild(div);
    });
}

window.toggleLyrics = function (index) {
    const block = document.getElementById(`lyrics-${index}`);
    if (block) {
        block.style.display = block.style.display === "none" ? "block" :
"none";
    }
};
});

```

Фрагмент коду форми для додавання треків

```
class TrackForm(forms.ModelForm):

    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.fields['audio_file'].required = False

    class Meta:
        model = Track
        fields = ['title', 'audio_file', 'lyrics']
        widgets = {
            'lyrics': forms.Textarea(attrs={
                'rows': 4,
                'cols': 40,
                'placeholder': 'Введіть текст пісні...',
                'style': 'resize: vertical;'
            })
        }
```

Додаток 3
Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ
СІКОРСЬКОГО”



ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРНИХ СИСТЕМ

ПРОГРАМНА СИСТЕМА УПРАВЛІННЯ ВИПУСКОМ МУЗИКИ ДЛЯ НЕЗАЛЕЖНИХ ЛЕЙБЛІВ

Виконав: студент групи КП-12 Чердаклієв Кирило Євгенович

Керівник: асистент кафедри ПЗКС, д-р філософії, Юсин Яків Олексійович

Київ – 2025



ПОСТАНОВКА ЗАДАЧІ

Мета проекту: розробити програмну систему для управління музичними релізами незалежних лейблів, яка забезпечить структурований та ефективний процес створення, модерації та підготовки музичних релізів до публікації

Завдання:

1. Проаналізувати наявні рішення ринку дистрибуції та музичного менеджменту
2. Обрати засоби реалізації
3. Розробити архітектуру системи та реалізувати MVP застосунку
4. Протестувати вебзастосунок відповідно до технічного завдання
5. Визначити перспективи розвитку та розширення системи



АКТУАЛЬНІСТЬ

- 1.Зростання незалежного музичного продюсування
- 2.Відсутність комплексних рішень
- 3.Потреба в автоматизації рутинних процесів
- 4.Необхідність централізації роботи з релізом



