

# Program do katalogowania plików dźwiękowych

## Spis treści

Wstęp .....	4
1. Przegląd istniejących narzędzi do katalogowania plików dźwiękowych.....	6
2. Zastosowane technologie .....	12
3. Specyfikacja techniczna i projekt interfejsu .....	15
3.1. Diagram ERD .....	15
3.2. Diagramy DFD .....	17
3.3. Zastosowanie zasad Nielsena .....	20
3.4. Zastosowanie Material Design przy projektowaniu interfejsu .....	21
4. Opis aplikacji .....	24
4.1. Pasek nawigacyjny .....	24
4.2. Strona główna .....	25
4.3. Strona rejestracji .....	26
4.4. Strona logowania .....	27
4.5. Strona z listą plików .....	28
4.6. Strona szczegółów plików audio .....	30
4.7. Strona dodawania nowego pliku audio.....	31
4.8. Strona wgrywania nowego pliku z jego metadanymi.....	33
4.9. Strona wgrywania nowego pliku z wyszukiwaniem jego metadanych .....	37
4.10. Strona profilu.....	38
4.11. Wygląd aplikacji na urządzeniu mobilnym .....	39
Podsumowanie .....	41
Spis literatury .....	43



## Wstęp

Celem projektu jest opracowanie i implementacja programu do katalogowania plików dźwiękowych. Program ma umożliwić efektywne zarządzanie kolekcjami plików audio różnych formatów poprzez zapewnienie narzędzi do ich przeszukiwania oraz organizowania. Projekt obejmuje analizę istniejących rozwiązań dostępnych na rynku. Obecnie dostępne rozwiązania szczególnie w języku polskim, często są zbyt skomplikowane lub nie spełniają potrzeb użytkowników. W ramach tej analizy oceniane będą niezbędne funkcje takich aplikacji, ich intuicyjność, bezpieczeństwo oraz poziom atrakcyjności wizualnej. Szczególny nacisk zostanie położony na zidentyfikowanie braków oraz ograniczeń w dostępnych programach, które mogą wpływać na doświadczenie użytkownika. Na podstawie wyników analizy zdefiniowane zostaną kluczowe funkcje, które powinny znaleźć się w projektowanym systemie. W ramach realizacji projektu zastosowano nowoczesne technologie, które zapewniają wysoką wydajność, bezpieczeństwo oraz łatwość rozbudowy systemu. Przeprowadzona zostanie również szczegółowa analiza techniczna, która obejmie opracowanie interfejsu użytkownika oraz specyfikacji technicznej. Praca obejmie również stworzenie diagramu ERD, który przedstawi strukturę bazy danych i relacje pomiędzy jej elementami, co zapewni efektywne przechowywanie danych. W celu zrozumienia przepływu danych w systemie opracowane zostaną diagramy DFD, które ilustrują procesy przetwarzania informacji w aplikacji. Interfejs stworzony zostanie zgodnie z zasadami użyteczności Nielsena, co gwarantuje intuicyjność obsługi oraz łatwość korzystania z aplikacji. Dodatkowo podczas projektowania interfejsu uwzględnione zostaną wytyczne Material Design, które zapewniają estetyczny wygląd oraz spójność wizualną. Dzięki temu aplikacja łączy ze sobą użyteczność wraz z atrakcyjnym wyglądem, co sprawia, że jest dostosowana do potrzeb użytkowników. Ze względu na to, że większość dostępnych narzędzi nie spełnia dzisiejszych standardów, aplikacja ma na celu wypełnienie tej luki poprzez stworzenie nowoczesnego narzędzia, które będzie zawierało najważniejsze funkcje, do których przyzwyczaili się użytkownicy takich aplikacji. Dzięki rozwiązaniom, takim jak automatyczne pobieranie metadanych z zewnętrznych baz danych czy jednoczesnego dodawania dużej ilości plików, aplikacja ma szansę przyciągnąć nowych

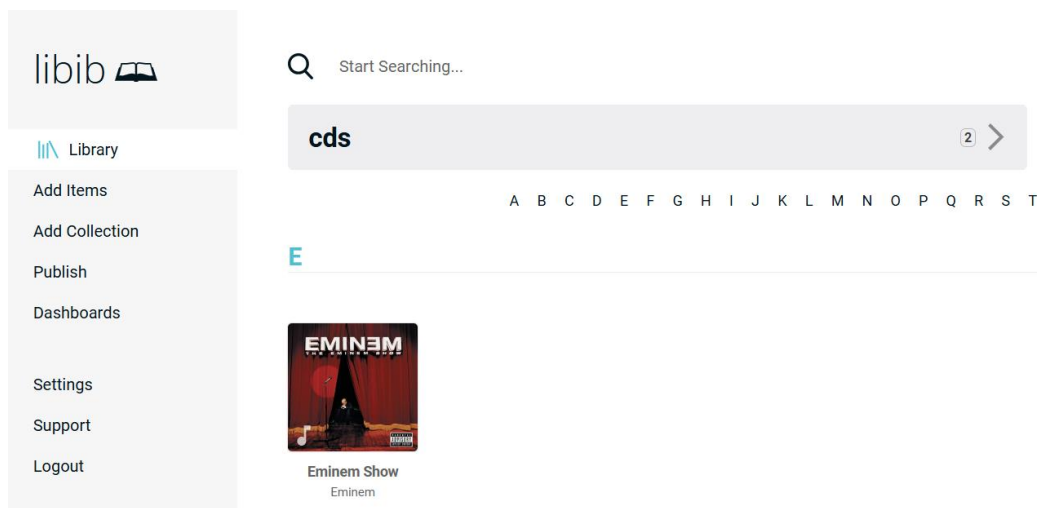
użytkowników. Projekt uwzględnia również rosnące wymagania użytkowników dotyczące bezpieczeństwa danych oraz kompatybilności z urządzeniami mobilnymi.

## 1. Przegląd istniejących narzędzi do katalogowania plików dźwiękowych

W projekcie istotne jest przeprowadzenie szczegółowego przeglądu dostępnych na rynku programów do katalogowania plików dźwiękowych. Dzięki temu możliwe będzie zidentyfikowanie już istniejących funkcji oraz określenie, które funkcje są najbardziej pożądane przez użytkowników. Poniżej przedstawiony jest przegląd wybranych narzędzi.

### **Libib**

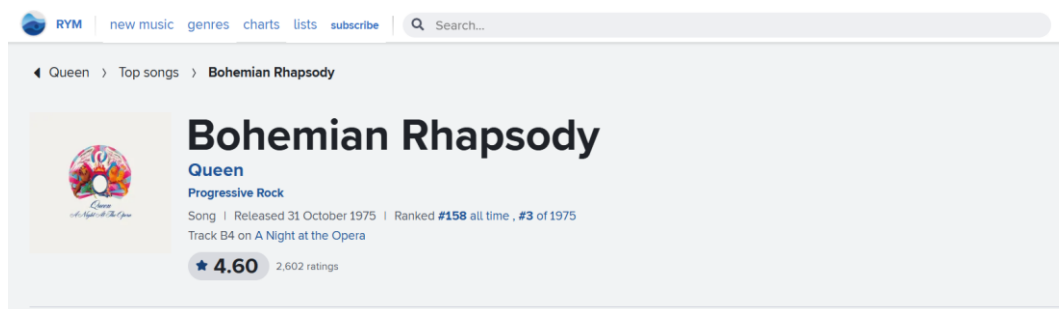
Libib to aplikacja webowa stosowana do zarządzania różnymi kolekcjami multimedialnymi w tym również plikami dźwiękowymi. Umożliwia tworzenie oraz zarządzanie kolekcjami. Plik można dodawać poprzez ręczne wprowadzanie danych lub skanowanie kodów kreskowych fizycznych nośników. Dzięki temu proces dodawania plików jest szybki i efektywny, niezależnie od formatu źródła. Użytkownicy mogą dodawać tagi, opisy, metadane. Metadane mogą obejmować informacje takie jak tytuł, artysta, album, rok wydania, gatunek i inne istotne szczegóły, co ułatwia przeszukiwanie i zarządzanie dużymi zbiorami plików dźwiękowych. Libib oferuje zaawansowane funkcje wyszukiwania, które pozwalają na szybkie odnalezienie konkretnych plików w kolekcji. Użytkownicy mogą filtrować pliki według różnych kryteriów, takich jak tytuł, autor, gatunek, tagi i inne metadane. Libib integruje się z różnymi bazami danych online, co umożliwia automatyczne pobieranie metadanych i okładek dla dodawanych plików. Ta funkcja znacznie upraszcza proces organizowania kolekcji, ponieważ użytkownicy nie muszą ręcznie wprowadzać wszystkich szczegółów. Dzięki integracji z bazami danych, Libib może automatycznie aktualizować informacje o plikach, zapewniając, że wszystkie metadane są zawsze aktualne i dokładne. Wygląd aplikacji został przedstawiony na rysunku poniżej [1].



Rysunek 1 Strona libib

## RateYourMusic

RateYourMusic to aplikacja internetowa, która umożliwia pasjonatom muzyki katalogowanie swoich kolekcji muzycznych, ocenianie albumów i utworów oraz dzielenie się opiniami. Użytkownicy mogą tworzyć szczegółowe katalogi, które obejmują zarówno fizyczne nośniki, jak i pliki dźwiękowe. Każdy dodany plik dźwiękowy można opatrzyć metadanymi, takimi jak tytuł, artysta, data wydania, gatunek muzyczny i więcej. Platforma dysponuje obszerną bazą danych zawierającą szczegółowe informacje o albumach, singlach i innych wydawnictwach muzycznych. Interfejs aplikacji uznawany jest za przestarzały i mało intuicyjny, co może stanowić wyzwanie dla nowych użytkowników. Dodatkowo brakuje automatycznej synchronizacji plików, co oznacza konieczność ręcznego dodawania nowych utworów do kolekcji. Niemniej jednak, silnym punktem RateYourMusic jest aktywna społeczność, która dodaje oceny i recenzje, co pozwala użytkownikom organizować swoje kolekcje według preferencji i gustów muzycznych. Wygląd strony można zobaczyć na poniższym zdjęciu [2].