ІСНУЮЧІ АНАЛОГИ



REPRTOIR

ПОРІВНЯННЯ АНАЛОГІВ

Критерій	Аналоги	
	LabelGrid	Reprtoir
Управління релізами	+	+
Аналітика за прослуховуваннями	-	+
Планування маркетингових активностей	-	+
Доступ артиста до кабінету	-	+
Можливість дистрибуції	+	-
Автоматизація документообігу	-	+

ФУНКЦІОНАЛЬНІ ВИМОГИ ДО ВЕБЗАСТОСУНКУ



- 1) можливість авторизації для всіх ролей (менеджер, артист);
- 2) можливість створення артистів менеджером лейблу;
- 3) можливість створення релізів артистами із заповненням усіх метаданих;
- 4) можливість перевірки релізу менеджером: перегляд, прослуховування, затвердження або відхилення;
- 5) перегляд статистики релізів у вигляді графіків і діаграм;
- 6) можливість редагування відхиленого релізу та повторного подання на перевірку;
- 7) можливість додавати інформацію до майбутнього релізу.



ЗАСОБИ РЕАЛІЗАЦІЇ

Серверна частина: **Python**, Фреймворк:

Django 5

Клієнтська частина: **HTML5**, **CSS3**,

мова програмування **JavaScript**,

технологія **AJAX** для динамічної взаємодії

сервером

База даних: **PostgreSQL**

Додаткові засоби: **Chart.js**



HTML



CSS



АРХІТЕКТУРА ВЕБЗАСТОСУНКУ

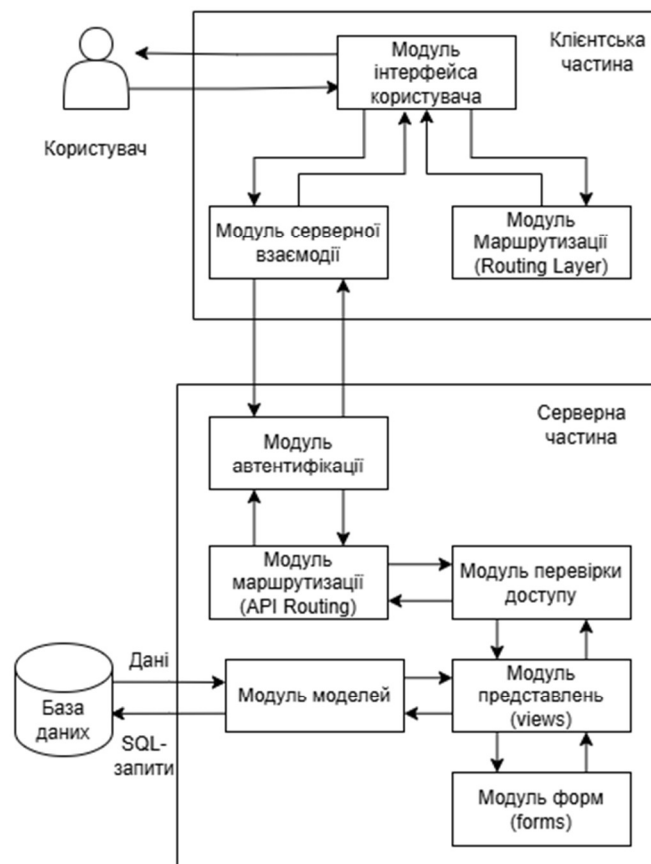


СХЕМА ОБРОБЛЕННЯ HTTP-ЗАПИТІВ

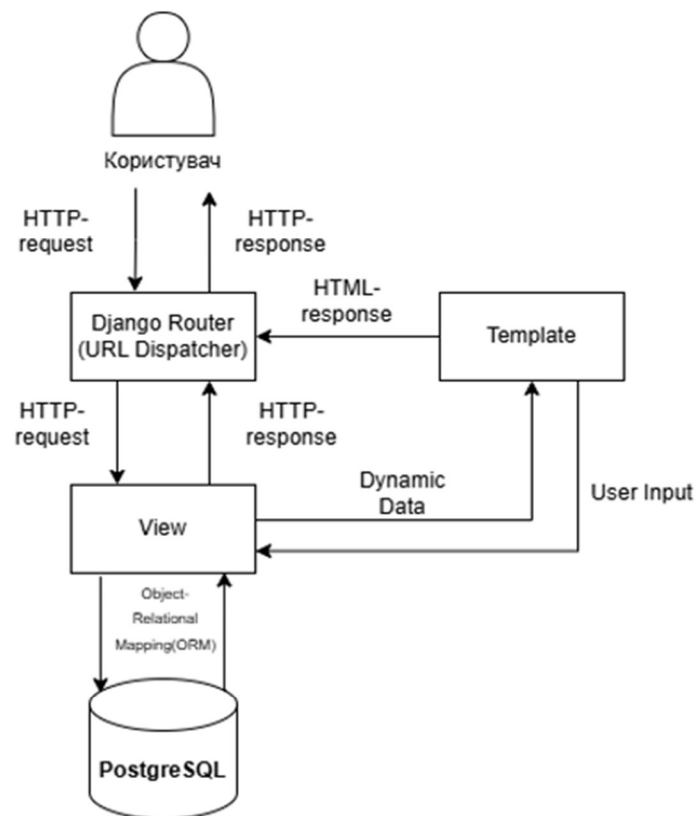
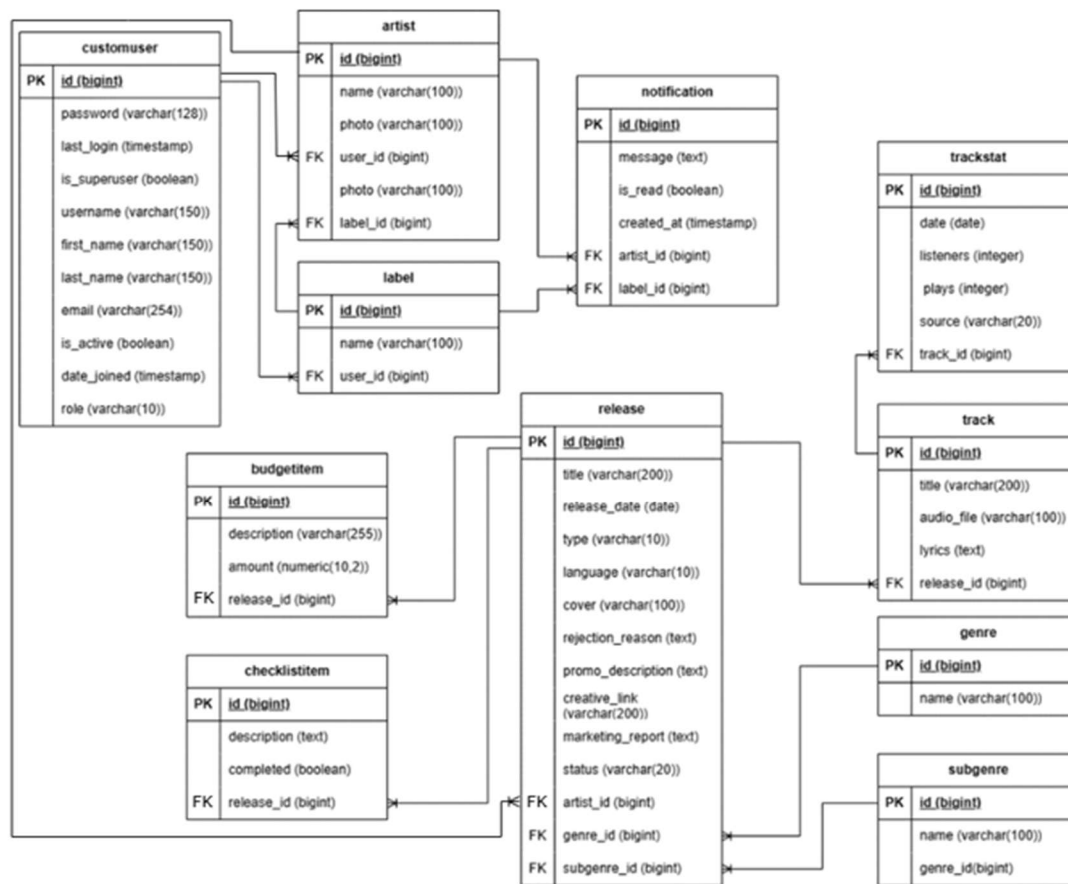
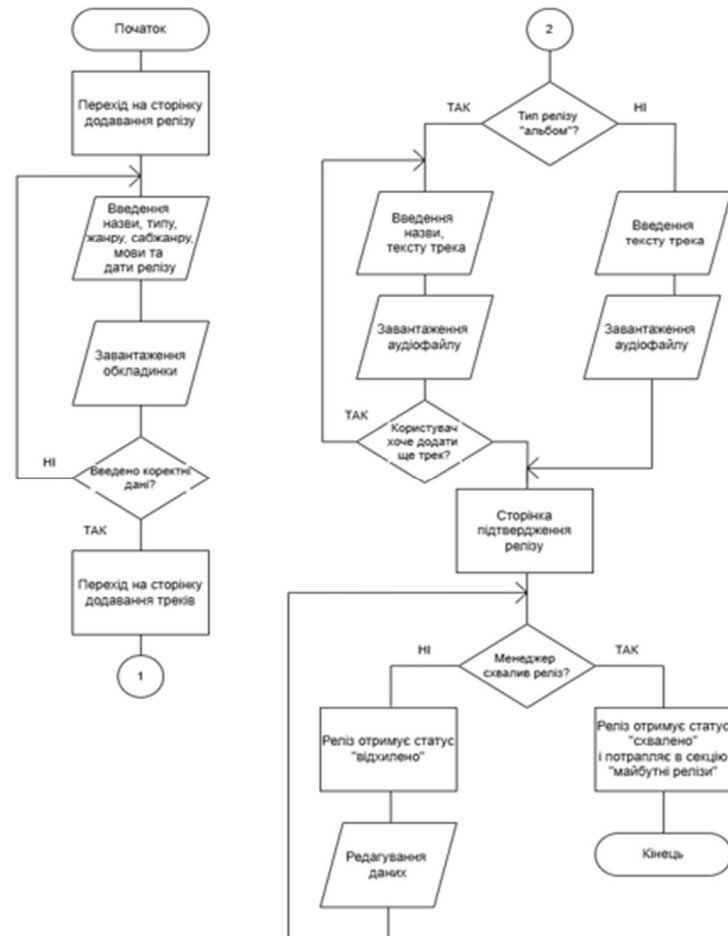
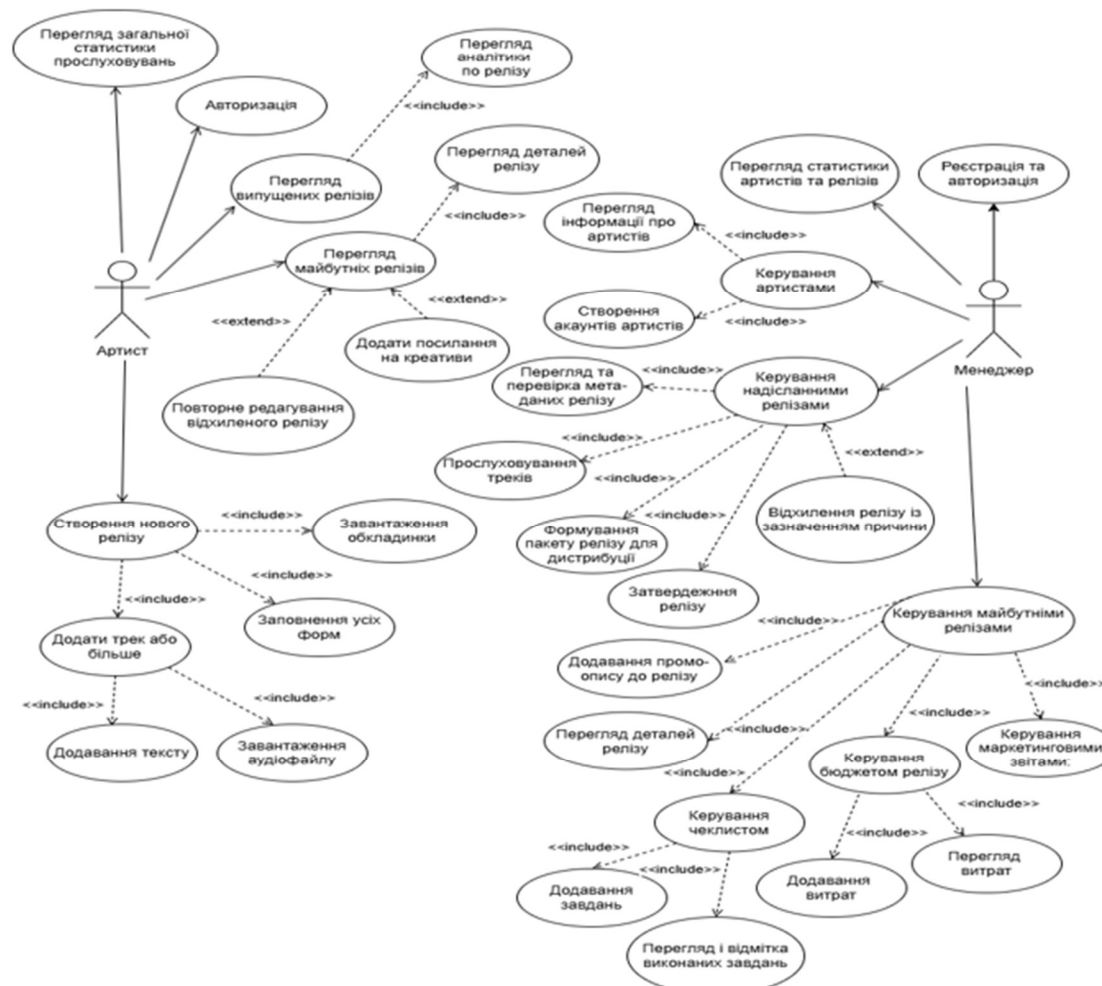


СХЕМА БАЗИ ДАНИХ



АЛГОРИТМ ДОДАВАННЯ РЕЛІЗУ АРТИСТОМ





ТЕСТУВАННЯ ВЕБЗАСТОСУНКУ



- Ручне динамічне тестування інтерфейсу користувача
- Ручне тестування відповідності функціональним вимогам

НАПРЯМИ ПОДАЛЬШОГО ВДОСКОНАЛЕННЯ



- 1.Інтеграція з платформами цифрової дистрибуції через API для автоматизованої публікації релізів
- 2.Додавання внутрішнього чату між артистом і менеджером
- 3.Генерація контрактів з електронним підписом.
- 4.Створення мобільної версії або PWA для роботи з релізами на смартфоні
- 5.Публічна сторінка артиста з інформацією про релізи та посиланнями
- 6.Підтримка багатомовного інтерфейсу для розширення цільової аудиторії

АПРОБАЦІЯ



Було отримано акт від ТОВ «БЛЕК БІТС» про зацікавленість підприємства у впровадженні отриманих результатів дипломного проєкту.



ВИСНОВКИ



1. Проаналізовано наявні системи управління музичними релізами та їх обмеження
2. Розроблено архітектуру вебзастосунку та спроєктовано реляційну базу даних.
3. Реалізовано серверну та клієнтську частину застосунку з урахуванням розподілу ролей (артист, менеджер).
4. Реалізовано основні функціональні можливості системи: створення, модерація та планування релізів, аналітика, бюджетування.
5. Проведено тестування функцій та перевірку відповідності вимогам технічного завдання.
6. Визначено перспективні напрями подальшого розвитку системи.



Дякую за увагу!

Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Євгенія СУЛЕМА

“ ____ ” _____ 2024 р.

ПРОГРАМНА СИСТЕМА УПРАВЛІННЯ ВИПУСКОМ МУЗИКИ
ДЛЯ НЕЗАЛЕЖНИХ ЛЕЙБЛІВ

Програма та методика тестування

ДП.045440-04-51

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Яків ЮСИН

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Кирило ЧЕРДАКЛІЄВ

ЗМІСТ

1. Об'єкт випробувань	3
2. Мета тестування	3
3. Методи тестування.....	3
4. Засоби та порядок тестування.....	4

1. ОБ'ЄКТ ВИПРОБУВАНЬ

Вебзастосунок ReleaseM Manager, призначений для управління музичними релізами незалежних лейблів. Технічна реалізація виконана з використанням фреймворку Django (Python), системи керування базами даних PostgreSQL, мови розмітки HTML, стилів CSS, мови програмування JavaScript, а також технології AJAX для динамічної взаємодії інтерфейсу з сервером без перезавантаження сторінки.

2. МЕТА ТЕСТУВАННЯ

Метою тестування є перевірка:

- 1) працездатності основних компонентів вебінтерфейсу та взаємодії між ними;
- 2) коректної реалізації маршрутизації у межах ролей «артист» та «менеджер»;
- 3) логіки роботи двоетапного процесу створення релізу;
- 4) доступності та стабільної взаємодії серверної частини з базою даних;
- 5) відповідності клієнтсько-серверної взаємодії очікуваній бізнес-логіці;
- 6) відповідності реалізованих функціональних можливостей вимогам, зазначеним у Технічному завданні;
- 7) коректності обробки форм, файлів, повідомлень та статистичних звітів.

3. МЕТОДИ ТЕСТУВАННЯ

На кожному етапі життєвого циклу розробки застосунку виконувалося інкрементальне ручне тестування нових функціональних

можливостей. Такий підхід дозволив вчасно виявити й усунути помилки, що могли вплинути на працездатність системи. Протягом усього періоду розробки застосовувався емпіричний підхід до тестування, в межах якого розробник виступав у ролі кінцевого користувача. Перед початком тестування визначались функціональні ділянки, що підлягали перевірці, після чого формувався список тестових випадків. Кожен тестовий випадок відтворювався у практиці з фіксацією фактичних результатів і порівнянням їх з очікуваними. У разі виявлення невідповідностей проводилось усунення виявленої помилки з подальшим повторним тестуванням для підтвердження коректності роботи функціоналу.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування працездатності застосунку ReleaseM Manager здійснювалося за такими напрямками: динамічного ручного тестування – введенням граничних та недопустимих значень в поля, які можна редагувати;

- 1) динамічне ручне тестування введенням коректних, граничних та некоректних значень у поля форм;
- 2) перевірка відповідності поведінки елементів інтерфейсу функціональним вимогам;
- 3) статичний аналіз коду;
- 4) тестування застосунку в різних браузерях;
- 5) перевірка стабільності роботи при різних сценаріях взаємодії;
- 6) тестування зручності та інтуїтивності інтерфейсу для двох основних ролей – артист та менеджер;
- 7) перевірка динамічних елементів інтерфейсу.

Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Євгенія СУЛЕМА

“ ____ ” _____ 2025 р.

ПРОГРАМНА СИСТЕМА УПРАВЛІННЯ ВИПУСКОМ МУЗИКИ
ДЛЯ НЕЗАЛЕЖНИХ ЛЕЙБЛІВ

Керівництво користувача

ДП.045440-05-34

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Яків ЮСИН

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Кирило ЧЕРДАКЛІЄВ

ЗМІСТ

1. Опис структури вебзастосунку	3
2. Інтерфейс для менеджера лейблу	4
3. Інтерфейс для артиста.....	12

1. Опис структури вебзастосунку

Вебзастосунок «ReleaseM Manager» призначений для централізованого управління музичними релізами незалежних лейблів. Система має розподілену рольову модель доступу, що передбачає окремі інтерфейси та сценарії для двох типів користувачів – артиста та менеджера лейблу. Кожен з користувачів після авторизації потрапляє до власного кабінету, у межах якого реалізовано відповідний набір функцій.