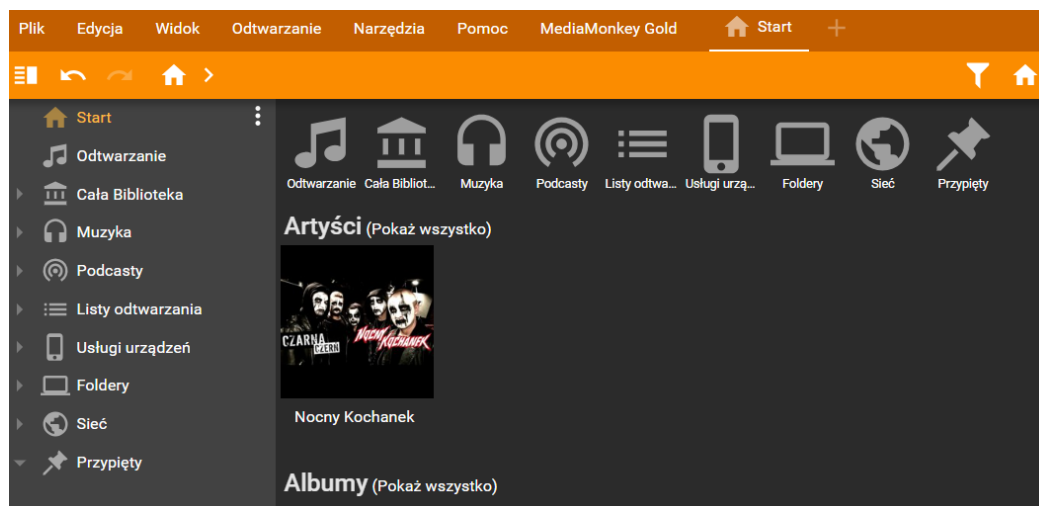


*Rysunek 2 Strona RateYourMusic*

## MediaMonkey

MediaMonkey jest programem służącym do zarządzania plikami dźwiękowymi, który posiada dużo funkcji ułatwiających katalogowanie oraz organizowania plików. Program umożliwia użytkownikom zarządzanie dużymi zbiorami plików audio, w tym działania na metadanych. Aplikacja obsługuje różne formaty audio, dzięki czemu możliwe jest zarządzanie katalogami poprzez formaty. Narzędzie automatycznie skanuje zdefiniowane lokalizacje na dysku w celu znalezienia plików muzycznych, po czym automatycznie dodaje je do biblioteki. Opcja ta ułatwia utrzymanie biblioteki bez konieczności ręcznego dodawania każdego pojedynczego pliku. Funkcje takie jak zbiorowe edytowanie tagów, w szybki sposób aktualizują informacje o kilku plikach jednocześnie. Korzystanie z internetowej bazy danych zapewnia aktualność oraz kompletność danych. MediaMonkey oferuje zaawansowane opcje odtwarzania audio, w tym możliwość tworzenia niestandardowych playlist i list opartych na konkretnych kryteriach, takich jak ocena, data czy gatunek. Program pozwala również na konwersję plików audio między różnymi formatami, dzięki czemu jest zachowana kompatybilność między różnymi urządzeniami. Wygląd aplikacji widać na rysunku poniżej [3].

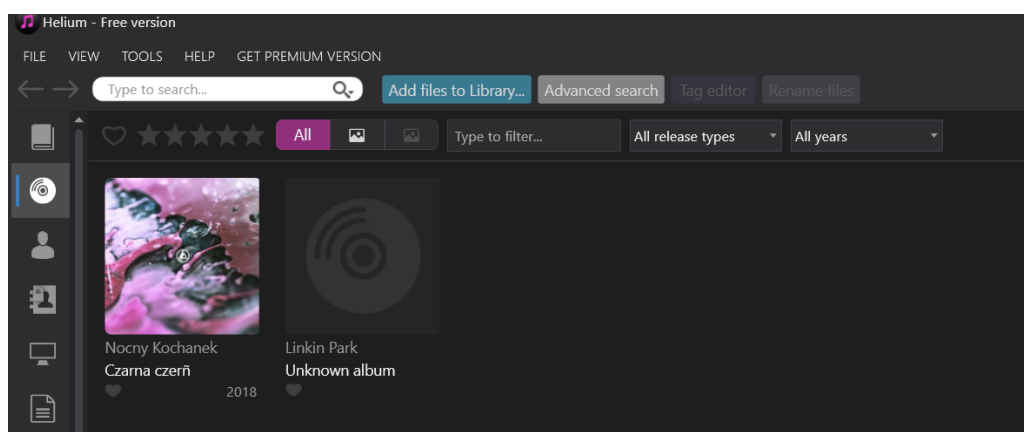




Rysunek 3 Aplikacja MediaMonkey

## Helium Music Manager

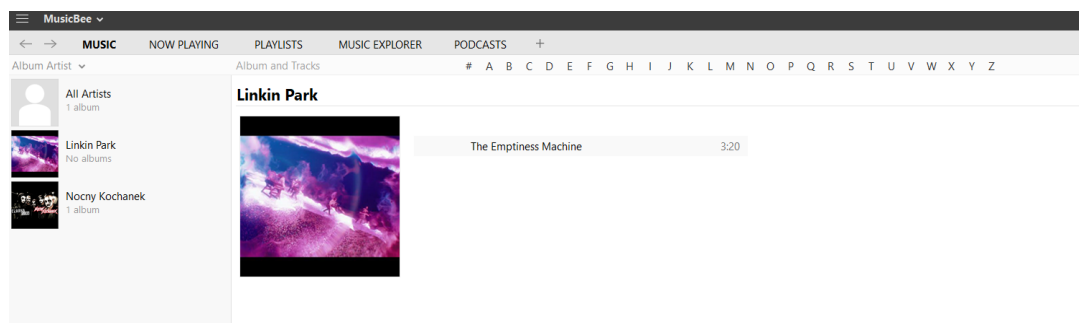
Helium Music Manager to wszechstronne oprogramowanie do zarządzania kolekcjami plików audio, które oferuje bogaty zestaw funkcji dla użytkowników pragnących efektywnie katalogować i organizować swoją muzykę. Program pozwala na łatwe dodawanie plików do biblioteki, automatyczne skanowanie lokalnych folderów w celu wykrywania plików audio oraz edytowanie ich metadanych. Helium Music Manager obsługuje wiele formatów plików audio, co czyni go elastycznym rozwiązaniem dla różnych potrzeb użytkowników. Wygląd aplikacji przedstawiony jest na poniższym rysunku [4].



Rysunek 4 Aplikacja Helium Music Manager

## MusicBee

MusicBee to potężne narzędzie do zarządzania muzyką, które łączy w sobie funkcje odtwarzania, katalogowania oraz organizowania plików audio. Program charakteryzuje się intuicyjnym interfejsem, który umożliwia łatwe poruszanie się po bibliotece muzycznej, a także personalizację wyglądu i układu. Użytkownicy mogą dodawać pliki muzyczne do biblioteki poprzez przeciąganie ich z lokalnych folderów, a program automatycznie skanuje system w poszukiwaniu nowych utworów. MusicBee wspiera wiele formatów audio i pozwala na edytowanie metadanych, takich jak tytuł, artysta, album, rok wydania, gatunek i inne. Użytkownicy mogą tworzyć zaawansowane playlisty, korzystać z funkcji automatycznego uzupełniania metadanych na podstawie informacji z internetowych baz danych oraz synchronizować swoje kolekcje z urządzeniami mobilnymi. Wygląd aplikacji przedstawiony został na zdjęciu poniżej [5].

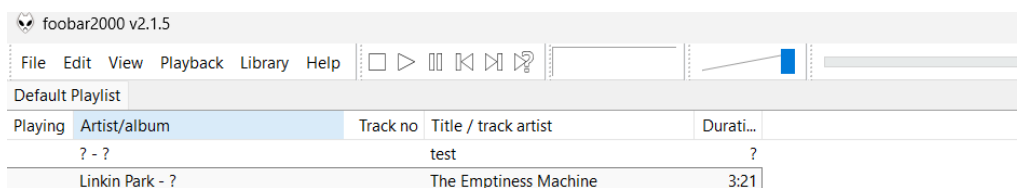


*Rysunek 5 Aplikacja MusicBee*

## **Foobar2000**

Foobar2000 to zaawansowany odtwarzacz multimedialny, który poza funkcjami odtwarzania oferuje również rozbudowane możliwości zarządzania plikami audio. Program wyróżnia się minimalistycznym i konfigurowalnym interfejsem, co pozwala użytkownikom dostosować go do własnych potrzeb. Foobar2000 obsługuje wiele formatów plików audio, co czyni go elastycznym rozwiązaniem do zarządzania różnorodnymi zbiorami muzycznymi. Program umożliwia edytowanie metadanych, tworzenie niestandardowych list odtwarzania oraz filtrowanie plików według różnych kryteriów. Choć interfejs może być mniej intuicyjny dla nowych użytkowników, jego zaletą jest ogromna możliwość dostosowania poprzez wtyczki, które rozszerzają funkcje oprogramowania. Foobar2000 nie posiada natywnej integracji

z internetowymi bazami danych, co oznacza, że użytkownicy muszą ręcznie dodawać i aktualizować metadane. Widok aplikacji przedstawiony został poniżej [6].



Rysunek 6 Foobar2000

Wnioski z przeglądu programów do katalogowania plików dźwiękowych wskazują na główne funkcje, które powinny znaleźć się w aplikacji. Użytkownik powinien mieć możliwość ręcznego ustalania metadanych oraz ich intuicyjnej edycji, a także korzystania z opcji automatycznego pobierania danych z zewnętrznych baz, co ułatwi i przyspieszy proces katalogowania. Implementacja zaawansowanego wyszukiwania według różnych kryteriów pozwoli na szybkie znajdowanie potrzebnych danych. Wsparcie dla wielu formatów plików umożliwi łatwiejsze zarządzanie zbiorami muzycznymi. Zarządzanie plikami powinno obejmować dodawanie nowych elementów do biblioteki poprzez wybranie z lokalizacji na dysku lub przeciągnięcie na stronę, a także tworzenie niestandardowych playlist opartych na różnych kryteriach. Intuicyjny i przyjazny interfejs użytkownika, umożliwiający łatwą nawigację i personalizację układu, jest niezbędny, podobnie jak opcja zmiany wyglądu aplikacji zgodnie z preferencjami użytkowników. Aplikacja powinna być dostępna zarówno na urządzeniach mobilnych, jak i komputerach, co zapewni wygodę i elastyczność w zarządzaniu kolekcjami audio. Kluczowe jest także wdrożenie środków bezpieczeństwa, takich jak szyfrowanie danych, aby chronić zasoby aplikacji i prywatność użytkowników.

## 2. Zastosowane technologie

W niniejszym rozdziale omówione zostaną technologie wykorzystane w projekcie, które przyczyniły się do stworzenia nowoczesnej aplikacji webowej do zarządzania plikami audio. Każda z tych technologii odegrała istotną rolę w budowie aplikacji, zapewniając jej niezawodność, bezpieczeństwo oraz wydajność. Dzięki nim możliwe było zrealizowanie zaawansowanych funkcji, które spełniają oczekiwania użytkowników. W kolejnych podpunktach przedstawione zostaną szczegółowe informacje na temat każdej z tych technologii, ich funkcji oraz korzyści, jakie wniosły do projektu.

Java jest wszechstronnym i popularnym językiem programowania, znanym ze swojej niezawodności i wydajności. Dzięki obiektowości oraz rozbudowanej bibliotece klas, Java jest idealnym wyborem do tworzenia aplikacji backendowych. Jedną z jej zalet jest możliwość uruchamiania aplikacji na różnych systemach operacyjnych bez potrzeby wprowadzania modyfikacji w kodzie źródłowym [7].

Spring to potężny framework dla Javy, który znacznie ułatwia rozwój aplikacji, szczególnie w zakresie architektury mikroservisowej. Oferuje zestaw narzędzi do zarządzania zależnościami, programowania aspektowego oraz integracji z różnymi technologiami. Spring wspiera również różne podejścia do tworzenia aplikacji, takie jak RESTful API, co jest kluczowe w przypadku aplikacji webowych [8].

Spring Security to rozszerzenie frameworku Spring, które zapewnia kompleksowe wsparcie w zakresie bezpieczeństwa aplikacji. Umożliwia implementację autoryzacji użytkowników, co jest niezbędne w przypadku aplikacji przetwarzających wrażliwe dane, jak pliki audio [9].

React to popularna biblioteka JavaScript do budowy interfejsów użytkownika, która umożliwia tworzenie dynamicznych i responsywnych aplikacji webowych. Dzięki architekturze komponentowej, React pozwala na łatwe zarządzanie stanem aplikacji oraz ponowne wykorzystanie kodu. Interakcja z backendem za pomocą API REST, które jest obsługiwane przez aplikację napisaną w Spring, sprawia, że komunikacja między frontendem a backendem jest prosta i wydajna [10].

PostgreSQL to zaawansowany system zarządzania bazami danych, znany z wysokiej wydajności. Obsługuje zaawansowane typy danych oraz skomplikowane zapytania, co sprawia, że jest idealnym wyborem dla aplikacji wymagających przechowywania i przetwarzania dużych ilości danych. Dzięki wsparciu dla transakcji oraz mechanizmowi zapewnienia integralności danych, PostgreSQL zapewnia bezpieczeństwo i spójność przechowywanych informacji, co jest kluczowe w kontekście zarządzania plikami audio i metadanymi [11].

MusicBrainz to otwarta, społecznościowa baza danych, która zawiera szczegółowe informacje o utworach, artystach, albumach i innych aspektach muzyki. API MusicBrainz umożliwia deweloperom dostęp do tych informacji, co czyni go cennym narzędziem dla aplikacji związanych z muzyką. API MusicBrainz zostało wybrane dzięki swojej otwartości oraz dostępności, w przeciwieństwie do API Spotify, które również było brane pod uwagę, MusicBrainz nie wymaga przestrzegania restrykcyjnych zasad licencyjnych, przez co elastyczność aplikacji jest ograniczana. Dodatkowo API MusicBrainz jest dostępne globalnie i może być używane w każdym miejscu na świecie. MusicBrainz jest w pełni darmowe, dzięki czemu można z niego korzystać bez aktywnej subskrypcji [12].

ID3v2 jest obecnym standardem do opisu metadanych plików audio. Umożliwia dodawanie różnorodnych informacji o utworach. Dzięki zawartym w nim danym ułatwia katalogowanie oraz organizację zbiorów muzycznych, pozwalając na przeszukiwanie według metadanych [13].

Cover Art Archive jest otwartym repozytorium okładek albumów, który działa we współpracy z MusicBrainz. Unikalny identyfikator utworów MBID pozwala na łatwą integrację tych dwóch programów i wykorzystanie ich w jednej aplikacji. Cover Art Archive wspiera wyszukiwanie oraz pobieranie okładek w różnych rozdzielczościach, co umożliwia wyświetlanie ich na stronie internetowej zarówno na komputerze jak i na telefonie. Dane z tej bazy dostępne są na licencji open-source, dzięki czemu nie występują problemy z własnością grafik [14].



### 3. Specyfikacja techniczna i projekt interfejsu

W tym rozdziale przedstawione zostaną aspekty techniczne, który brane były pod uwagę w czasie tworzenia aplikacji.

#### 3.1. Diagram ERD

W diagramie relacji encji aplikacji internetowej, gdzie każdy użytkownik ma swoje własne pliki, niezbędne jest utworzenie systemu z rejestracją oraz logowaniem. Dla jak najlepszego działania i pomocy użytkownikom w razie problemów, dodane zostało pole określające role, można w nim ustawić administratora, który posiada dodatkowe funkcje. Kluczem głównym jest pole id, zawarte są również standardowe pole dla takich systemów, czyli nazwa użytkownika, email oraz hasło.

Do przechowywanie danych o plikach została stworzona tabela audio. Pola zostały wybrane na podstawie standardu Id3v2 oraz indywidualnych potrzeb systemu. Kluczem głównym jest id, a dodatkowe pola obejmują:

audio\_file – przechowuje dane binarne pliku audio

cover\_art – przechowuje dane binarne okładki pliku

title – tytuł utworu

artist, album, year – dane opisowe o utworze

genre, track, lyrics, comment – dodatkowe informacje, takie jak gatunek, tekst piosenki czy komentarze

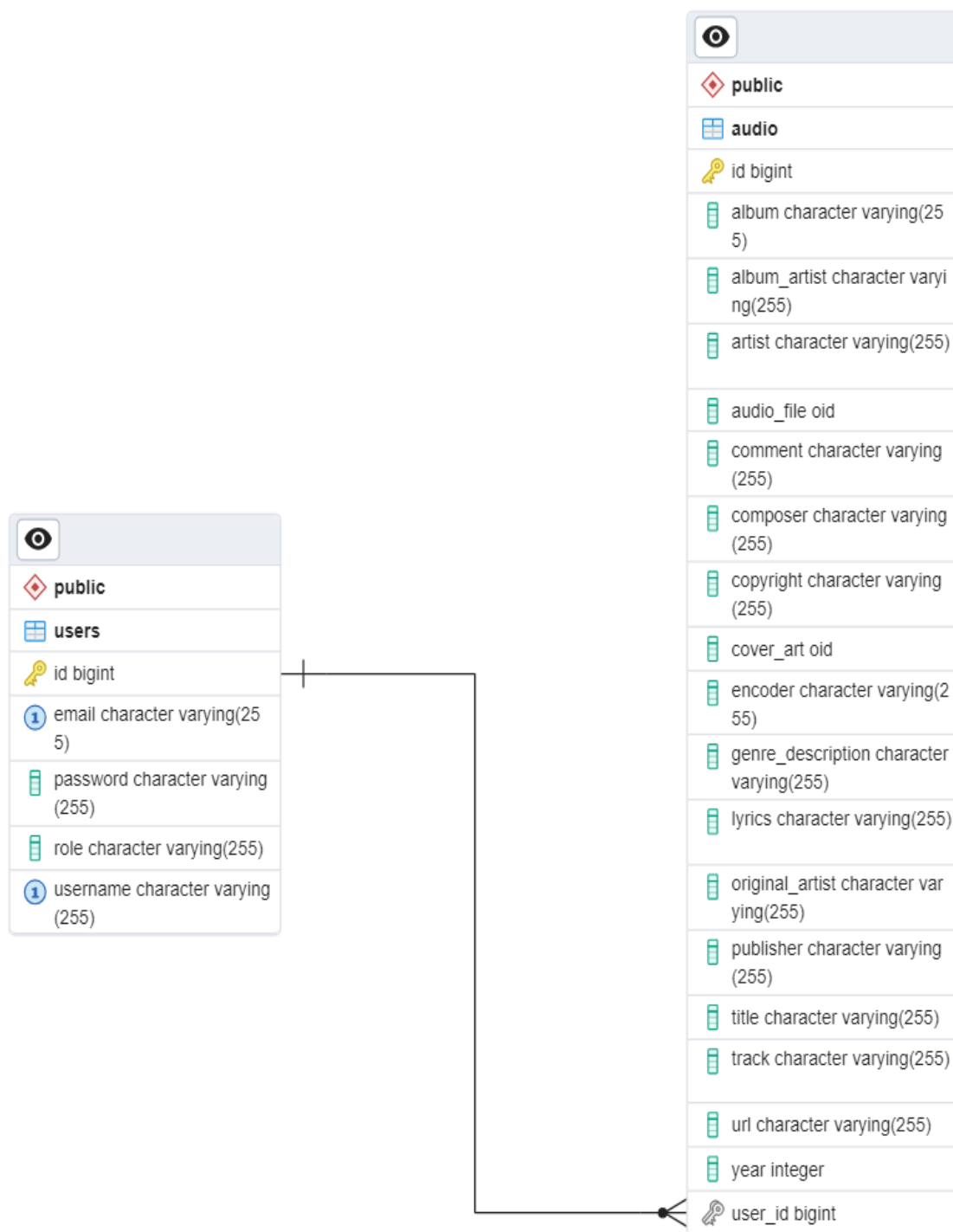
composer, publisher, original\_artist, album\_artist – dane związane z twórcami i producentami utworu