Вебзастосунок складається із таких сторінок:

- 1) головна сторінка;
- 2) сторінка реєстрації для лейблу;
- 3) сторінка авторизації;
- 4) особистий кабінет артиста;
- 5) сторінка завантажування релізу;
- 6) сторінка майбутнього релізу;
- 7) сторінка випущеного релізу;
- 8) особистий кабінет менеджера;
- 9) сторінка додавання артиста;
- 10) сторінка модерації артиста;
- 11) сторінка статистики артиста;
- 12) сторінка статистики будь-якого релізу артиста.

Кожна сторінка застосунку відображається динамічно залежно від ролі, та статусу релізу. Структура побудована таким чином, щоб забезпечити логічну послідовність дій – від створення релізу до його випуску та аналізу результатів.

2. Інтерфейс для менеджера лейблу

Щоб почати роботу в системі «ReleaseM Manager», користувач обирає свою роль на головній сторінці (рис. 1). У випадку, якщо користувач є менеджером лейблу, він має натиснути кнопку «Зареєструватись» для

створення нового облікового запису або «Увійти» для авторизації, якщо обліковий запис уже існує.

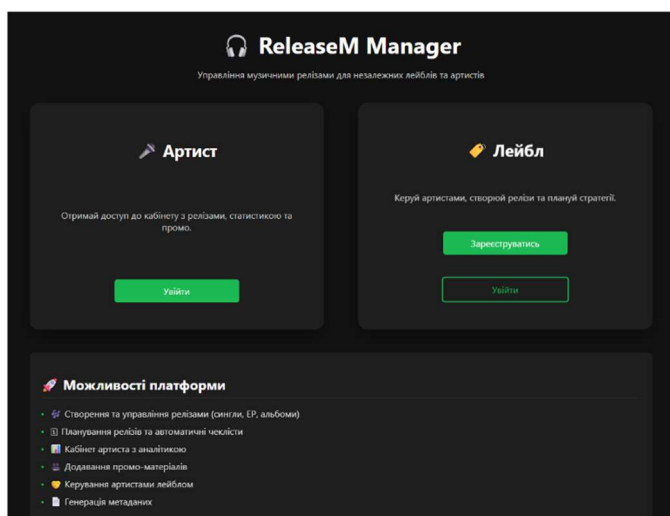


Рис. 1. Головна сторінка

У разі відсутності облікового запису менеджера необхідно зареєструватися на сторінці реєстрації лейблу (рис. 2). На цій сторінці передбачено заповнення обов'язкових полів: електронна пошта, пароль, підтвердження пароля та назва лейблу (яка надалі використовується як унікальний ідентифікатор/логін). Після успішного заповнення форми слід натиснути кнопку «Зареєструватись». Система автоматично перенаправляє користувача до сторінки авторизації.

Рис. 2. Сторінка реєстрації

Після створення облікового запису або за його наявності менеджер може авторизуватись у системі, використовуючи назву лейблу та пароль (рис. 3). Після введення коректних даних необхідно натиснути кнопку «Увійти», після чого користувач потрапляє до персонального кабінету.

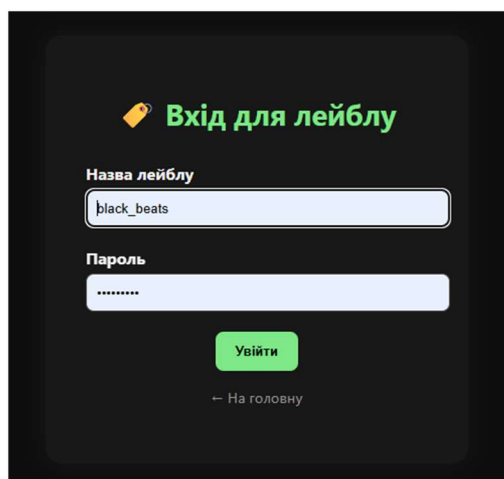


Рис. 3. Сторінка авторизації

Після авторизації в системі менеджер потрапляє до особистого кабінету (рис. 4). Інтерфейс кабінету розділено на дві основні секції: панель релізів та бічну панель артистів. У верхній частині кабінету відображається назва лейблу, під яким авторизовано користувача, а також кнопка «Вийти» для завершення сеансу роботи.

У секції «Релізи на модерації» виводиться перелік усіх музичних релізів, які були завантажені артистами лейблу та наразі перебувають у статусі «Очікує перевірки». Менеджер може перейти до перегляду релізу для його затвердження або відхилення з коментарем.

Нижче розташований блок «Майбутні релізи», який містить список релізів, уже затверджених для публікації, але дата їх виходу ще не настала.

Панель артистів лейблу – список виконавців, закріплених за поточним менеджером. Для кожного артиста виводиться ім'я та аватар. У нижній частині цієї секції знаходиться кнопка «Додати артиста», яка відкриває форму створення нового артиста

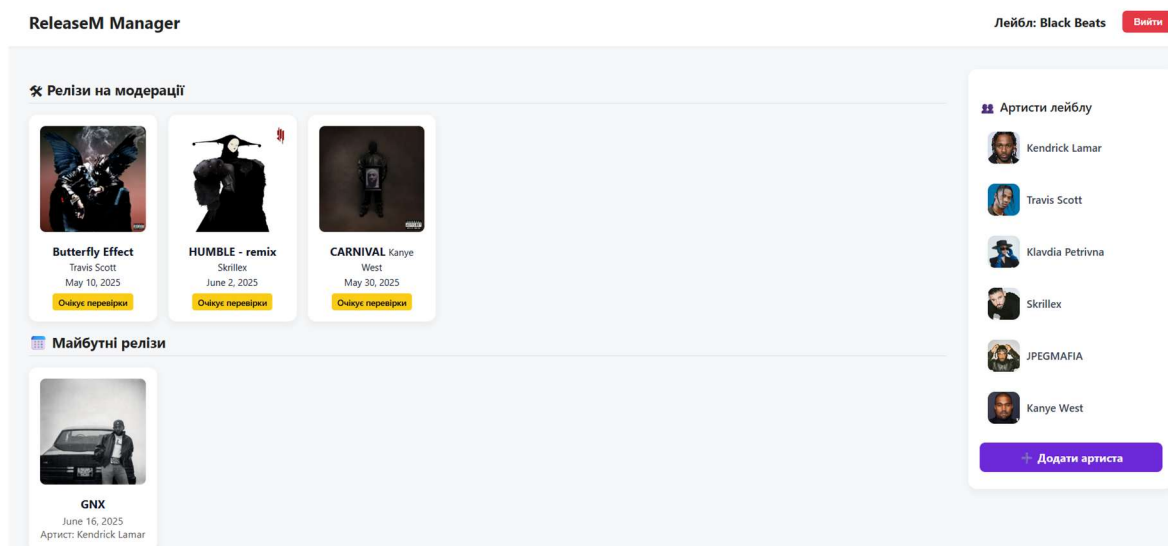


Рис. 4. Особистий кабінет менеджера

На сторінці додавання артиста (рис. 5) необхідно заповнити кілька полів:

- Name – сценічне ім'я артиста, яке відобразиться у всіх релізах.
- Фото артиста – опціональне зображення, яке додається як аватар виконавця.
- ПІБ – повне ім'я та прізвище виконавця, що використовується лише для внутрішньої ідентифікації.
- Email – адреса електронної пошти, яка буде використана для зв'язку, повідомлень, а також для відправлення згенерованого пароля.

Після заповнення всіх полів менеджер натискає кнопку «Додати», що ініціює створення нового користувача-артиста. Артист зможе авторизуватись у системі за допомогою назви лейблу, свого сценічного імені та згенерованого пароля, який надає менеджер. Усі нові артисти прив'язуються до лейблу, який виконує додавання, тому самостійна реєстрація артистів у системі не передбачена. Після створення акаунта артист отримує доступ до власного кабінету, де зможе переглядати свої релізи, додавати нові, а також отримувати повідомлення від менеджера лейблу.

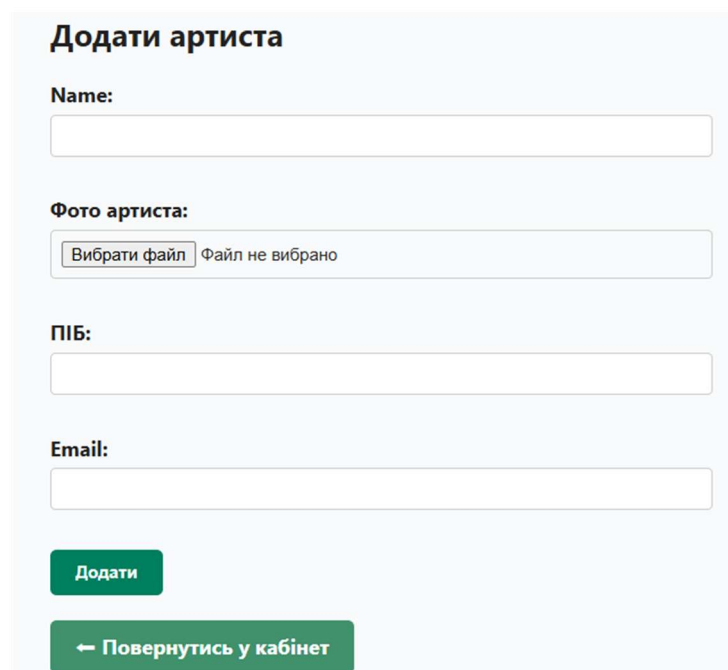
The image shows a web form titled "Додати артиста" (Add artist) in a light blue box. It contains four input fields: "Name:", "Фото артиста:" (Artist photo), "ПІБ:" (Full name), and "Email:". The "Фото артиста:" field has a button "Вибрати файл" (Select file) and the text "Файл не вибрано" (File not selected). At the bottom, there are two green buttons: "Додати" (Add) and "← Повернутись у кабінет" (← Return to cabinet).

Рис. 5. Форма додавання артиста

Після натискання на реліз зі статусом «Очікує перевірки» менеджер лейблу потрапляє на сторінку перевірки релізу (рис. 6). Ця сторінка призначена для ознайомлення з усіма метаданими, а також прослуховування аудіозапису перед прийняттям рішення щодо його схвалення або відхилення.

Нижче подано розділ «Треки», у якому менеджер може прослухати кожен трек релізу безпосередньо в інтерфейсі, а також переглянути повний текст пісні.

У нижній частині сторінки розміщені три основні дії:

- «Схвалити реліз» – змінює статус релізу на підтверджений для публікації.
- «Відхилити реліз» – відкриває форму для зазначення причини відмови.
- «Завантажити мета-дані» – експортує основну інформацію про реліз у форматі ZIP-архіву для зберігання або подальшої дистрибуції.

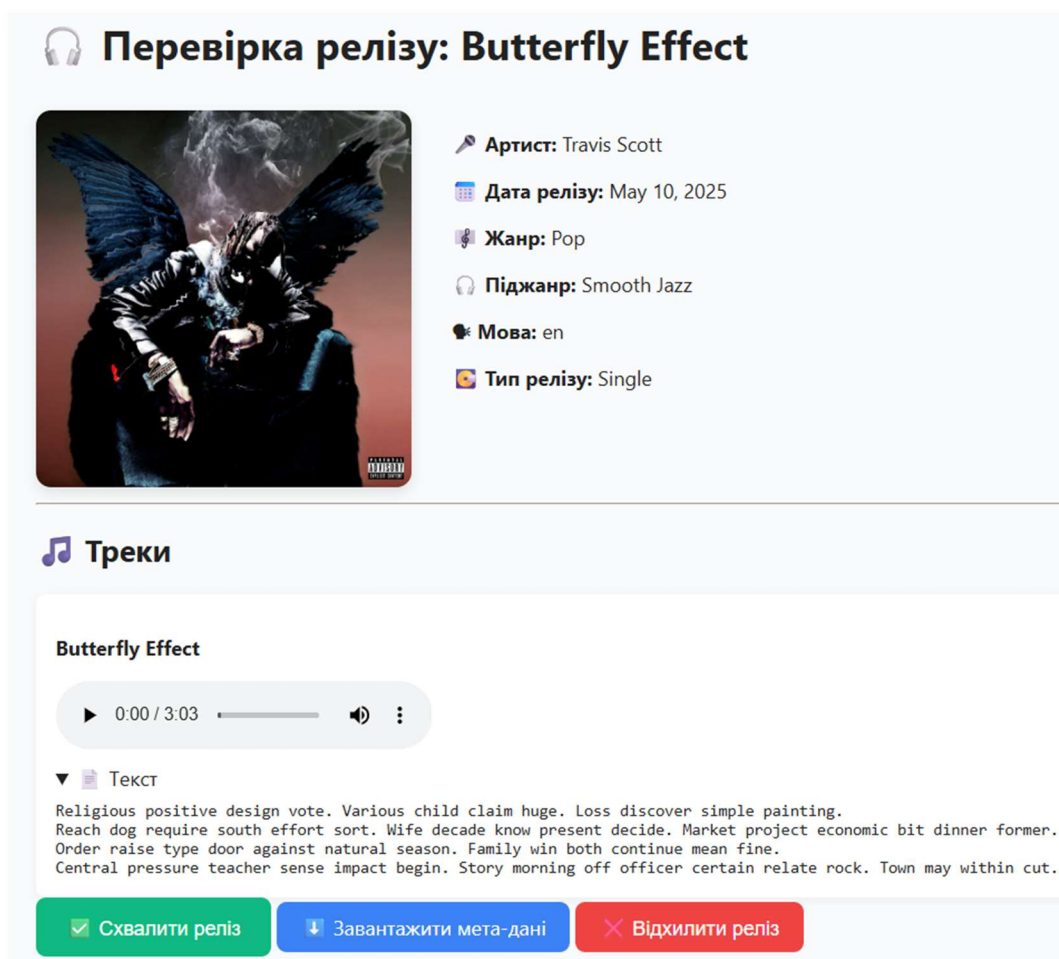


Рис. 6. Сторінка перевірки релізу

Сторінка майбутнього релізу відображає детальну інформацію про реліз, який уже схвалено, але ще не опубліковано (рис. 7).

На сторінці реалізовано блоки для підготовки промокампанії:

- Промо-опис – текстовий блок, у якому менеджер вказує короткий опис релізу, що може бути використаний у маркетингових цілях.
- Бюджет релізу – перелік витрат із зазначенням опису та суми. Доступна функція додавання та видалення позицій, а також автоматичний підрахунок загальної суми.