url, encoder – adres URL do źródła pliku oraz format kodowania audio

user\_id – klucz obcy, wskazujący na użytkownika, do którego plik należy

Relacja pomiędzy tabelą users a tabelą audio jest relacją typu jeden do wielu. Oznacza to, że jeden użytkownik, może być właścicielem wielu plików audio, ale każdy plik

audio posiada wyłącznie jednego użytkownika. Pełny diagram ERD został przedstawiony poniżej:



Rysunek 7 Diagram ERD

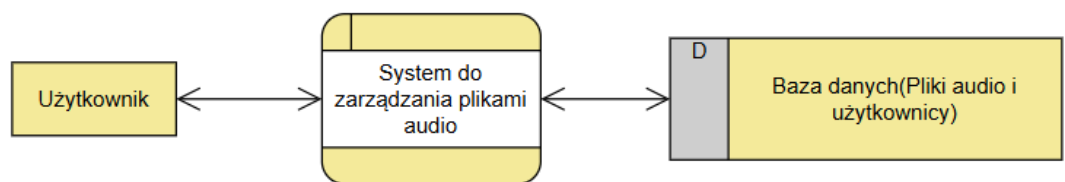


### 3.2. Diagramy DFD

W celu pełniejszej analizy aplikacji zaprojektowane zostały diagramy przepływu danych zerowego oraz pierwszego poziomu.

#### Diagram DFD poziomu 0

Na diagramie DFD poziomu 0 znajduje się użytkownik, który wysyła do systemu dane, takie jak rejestracja, logowanie czy przesyłanie plików. Użytkownik otrzymuje z systemu dane o plikach czy komunikaty o błędach. Proces główny przetwarza wszystkie operacje użytkownika oraz komunikuje się z bazą w celu zapisywania i pobierania danych. Baza danych przechowuje dane użytkowników i plików audio. Schemat został przedstawiony poniżej.

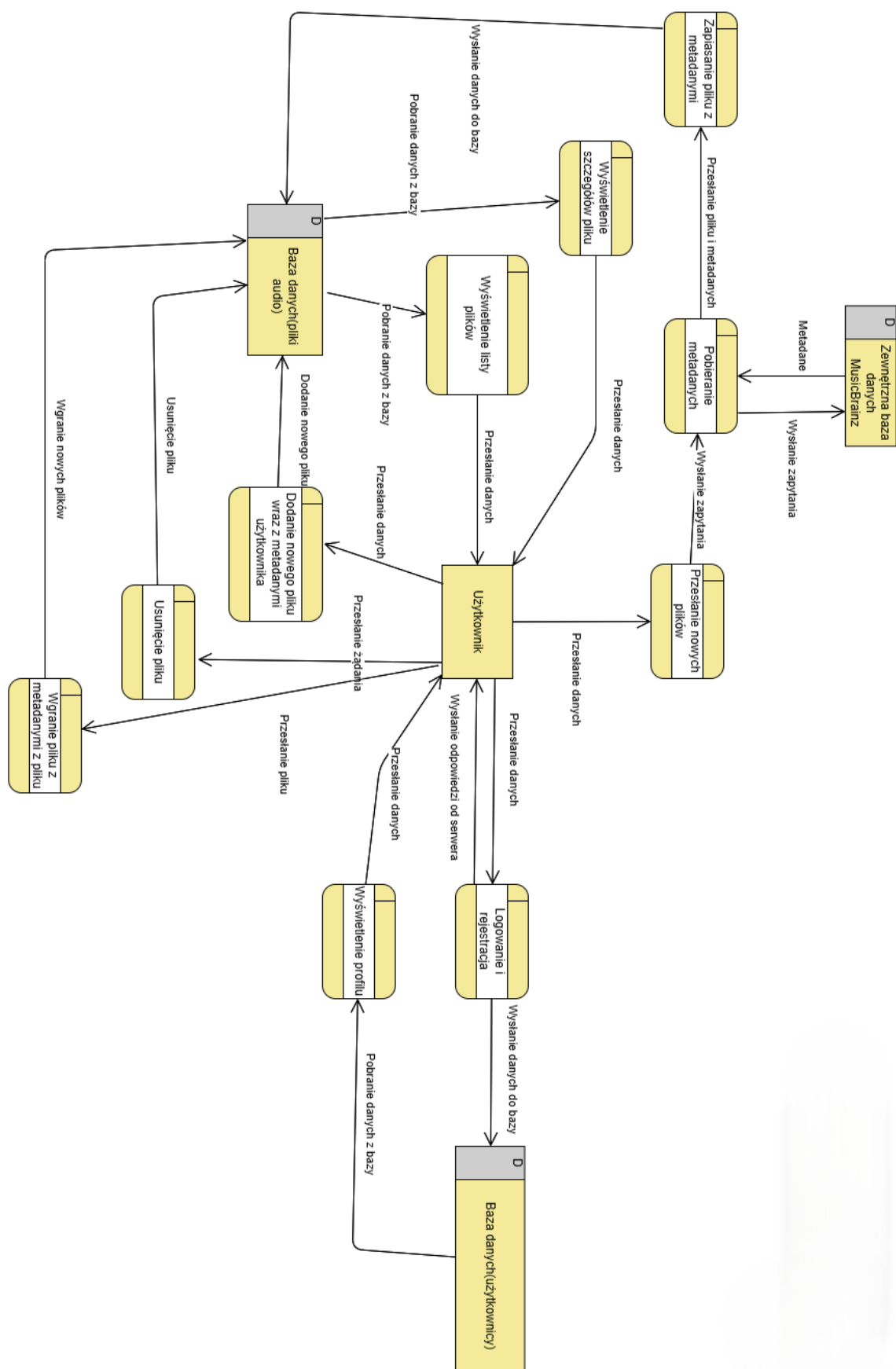


Rysunek 8 Diagram DFD poziomu 0

#### Diagram DFD poziomu 1

Diagram DFD poziomu 1 przedstawia strukturę aplikacji zarządzającej plikami audio, w której użytkownik pełni główną rolę. Ma on możliwość rejestracji oraz logowania, a dane są przesyłane do bazy danych. Po weryfikacji użytkownik dostaje odpowiedź z serwera. Również z tej samej tabeli użytkownik może dostać informacje o swoim profilu. Proces dodawania nowego pliku wraz z metadanymi użytkownika odbywa się poprzez przesłanie pliku, a następnie wrzucenie go do bazy. Proces wgrania pliku z metadanymi z pliku działa na podobnej zasadzie z tą różnicą, że metadane są pobierane z pliku a nie przesyłane razem z nim. Z kolei proces przesłanie nowych plików przechodzi w proces pobierania metadanych, gdzie dane pobierane są

z zewnętrznej bazy MusicBrainz. Po pobraniu danych plik oraz metadane zapisywane są w bazie danych. Procesy wyświetlenia listy plików oraz wyświetlanie szczegółów pliku pobierają odpowiednie dane z bazy i przekazują je użytkownikowi. Diagram został przedstawiony na poniższym rysunku.



Rysunek 9 Diagram DFD poziomu 1

### 3.3. Zastosowanie zasad Nielsena

Zasady Nielsena to zbiór 10 heurystyk użyteczności opracowanych przez Jakoba Nielsena, które pomagają projektować intuicyjne, estetyczne i przyjazne dla użytkownika interfejsy. Stanowią one praktyczne wskazówki do tworzenia aplikacji, które są łatwe w obsłudze i odpowiadają na potrzeby użytkowników [15].

Aplikacja informuje użytkownika o ważnych etapach, takich jak proces przesyłania plików do bazy czy ładowania nowych plików na liście, dzięki czemu użytkownik jest świadomy aktualnego stanu systemu. Proste terminologie, takie jak „Pliki” czy „Dodaj nowy plik”, ułatwiają użytkownikowi orientację i zrozumienie dostępnych funkcji. Użytkownik ma pełną kontrolę nad aplikacją, w każdej chwili mogąc usunąć niechciane pliki lub przerwać wgrywanie nowych. System zachowuje spójność w różnych częściach aplikacji, co pozwala na eliminację nauki układu na nowo. Przykładowo, wszystkie przyciski na górnym pasku są zawsze w tych samych miejscach, z niezmiennymi funkcjami.

Aplikacja minimalizuje ryzyko popełnienia błędów, oferując walidację danych i wyświetlając komunikaty w przypadku niepoprawnego ich wprowadzenia. Na przykład podczas dodawania pliku użytkownik może wybrać gatunek muzyczny jedynie z listy, co zapewnia zgodność z normą ID3v2. Istotne informacje, takie jak nazwa pliku czy artysta, są wyświetlane w sposób przejrzysty, co eliminuje konieczność zapamiętywania szczegółów, a szybki dostęp do listy plików i ich metadanych sprawia, że korzystanie z aplikacji jest intuicyjne. Projekt aplikacji uwzględnia elastyczność i efektywność, użytkownik może dodać plik z własnymi metadanymi lub z metadanymi zapisanymi w pliku, a także wybierać pomiędzy dodawaniem pojedynczych plików a przesyłaniem całych folderów, co zwiększa efektywność pracy.

Interfejs aplikacji został zaprojektowany z uwzględnieniem estetyki i umiaru – zawiera tylko niezbędne elementy, dzięki czemu użytkownik nie jest przytłoczony nadmiarem informacji. Obsługa błędów odbywa się za pomocą formularzy, które nie pozwalają na niepoprawne wprowadzenie danych, a w razie problemów, takich jak niewgranie pliku, użytkownik otrzymuje jasny komunikat o sytuacji. Pomoc w użytkowaniu jest zapewniona poprzez opisy w poszczególnych zakładkach, które

wyjaśniają ich funkcje, oraz wskazówki przy dodawaniu plików, które informują o wymaganych danych do poprawnego przesłania pliku wraz z metadanych.

### 3.4. Zastosowanie Material Design przy projektowaniu interfejsu

Material Design jest system projektowania opracowanym przez Google, zawiera on zbiór zasad i wytycznych dotyczących projektowania interfejsów użytkownika oraz doświadczeń użytkownika. Jego celem jest zapewnienie spójności wizualnej, intuicyjności oraz estetyki w aplikacjach na różnych platformach i urządzeniach. Wykorzystanie Material Design w projekcie umożliwiło stworzenie interfejsu, który jest nowoczesny i zgodny ze standardami dostępności WCAG. Dzięki temu aplikacja łączy atrakcyjny wygląd z wygodą użytkowania, jednocześnie zachowując spójność wizualną i zgodność z najlepszymi praktykami projektowania [16].

Jako główny kolor został wybrany fioletowy (#AB47BC), a jako pomocniczy czarny (#000000). Dzięki takiemu podejściu kolorystyka zgodna jest z ideą używania kolorów w Material Design, gdzie kolor główny i pomocniczy współgrają ze sobą, tworząc estetycznie przejścia oraz tło dla całej strony. Aby zachować spójność wizualną, inne elementy interfejsu, takie jak ramki, podświetlenia, nagłówki czy tabele, również wykorzystują wariacje głównego koloru #AB47BC. Kolory te są dobierane zgodnie z paletą Material Design, co pozwala na różnicowanie elementów w sposób estetyczny. Teksty i elementy kontrastowe są projektowane w kolorach zapewniających czytelność, takich jak biały lub czarny, w zależności od podstrony.

Kontenery, pola typu input, label oraz lista plików zostały zaprojektowane jako jednolite bloki. Tło tych elementów jest utrzymane w spójnym, wyróżniającym się na tle odcieniu koloru pomocniczego. Taki układ sprzyja także łatwiejszemu skupieniu uwagi użytkownika na istotnych elementach.

Czcionka Arial, zastosowana w aplikacji, dzięki byciu czcionką *sans-serif*, doskonale nadaje się do wyświetlania na ekranach zarówno o małej, jak i dużej rozdzielczości. Jej minimalistyczny styl idealnie wpisuje się w zasady Material Design, które kładą nacisk na spójność i klarowność interfejsu użytkownika.

Kluczowym elementem Material Design jest zapewnienie responsywności interfejsu na różnych urządzeniach. W tym celu zaprojektowano przestronne i elastyczne układy. W aplikacji wdrożono mechanizmy dostosowujące rozmiar oraz rozmieszczenie elementów interfejsu, takich jak kontenery, pola wejściowe czy listy plików, aby zapewnić optymalne doświadczenie użytkownika niezależnie od urządzenia. Na dużych ekranach zastosowano przestronny układ z większymi marginesami i bardziej złożoną siatką. Dla mniejszych ekranów, takich jak telefony, interfejs dynamicznie przełącza się na prostszy układ z pionowym rozmieszczeniem wszystkich elementów, co pozwala zachować estetykę i funkcje aplikacji.

W aplikacji wykorzystano grafikę w formie symbolicznej nutki, która pełni funkcję przycisku nawigacyjnego prowadzącego na stronę główną. Element został zaprojektowany zgodnie z zasadami Material Design, gdzie priorytetem są czytelność, estetyka i intuicyjność. Nutka jako motyw graficzny nie tylko estetycznie wzbogaca interfejs, ale także bezpośrednio nawiązuje do głównej tematyki aplikacji, czyli zarządzanie plikami audio. Przycisk w formie ikony jest łatwy do zrozumienia dla użytkownika, co wspiera zasadę dostępności interfejsu



## 4. Opis aplikacji

W rozdziale przedstawiona zostanie cała aplikacja. W kolejnych podrozdziałach przedstawione zostaną poszczególne strony, ich funkcje oraz najbardziej interesujące części kodu odpowiadające za poszczególne funkcje.

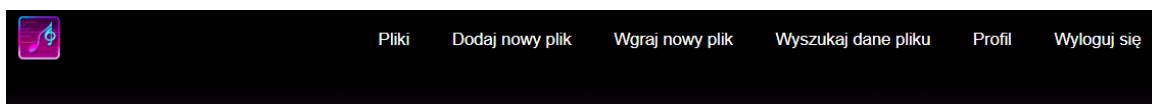
### 4.1. Pasek nawigacyjny

Dla użytkownika niezalogowanego w pasku widnieją przyciski przekierowujące na zakładkę z logowaniem oraz rejestracją. Dodatkowo znajduje się tam logo aplikacji, które po kliknięciu przenosi na stronę główną. Pasek przedstawiony został poniżej.



*Rysunek 10 Pasek nawigacyjny dla użytkownika niezalogowanego*

Użytkownik zalogowany również ma możliwość wrócenia do strony głównej poprzez kliknięcie ikony. Ponadto na górnym pasku znajdują się jeszcze przycisk „Pliki”, który przenosi na listę z wszystkimi plikami danego użytkownika. Przycisk „Dodaj nowy plik” przenosi na stronę, gdzie można dodać plik, który będzie miał metadane ustawione przez użytkownika. Zakładka „Wgraj nowy plik” pozwala użytkownikowi na wgranie jednego lub wielu plików wraz z jego metadanymi. Strona „Wyszukaj dane pliku” również daje możliwość wgrania swoich plików, jednak metadane pobierane są z zewnętrznych serwisów. W prawym rogu znajduje się jeszcze zakładka z profilem oraz przycisk do wylogowania się. Pasek przedstawiony został na poniższym rysunku.



*Rysunek 11 Pasek nawigacyjny dla użytkownika niezalogowanego*



## 4.2. Strona główna

Strona główna dla użytkownika niezalogowanego informuje użytkownika, jakie kroki powinien zrobić, aby zacząć korzystać z aplikacji. Przystawione jest to na poniższym zdjęciu.



*Rysunek 12 Strona główna dla niezalogowanego użytkownika*

Strona główna użytkownika zalogowanego informuje go ile plików przechowuje w aplikacji oraz tłumaczy do czego służą poszczególne zakładki. Przedstawione zostało to na rysunku niżej.



Rysunek 13 Strona główna dla zalogowanego użytkownika

#### 4.3. Strona rejestracji

Strona rejestracji umożliwia użytkownikom utworzenie nowego konta w aplikacji. Pole *nazwa* musi zawierać unikalną nazwę użytkownika. Przy polu *email* występuje walidacja. Jeśli hasła nie są takie same, użytkownik dostaje stosowny komunikat. Sprawdzana jest również kompletność, wszystkie pola są wymagane. W przeciwnym razie wyświetlany jest komunikat o wpisaniu wszystkich pól. Całość rejestracji po stronie backend została stworzona przy użyciu Spring Security. Formularz rejestracji przedstawiony został na poniższym rysunku.

The image shows a registration form titled "Rejestracja" on a dark purple background. The form is a dark gray rounded rectangle containing four white input fields and a button. The labels "Nazwa:", "Email:", "Hasło:", and "Powtórz hasło:" are in white text to the left of their respective fields. The button at the bottom is dark gray with the text "Zarejestruj się" in white.

*Rysunek 14 Strona rejestracji*

#### 4.4. Strona logowania

Strona logowania umożliwia uwierzytelnienie użytkowników w aplikacji. W przypadku niepoprawnych danych użytkownik dostaje stosowny komunikat. Zalogowany użytkownik automatycznie zostaje przeniesiony na stronę główną. Logika logowania została zaimplementowana przy użyciu Spring Security. Wygląd okna logowania przedstawiony został poniżej.

The image shows a login form titled "Logowanie" on a dark purple background. The form is a dark gray rounded rectangle containing two white input fields and a button. The labels "Nazwa:" and "Hasło:" are in white text to the left of their respective fields. The button at the bottom is dark gray with the text "Zaloguj" in white.