- Чекліст – список завдань, пов’язаних з промоактивністю (наприклад, зйомка кліпу, фотосесія). Завдання можна додавати вручну та відмічати як виконані.
- Посилання на креативи – поле для вставки посилання на Google Диск чи інший хостинг, де зберігаються візуальні чи медіаматеріали.



GNX

Дата виходу: June 16, 2025

Жанр: Hip-Hop

Артист: Kendrick Lamar

Промо-опис

Перший альбом виконавця, в якому він розкаже про своє життя

Зберегти

Чекліст

☐ кліп 27.09

☐ фотосесія 21.09

Додати завдання

Бюджет релізу

інстаграм сторіс — 200.00\$	✕
тіток реклама — 500.00\$	✕
блогер 1 — 500.00\$	✕
блогер 2 — 1000.00\$	✕

Загальна сума: 2200.00\$

Опис витрати

Сума \$

Додати

Посилання на креативи

Немає посилання

Рис. 7. Сторінка перевірки релізу

Менеджер лейблу має доступ до детальної статистики кожного артиста, з яким він працює. Для цього достатньо обрати артиста зі списку в кабінеті – відкриється персоналізована аналітична сторінка (рис. 8).

У лівій частині відображається базова інформація про артиста. Нижче наведено загальну статистику, яка містить кількість унікальних слухачів та загальну кількість прослуховувань усіх релізів артиста.

Кругова діаграма ілюструє розподіл прослуховувань по релізах, допомагаючи зрозуміти, які з них є найуспішнішими, а лінійний графік відображає динаміку прослуховувань і слухачів за місяцями, що дає змогу відстежувати ріст аудиторії в часі. У нижній частині сторінки виводиться блок опублікованих релізів артиста. Це дозволяє швидко перейти до перегляду кожного з них або до їхньої статистики. Менеджер може використати ці дані для прийняття рішень щодо подальшої промокампанії, бюджету або оновлення стратегії релізів. Аналітична сторінка також дає змогу виявляти тенденції у вподобаннях слухачів та порівнювати результати різних релізів між собою.

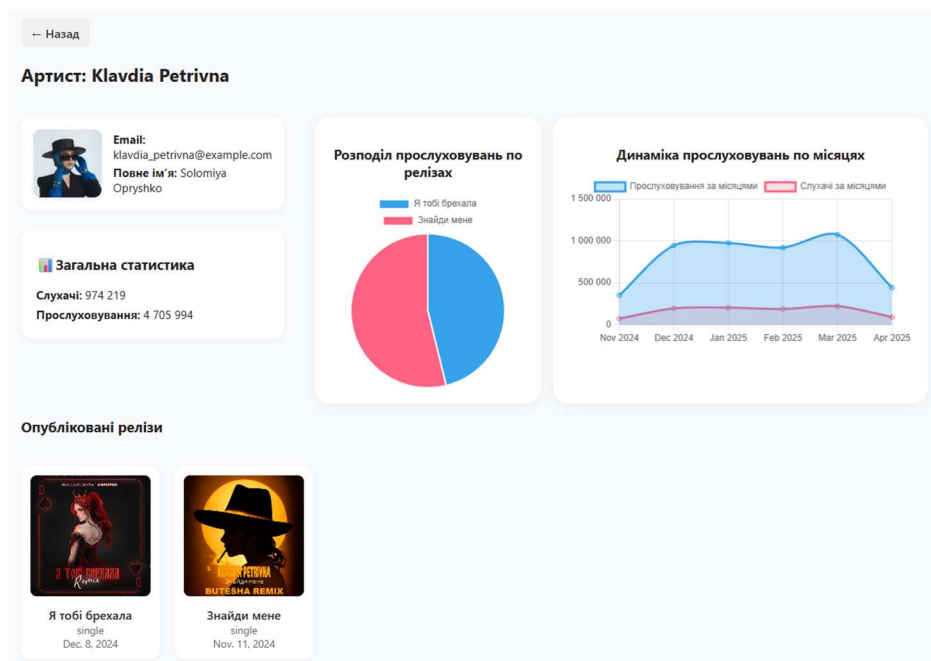


Рис. 8. Сторінка статистики артиста

Менеджер лейблу може переглядати детальну аналітику для кожного конкретного релізу. Сторінка статистики релізу (рис. 9) відображає повний набір даних щодо динаміки прослуховувань, слухачів, платформ і загального охоплення. Також менеджер може додати маркетинговий опис до релізу.

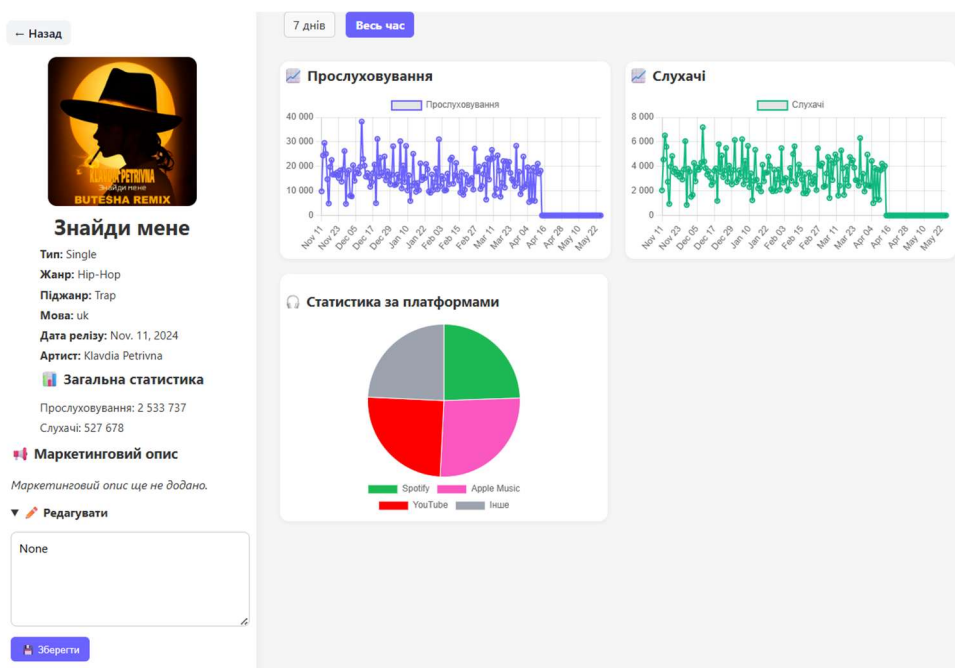


Рис. 9. Сторінка статистики артиста

3. Інтерфейс для артиста

Після авторизації артист потрапляє до свого особистого кабінету – головної сторінки, з якої починається робота із системою (рис. 10). У верхній частині екрану виводиться привітання з іменем користувача та його фотографією профілю. Праворуч розташовані функціональні кнопки:

- «Статистика артиста» – відкриває доступ до аналітичної інформації.
- «Додати новий» – ініціює процес створення нового релізу.
- «Вийти» – вихід із системи.

Нижче представлений перелік релізів артиста, згрупованих за типом:

- Сингли – всі однотрекові релізи, що були опубліковані.
- EP – мініальбоми (у поточному прикладі відсутні).
- Альбоми – повноформатні релізи з 7 треків і більше.