## 4.5. Strona z listą plików

Na stronie wyświetlana jest lista wszystkich plików danego użytkownika. Aby zoptymalizować ładowanie danych, podczas pierwszego wejścia na stronę pobierane są jedynie podstawowe informacje o plikach, takie jak tytuł, album, rok i artysta. Szczegółowe dane, w tym okładka i plik audio, są dostępne dopiero po przejściu na stronę szczegółów wybranego pliku. Dla lepszej wydajności początkowo ładowanych jest 10 pierwszych utworów. Kolejne rekordy są automatycznie pobierane w momencie przewinięcia strony na dół, aż do załadowania wszystkich danych. Użytkownik ma możliwość filtrowania listy plików według tytułu, albumu, roku oraz artysty. Filtry można łączyć w dowolny sposób, co umożliwia bardziej precyzyjne wyszukiwanie. Mechanizm filtracji został zaimplementowany za pomocą klasy `AudioSpecification` (Listing 1), która korzysta z JPA Criteria API.

```
public class AudioSpecification {
    public static Specification<Audio> filterByCriteria(
        String title,
        Integer year,
        String artist,
        String album,
        Long userId
    ) {
        return (root, query, criteriaBuilder) -> {
            List<Predicate> predicates = new ArrayList<>();
            if (title != null && !title.isEmpty()) {
                predicates.add(criteriaBuilder.like(
                    criteriaBuilder.lower(root.get("title")),
                    "%" + title.toLowerCase() + "%"
                ));
            }
            if (year != null) {
                predicates.add(criteriaBuilder.equal(root.get("year"), year));
            }
            if (artist != null && !artist.isEmpty()) {
                predicates.add(criteriaBuilder.like(
```

```

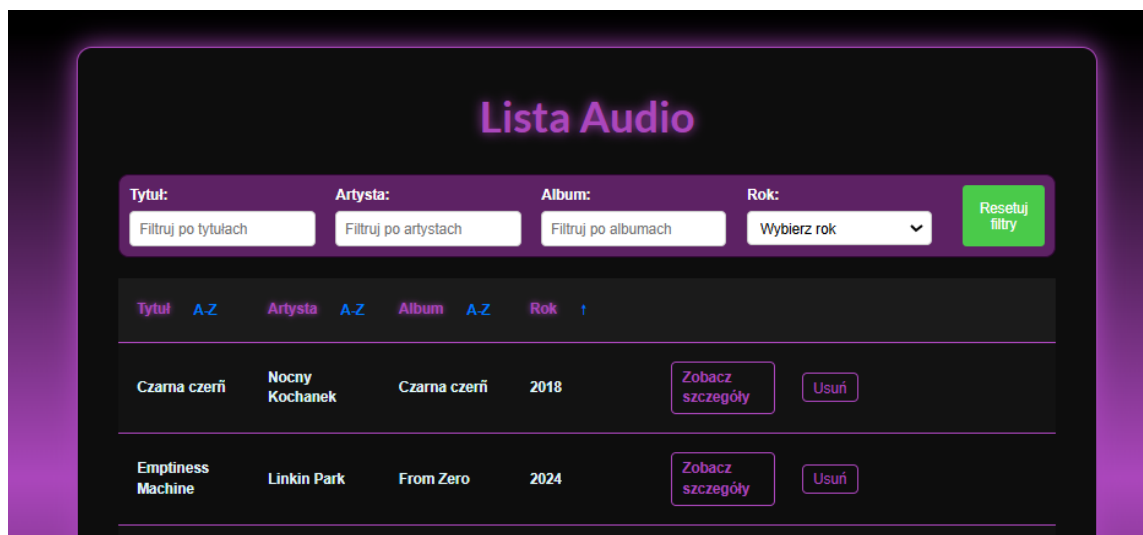
        criteriaBuilder.lower(root.get("artist")),
        "%" + artist.toLowerCase() + "%"
    ));
}
if (album != null && !album.isEmpty()) {
    predicates.add(criteriaBuilder.like(
        criteriaBuilder.lower(root.get("album")),
        "%" + album.toLowerCase() + "%"
    ));
}
if (userId != null) {
    predicates.add(criteriaBuilder.equal(root.get("user").get("id"),
    userId));
}
return criteriaBuilder.and(predicates.toArray(new
Predicate[0]));
};
}
}

```

*Listing 1*

Metoda pokazana na powyższym listingu przyjmuje argumenty jakie mają być filtrowane czyli: tytuł, artystę, rok, album oraz id użytkownika. Parametry te są wykorzystywane do budowy zapytania w zależności od tego, które zostały podane przez użytkownika. Obiekty *predicates* tworzą warunki do zapytania SQL. Wszystkie warunki łączone są przy użyciu metody *criteriaBuilder.and*, w wyniku czego zwracane są wyniki zgodne z oczekiwaniami. Kolejną funkcją jest sortowanie. Użytkownik może sortować utwory alfabetycznie (w przypadku tytułu, artysty i albumu) lub według roku w porządku rosnącym lub malejącym. Logika sortowania została zaimplementowana po stronie frontendu, co pozwala na szybkie i płynne działanie aplikacji.

Przy każdym utworze istnieje przycisk do usunięcia go oraz do zobaczenia jego szczegółów, który przenosi na zakładkę związaną z danym utworem. Całość tej strony została przedstawiona na poniższym rysunku:



Rysunek 16 Lista Audio


#### 4.6. Strona szczegółów plików audio

Strona szczegółów pliku audio wyświetla pełne metadane dotyczące wybranego utworu, umożliwiając użytkownikowi dostęp do wszystkich informacji związanych z plikiem. Na stronie znajdują się dane takie jak: tytuł, artysta, album, rok, gatunek, wydawca, numer utworu, kompozytor, autor, a także okładka albumu. W celu podglądu wizualnego na stronie wyświetlana jest okładka albumu, z którego pochodzi utwór. Użytkownik ma możliwość pobrania pliku audio na swoje urządzenie za pomocą dedykowanego przycisku, który umożliwia ściągnięcie utworu w oryginalnym formacie. Dzięki wbudowanemu odtwarzaczowi użytkownik ma możliwość odsłuchania pliku audio bezpośrednio na stronie, zachowując przy tym pełną kontrolę nad zatrzymywaniem czy przewijaniem. Istnieje także przycisk, który pozwala użytkownikowi powrócić do listy wszystkich plików audio, co pozwala na szybkie przejście bez konieczności nawigowania po pasku.

← Powrót do listy

## Emptiness Machine

Artysta: Linkin Park	Numer utworu: 12
Album: From Zero	Rok: 2024
Gatunek: Metal	Komentarz:
Tekst:	Kompozytor: Linkin Park
Wydawca: Sony	Artysta oryginalny: Linkin Park
Artysta albumu: Linkin Park	Prawa autorskie: Linkin Park
URL:	Format kodowania:



0:00 / 3:20

🔊 ⋮

Pobierz plik audio

Rysunek 17 Strona szczegółów pliku

#### 4.7. Strona dodawania nowego pliku audio

Strona dodawania nowego pliku pozwala użytkownikowi na przesłanie pliku audio oraz samodzielne wprowadzenie lub edycję metadanych. Przy jej tworzeniu zastosowano minimalistyczny interfejs umożliwiający intuicyjne dodawanie utworu. W celu przesłania pliku użytkownik musi podać tytuł oraz wybrać ów plik. Opcjonalnie może dodać okładkę albumu jako plik graficzny. Pozostałe pola również są opcjonalne, przy czym pole roku przyjmuje wyłącznie liczby, co zapobiega wprowadzeniu błędnych danych. Natomiast pole gatunek jest w formie listy rozwijanej, aby zapewnić pełną zgodność z tagami ID3v2. Dane z formularza

przesyłane są w formacie FormData do backendu, gdzie kod odpowiedzialny za przetworzenie pliku przedstawiony został na listingu 2.

```
@Operation(security = {@SecurityRequirement(name =
BASIC_AUTH_SECURITY_SCHEME)})
@ResponseStatus(HttpStatus.CREATED)
@PostMapping(consumes = {"multipart/form-data"}, path = "/add")
public AudioDto createAudio(@Valid @ModelAttribute
CreateAudioRequest createAudioRequest) {
    Long userId = createAudioRequest.getUserId();
    Optional<User> optionalUser = userRepository.findById(userId);
    User user = optionalUser.get();
    Audio audio = audioMapper.toAudio(createAudioRequest, user);
    return audioMapper.toAudioDto(audioService.saveAudio(audio));
}
```

*Listing 2*

Metoda *createAudio* przedstawiona na powyższym listingu obsługuje proces dodawania nowego pliku audio do aplikacji. Jest częścią kontrolera odpowiedzialnego za zarządzanie zasobami audio i realizuje operacje na danych przesłanych przez użytkownika w formularzu. Na podstawie przekazanego ID użytkownika metoda sprawdza, czy użytkownik istnieje w bazie danych. Jeśli użytkownik nie zostanie znaleziony, aplikacja zgłasza błąd. Dane z formularza, takie jak plik, okładka i metadane, są przekształcane na obiekt *Audio*, który może zostać zapisany w bazie danych. Wykorzystano tutaj wzorzec mapowania, który pozwala na przekształcenie danych z formularza użytkownika na obiekt typu *Audio*, który jest wykorzystywany w backendzie. Obiekt audio jest zapisywany w bazie danych za pomocą serwisu. Po pomyślnym dodaniu pliku metoda zwraca dane o nowo utworzonym pliku w formacie DTO. W razie wystąpienia błędów lub poprawnego przesłania pliku, użytkownik zostanie poinformowany odpowiednim komunikatem. Wygląd formularza przedstawiony został na poniższym rysunku.