- Майбутні релізи – усі релізи, які були відправлені артистом з будь-яким статусом.

На кожен свій випущений реліз артист може перейти та переглянути статистику. Сторінка статистики по релізу така сама як і для менеджера.

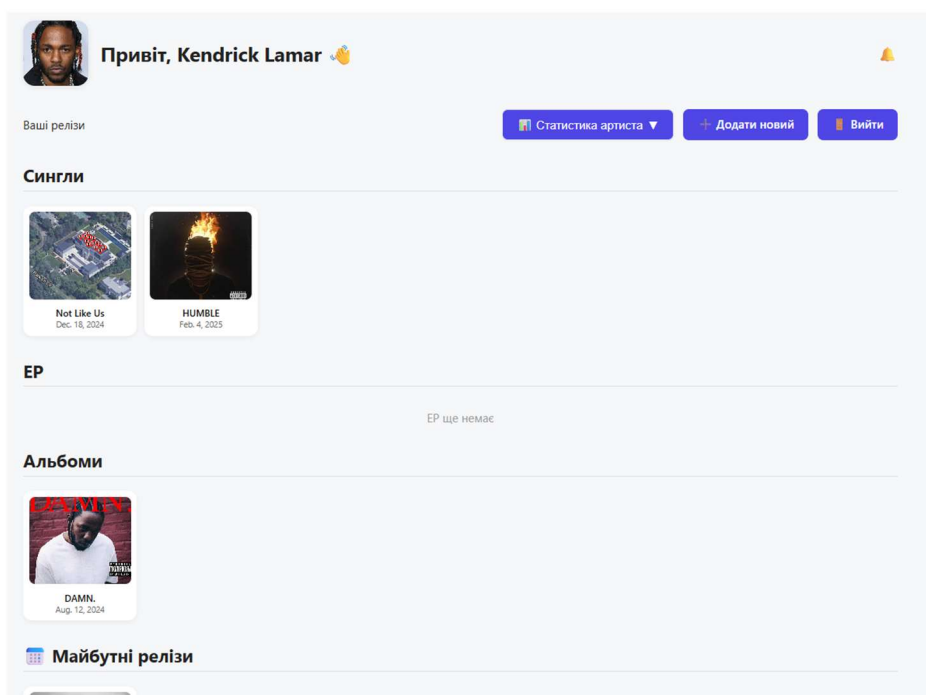


Рис. 10. Особистий кабінет артиста

У розділі «Майбутні релізи» артист може переглянути список релізів, які вже пройшли модерацію, але ще не опубліковані (рис. 11). Цей розділ також включає релізи, що були відхилені менеджером, із вказаною причиною та можливістю повторного редагування.

Таким чином, артист має змогу оперативно реагувати на коментарі менеджера, виправити недоліки та повторно подати реліз на перевірку.

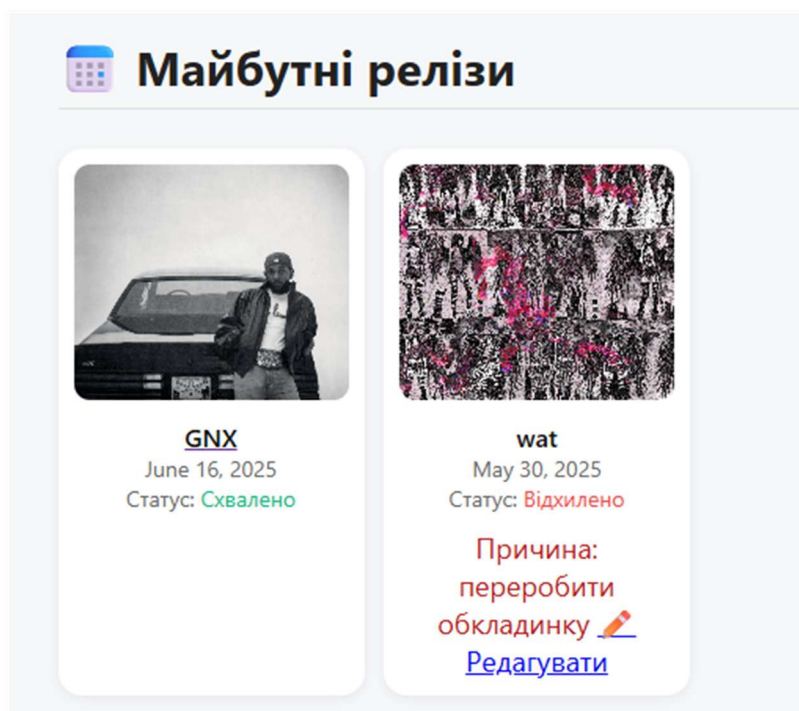


Рис. 11. Розділ майбутні релізи для артиста

У випадку, якщо реліз було відхилено менеджером, артист отримує можливість внести зміни відповідно до наданого коментаря та надіслати реліз повторно на модерацію. При переході до такого релізу відкривається форма редагування (рис. 12), де артист має можливість замінити всі метадані, включаючи аудіофайл та обкладинку. Після внесення змін артист натискає кнопку «Надіслати повторно», що відправляє оновлений реліз менеджеру для повторної перевірки.

Редагування релізу: wat

Ваш реліз було відхилено. Ви можете внести зміни та надіслати його повторно на модерацію.

Причина відхилення: переробити обкладинку

Дані релізу

Title:

Type:

Genre:

Subgenre:

Language:

Release date:

Cover: Currently: [releases/covers/обла в пеклі 3 dUlgUO.png](#)

Change: Файл не вибрано

Трек

Audio file: Currently: [tracks/audio/hyphnic beat 107 C.mp3](#)

Change: Файл не вибрано

Lyrics:

Brace yourself, I'll take you on a trip
down memory lane
This is not a rap on how I'm slingin'
crack or move cocaine

▲
▼

 Надіслати повторно

Скасувати

Рис. 12. Сторінка редагування відхиленого релізу

Після схвалення релізу менеджером, артист отримує доступ до його повної структури в секції «Майбутні релізи» (рис. 13) без можливості редагування блоків, окрім «Посилання на креативи».



GNX

Дата виходу: June 16, 2025

Жанр: Hip-Hop

Промо-опис

Перший альбом виконавця, в якому він розкажує про своє життя

Бюджет релізу

інстаграм сторіс — 200.00\$

тікток реклама — 500.00\$

блогер 1 — 500.00\$

блогер 2 — 1000.00\$

Загальна сума: 2200.00\$

Чекліст

☐

кліп 27.09

☐

фотосесія 21.09

Посилання на креативи

<https://drive.google.com/drive/folders/1B3nerNvgZZCyd6S9KGcqTINMPsl-1M1N>

 Зберегти

Рис. 13. Сторінка майбутнього релізу для артиста

У разі, якщо артист випустив повноформатний альбом, система надає окремий розширений інтерфейс для перегляду аналітики по всіх треках, що входять до його складу (рис. 14). На сторінці розташовані три функціональні елементи аналітики:

- Список треків з короткою статистикою по кожному (прослуховування та слухачі).
- Гістограма прослуховувань треків – дозволяє порівняти популярність усіх композицій в альбомі.
- Детальна статистика обраного треку – лінійний графік прослуховувань і слухачів за часом. Артист може обирати трек зі списку, щоб побачити його динаміку.

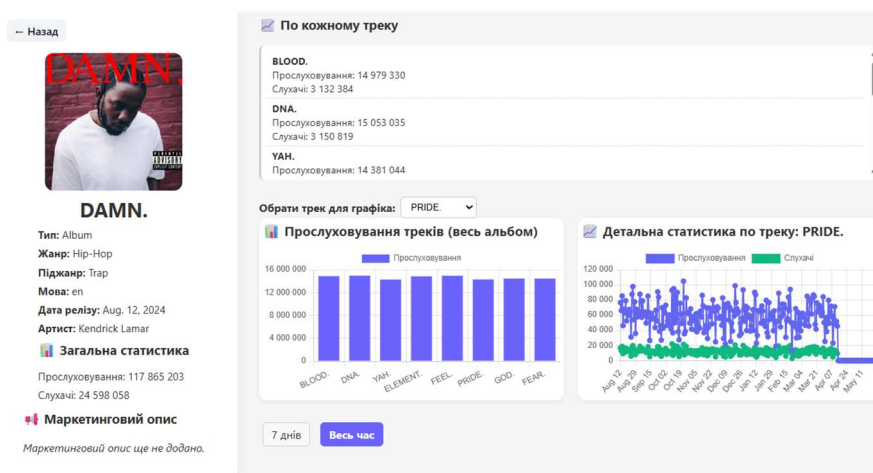


Рис. 14. Сторінка перегляду статистики випущеного альбому для артиста

Також в своєму особистому кабінеті артист може переглянути свою загальну статистику (рис. 15), де показується загальна кількість релізів, слухачів та прослуховувань. Кругова діаграма та лінійний графік демонструють динаміку прослуховувань по місяцях. Ця інформація дозволяє артисту самостійно відстежувати ефективність своєї творчої діяльності та аналізувати зміни у вподобаннях аудиторії з часом.

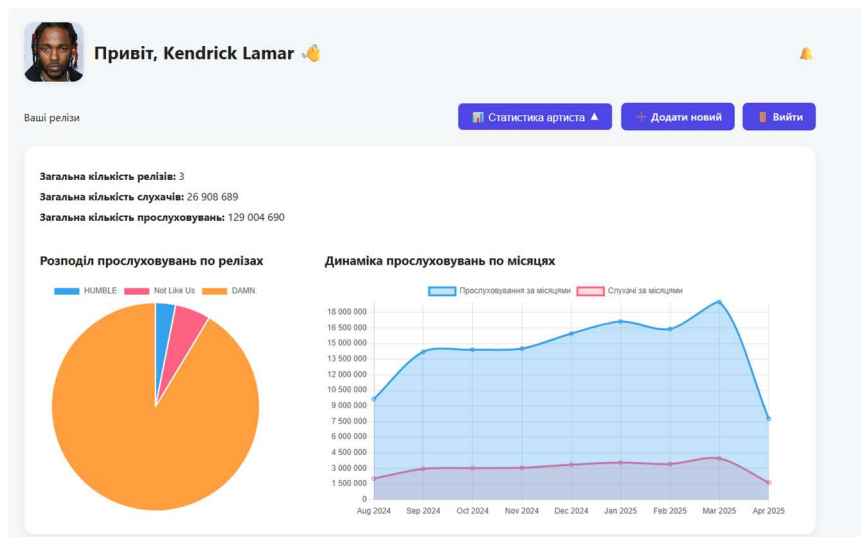


Рис. 15. Загальна статистика по артисту

Процес створення нового музичного релізу в системі розділено на три послідовні кроки. Першим етапом є введення основної інформації про реліз (рис. 16). На цій сторінці артист заповнює форму з усіма метаданими.

Новий реліз

1. Інфо | 2. Треки | 3. Підтвердження

Крок 1: Інформація про реліз

Title:

Type:

Genre:

Subgenre:

Language:

Release date:

Cover: 12.jpg

Рис. 16. Форма інформації про реліз

Наступний етап це додавання треків до релізу. Якщо тип релізу – сингл, то додається один трек. Якщо альбом – багато треків. Треки до релізу додаються динамічно через кнопку «Додати трек». До кожного треку треба додати назву, текст та аудіофайл. Форма додавання треків для альбому показана на рис. 17.

Крок 2: Треки

Title:

трек 1

Audio file:

Вибрати файл mike dean vecher 91 Cm.mp3

Lyrics:

Praesent luctus tempus ex in faucibus. Vestibulum aliquet commodo ante, eu lacinia odio varius in. Quisque suscipit velit ut elit efficitur aliquet. Ut malesuada nec dolor vitae imperdiet. Donec eleifend finibus pellentesque. Vivamus iaculis metus vitae bibendum accumsan. Interdum et malesuada fames ac ante ipsum primis in faucibus.

✕

Title:

трек 2

Audio file:

Вибрати файл lofi joji 120 C#maj popppp.mp3

Lyrics:

Fusce dapibus volutpat massa, vitae dapibus magna viverra nec. Cras ornare vel orci nec iaculis. Vestibulum ornare, eros sodales laoreet ultricies, elit lectus venenatis nulla, at finibus nunc leo quis purus. In ultrices tortor sed arcu dapibus volutpat. Phasellus eleifend ornare ex, eget aliquet quam molestie nec. Morbi feugiat ligula ac elit interdum, et ultrices ipsum venenatis. Nam

+ Додати трек
Назад
Переглянути

Рис. 17. Форма додавання треків для альбому

Після додавання усіх треків артист переходить на наступний етап – підтвердження (рис. 18). В цій секції артист переглядає усі метадані по релізу, які він заповнив та завантажив до цього. Якщо артист обрав тип релізу «альбом», але додав менше 7 треків, реліз автоматично позначається як «мініальбом». За потреби є можливість повернутись на минулі кроки, щоб виправити якусь інформацію. Після натискання кнопки «Підтвердити» реліз відправляється на модерацію менеджеру. Цей крок дозволяє артисту переконатися в коректності всієї введеної інформації перед надсиланням релізу на перевірку. Завдяки автоматичному визначенню типу релізу система мінімізує помилки класифікації, що спрощує подальшу модерацію. Після відправлення релізу артист не має можливості вносити зміни, доки менеджер лейблу не перегляне його. Це забезпечує чітку послідовність етапів публікації та зберігає цілісність процесу.

Крок 3: Підтвердження

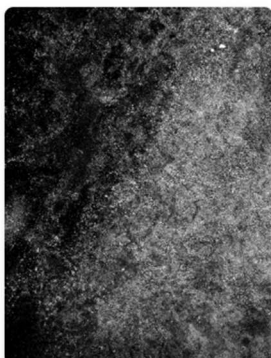
Назва: lets go

Жанр: Нір-Нор

Дата релізу: 2025-05-30

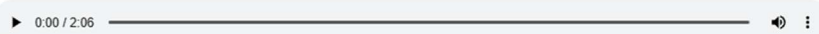
Мова: English

Тип релізу: Мініальбом



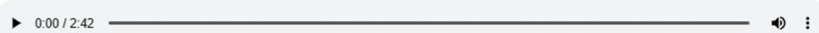
Треки:

трек 1



Показати/сховати текст

трек 2



Показати/сховати текст

Назад

Підтвердити та зберегти

Рис. 18. Секція підтвердження релізу