*Rysunek 18 Strona z dodawaniem nowego pliku*

#### 4.8. Strona wgrywania nowego pliku z jego metadanymi

Strona „Wgraj nowy plik” umożliwia użytkownikowi łatwe dodawanie plików audio do aplikacji. Jest zaprojektowana w taki sposób, aby wspierać zarówno wgrywanie pojedynczych plików, jak i całych folderów. Dzięki temu użytkownik może efektywnie zarządzać większą ilością danych audio bez konieczności wykonywania powtarzalnych operacji. Strona zawiera informacje o tym, jakie operacje można na niej wykonać. Użytkownik może wybrać folder zawierający pliki audio, w takim przypadku aplikacja przeszukuje cały folder w poszukiwaniu plików audio, wszystkie takie pliki wysyłane są do backendu. Funkcja odczytująca metadane znajduje się na listingu 3.

```

@Operation(security = {@SecurityRequirement(name =
    BASIC_AUTH_SECURITY_SCHEME)})
@ResponseStatus(HttpStatus.CREATED)
@PostMapping(consumes = {"multipart/form-data"}, path = "/upload")
public AudioDto uploadAudio(@Valid @ModelAttribute
    UploadAudioRequest uploadAudioRequest) {
    Long userId = uploadAudioRequest.getUserId();
    User user = userRepository.findById(userId)
        .orElseThrow(() -> new RuntimeException("User not
found"));
    Audio audio = new Audio();
    MultipartFile audioFile = uploadAudioRequest.getAudioFile();
    if (!audioFile.isEmpty()) {
        audio = audioMapper.toUploadAudio(uploadAudioRequest,
user);
        File tempFile = null;
        try {
            String originalFilename =
audioFile.getOriginalFilename();
            String fileExtension = originalFilename != null ?
originalFilename.substring(originalFilename.lastIndexOf(".")) :
".audio";
            tempFile = File.createTempFile("audio_",
fileExtension);
            audioFile.transferTo(tempFile);
            org.jaudiotagger.audio.AudioFile jaudioFile =
org.jaudiotagger.audio.AudioFileIO.read(tempFile);
            org.jaudiotagger.tag.Tag tag = jaudioFile.getTag();
            if (tag != null) {
audio.setTitle(tag.getFirst(org.jaudiotagger.tag.FieldKey.TITLE));
audio.setArtist(tag.getFirst(org.jaudiotagger.tag.FieldKey.ARTIST)
);
audio.setTrack(tag.getFirst(org.jaudiotagger.tag.FieldKey.TRACK));
audio.setAlbum(tag.getFirst(org.jaudiotagger.tag.FieldKey.ALBUM));
                String yearString =
tag.getFirst(org.jaudiotagger.tag.FieldKey.YEAR);
                if (!yearString.isEmpty()) {
                    audio.setYear(Integer.parseInt(yearString));
                }
            }
            audio.setGenreDescription(tag.getFirst(org.jaudiotagger.tag.FieldK
ey.GENRE));

```

```

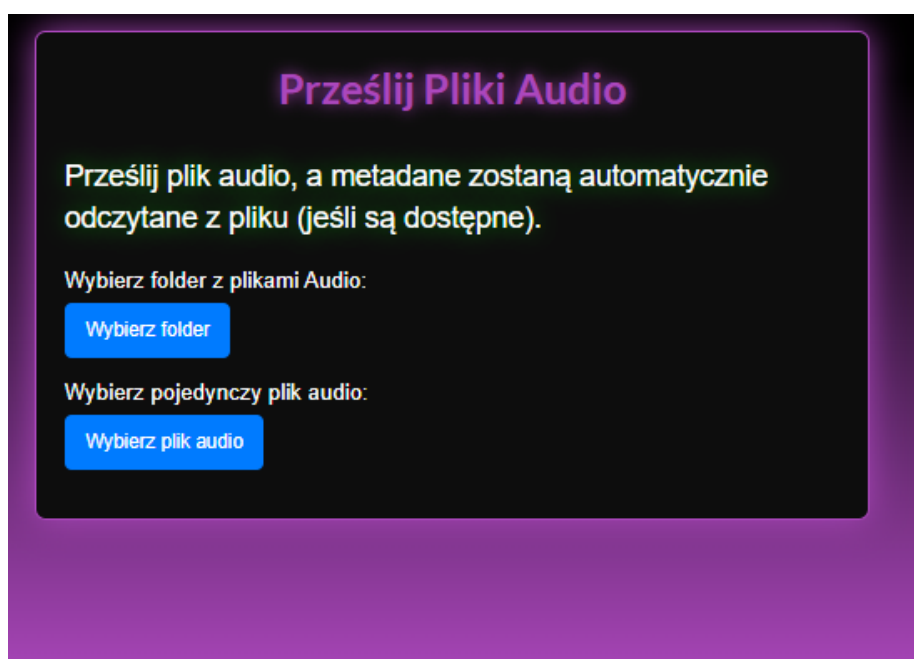
audio.setComment(tag.getFirst(org.jaudiotagger.tag.FieldKey.COMMENT));
audio.setLyrics(tag.getFirst(org.jaudiotagger.tag.FieldKey.LYRICS));
audio.setComposer(tag.getFirst(org.jaudiotagger.tag.FieldKey.COMPOSER));
audio.setOriginalArtist(tag.getFirst(org.jaudiotagger.tag.FieldKey.ORIGINAL_ARTIST));
audio.setAlbumArtist(tag.getFirst(org.jaudiotagger.tag.FieldKey.ALBUM_ARTIST));
audio.setUrl(tag.getFirst(org.jaudiotagger.tag.FieldKey.URL_OFFICIAL_RELEASE_SITE));
audio.setEncoder(tag.getFirst(org.jaudiotagger.tag.FieldKey.ENCODER));

        List<org.jaudiotagger.tag.images.Artwork>
artworkList = tag.getArtworkList();
        if (!artworkList.isEmpty()) {
            org.jaudiotagger.tag.images.Artwork artwork =
artworkList.get(0);
            byte[] imageData = artwork.getBinaryData();
            if (imageData != null) {
                audio.setCoverArt(imageData);
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
        throw new RuntimeException("Nie udało się odczytać
metadanych", e);
    } finally {
        if (tempFile != null && tempFile.exists()) {
            tempFile.delete();
        }
    }
    }
    audio.setUser(user);
    return audioMapper.toAudioDto(audioService.saveAudio(audio));
}

```

*Listing 3*

Metoda przedstawiona na powyższym listingu odpowiada za obsługę wgrywania plików audio oraz automatycznie odczytywanie ich metadanych. Po otrzymaniu żądania typu POST, metoda *uploadAudio* weryfikuje, czy przesłano poprawne dane użytkownika na podstawie identyfikatora *userId*. Jeśli użytkownik istnieje w bazie danych, tworzony jest obiekt Audio, który będzie przechowywał szczegóły dotyczące pliku. Następnie aplikacja przetwarza przesłany plik audio, który jest tymczasowo zapisywany na serwerze w celu odczytania jego metadanych za pomocą biblioteki JAudiotagger. Biblioteka ta pozwala na automatyczne odczytanie danych takich jak tytuł, artysta, album, rok, gatunek, kompozytor, numer ścieżki, tekst utworu, komentarze, czy URL powiązany z utworem. Dodatkowo, jeśli plik zawiera okładkę, jest ona odczytywana i przypisywana do pola *coverArt* obiektu Audio. Po zakończeniu odczytu tymczasowy plik jest usuwany z serwera, aby nie zajmować nadmiernie przestrzeni dyskowej. Na koniec wszystkie dane, wraz z informacją o użytkowniku, są zapisywane w bazie danych. Po zakończeniu procesu użytkownik otrzymuje powiadomienie o sukcesie wgrywania lub o ewentualnych błędach. Wygląd zakładki przedstawiony został poniżej.



Rysunek 19 Strona z wgrywaniem nowego pliku wraz z jego metadanymi

## 4.9. Strona wgrywania nowego pliku z wyszukiwaniem jego metadanych

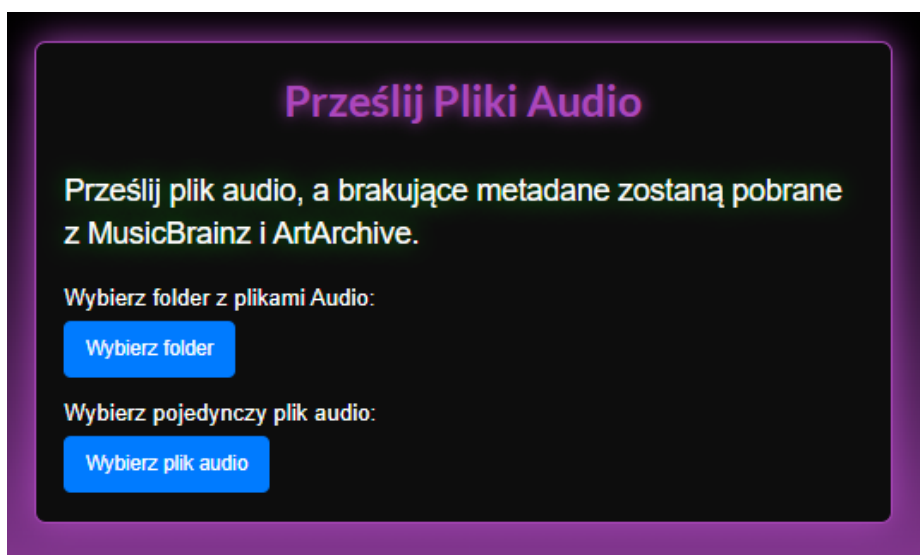
Strona „Wyszukaj dane pliku” umożliwia użytkownikowi wyszukiwanie i automatyczne uzupełnianie metadanych dla przesłanych plików audio. Na wstępie użytkownik jest informowany o funkcjach strony, która pozwala na przesłanie pojedynczego pliku audio lub całego folderu zawierającego wiele plików. Po dokonaniu wyboru i przesłaniu danych, aplikacja automatycznie przystępuje do analizy i wyszukiwania metadanych. Połączenie z MusicBrainz odbywa się poprzez funkcję *getRecordingInfoJson*, co widać na poniższym listingu.

```
public RecordingsResponse getRecordingInfoJson(String title,
String artist) {
    String query = String.format("recording:\"%s\" AND
artist:\"%s\"", title, artist);
    String url = "https://musicbrainz.org/ws/2/recording/?query="
+ query + "&fmt=json&limit=1";
    String json = restTemplate.getForObject(url, String.class);
    try {
        return objectMapper.readValue(json,
RecordingsResponse.class);
    } catch (Exception e) {
        throw new RuntimeException();
    }
}
```

*Listing 4*

Działanie funkcji polega na wysyłaniu zapytań do API MusicBrainz, aby odnaleźć informacje o nagraniach na podstawie podanego tytułu utworu oraz artysty. Na początku funkcja formatuje zapytanie, które zawiera oba parametry: tytuł oraz nazwę artysty, aby zapewnić precyzyjne wyniki. Następnie tworzy odpowiedni adres URL z parametrami zapytania i formatuje odpowiedź w formacie JSON. Korzystając z obiektu *RestTemplate*, funkcja wysyła żądanie HTTP GET do wygenerowanego

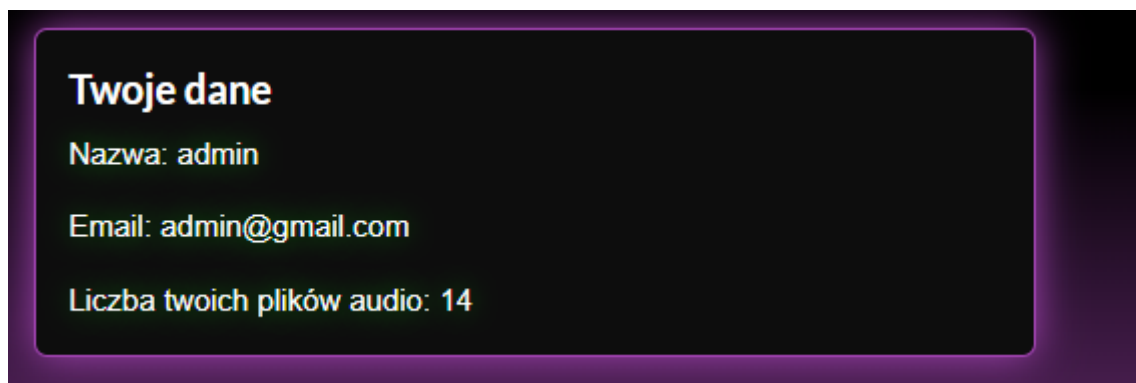
URL-a i pobiera odpowiedź w formie ciągu JSON. Za pomocą *ObjectMapper* z biblioteki Jackson konwertuje dane JSON na obiekt typu *RecordingsResponse*, który reprezentuje odpowiedź z MusicBrainz w strukturze przyjaznej dla aplikacji. Pobieranie okładki odbywa się poprzez uzyskanie z MusicBrainz ID utworu, a następnie wysłanie wraz z tym ID zapytania do strony CovertArtArchive. Całość strony widać na zdjęciu poniżej.



Rysunek 20 Strona z wgrywaniem pliku wraz z szukaniem jego metadanych

#### 4.10. Strona profilu

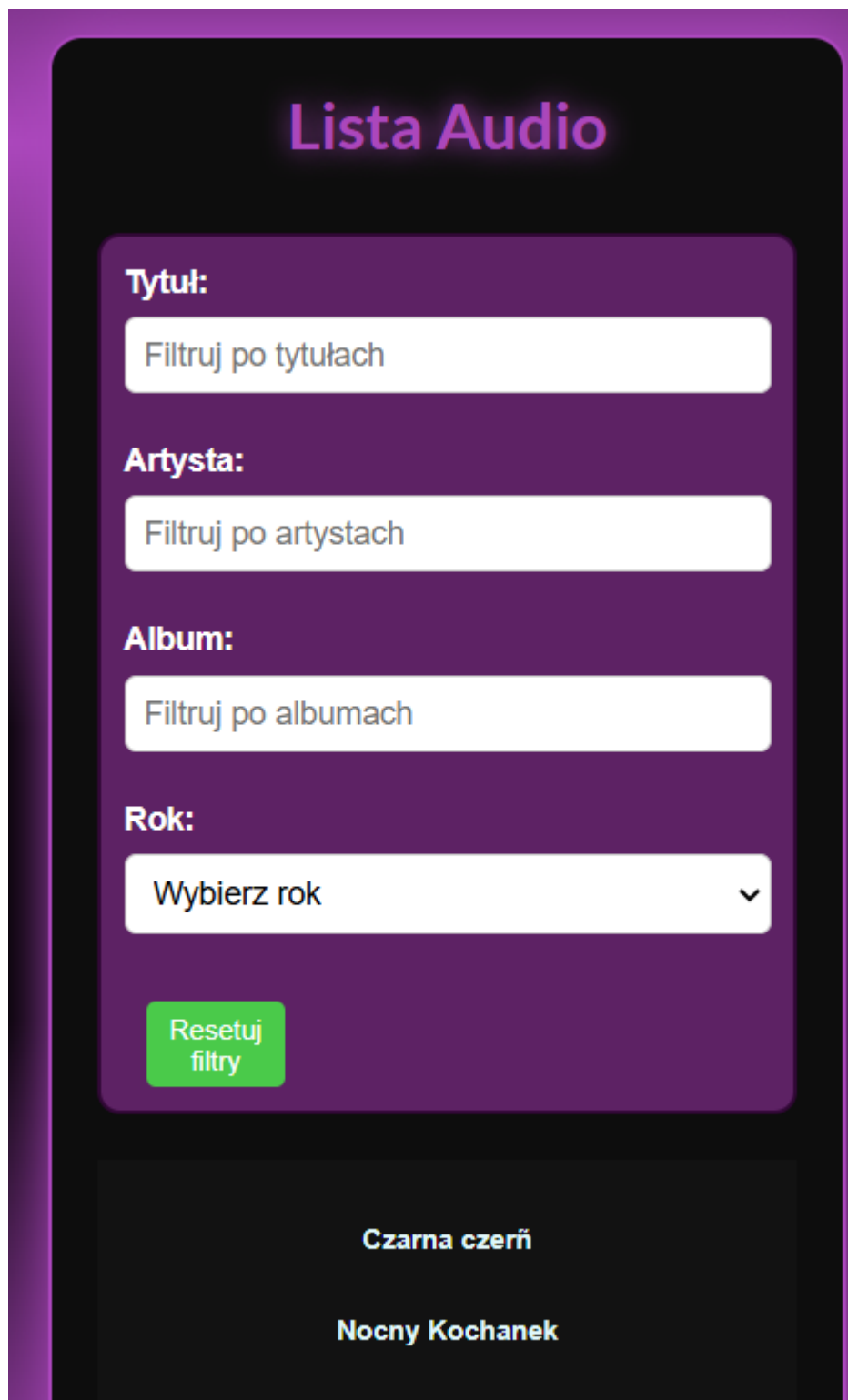
Strona profilu użytkownika pozwala na przeglądanie podstawowych informacji związanych z danym kontem. Na tej stronie użytkownik może zobaczyć swoje dane, takie jak nazwa użytkownika oraz adres email, które są wyświetlane w sposób przejrzysty i dostępny. Dodatkowo, użytkownik otrzymuje informację o liczbie swoich plików audio, co daje mu wgląd w to, jaką ilość zawartości dodał do systemu. Wszystkie te dane są pobierane bezpośrednio z bazy danych, a strona jest dynamicznie generowana po zalogowaniu, zapewniając dostęp do informacji powiązanych z konkretnym użytkownikiem. Przykład strony pokazany jest na rysunku poniżej.



*Rysunek 21 Strona profilu*

#### 4.11. Wygląd aplikacji na urządzeniu mobilnym

W celu zapewnienia pełnej responsywności oraz wygody korzystania z aplikacji na urządzeniach mobilnych, projekt został dostosowany do specyfiki ekranów o mniejszych rozmiarach. Głównym celem było zachowanie czytelności oraz intuicyjności interfejsu, niezależnie od wielkości urządzenia. W widoku mobilnym aplikacja wykorzystuje układ listowy, w którym wszystkie elementy układają się pionowo, jeden pod drugim. Taki układ pozwala na nawigację za pomocą przewijanie ekranu, co jest standardowym sposobem obsługi na urządzeniach mobilnych. Wygląd aplikacji na urządzeniu mobilnym został przedstawiony na poniższym zdjęciu.



Rysunek 22 Wygląd aplikacji na urządzeniu mobilnym



## Podsumowanie

Celem pracy było zaprojektowanie i implementacja aplikacji webowej umożliwiającej efektywne zarządzanie plikami audio. Aplikacja została stworzona z myślą o użytkownikach, którzy chcą przechowywać, organizować i przeglądać swoje pliki dźwiękowe w prosty i intuicyjny sposób. Proces przesyłania plików oraz zarządzania ich metadanymi, takimi jak artysta, album czy gatunek, odbywa się w sposób elastyczny, umożliwiając użytkownikowi zarówno manualne wprowadzanie danych, jak i automatyczne ich pobieranie z plików audio oraz z zewnętrznych źródeł, takich jak baza danych MusicBrainz. Aplikacja została zaprojektowana z wykorzystaniem nowoczesnych technologii, takich jak Java Spring Boot na backendzie oraz React na frontendzie, co zapewnia jej wysoką wydajność, skalowalność i bezpieczeństwo. Baza danych PostgreSQL umożliwia trwałe i efektywne przechowywanie informacji o plikach audio i ich metadanych. Dzięki zastosowaniu zasad Nielsena oraz wytycznych projektowania interfejsów użytkownika zgodnych z Material Design, aplikacja cechuje się atrakcyjnym wyglądem oraz intuicyjnością obsługi. Projekt ten stanowi solidną bazę do dalszego rozwoju funkcji zarządzania plikami audio. W przyszłości można rozważyć integrację z dodatkowymi źródłami danych, takimi jak Spotify. Stworzenie komunikacji pomiędzy użytkownikami pozwoli na wymianę utworów między nimi oraz wdrożenie systemu rekomendacji, który na podstawie preferencji użytkownika oraz analizy jego kolekcji sugerowałby nowe utwory do odkrycia. W przyszłości możliwe jest zaimplementowanie sztucznej inteligencji, która na podstawie samej próbki dźwięku rozpoznawałaby utwór oraz automatycznie znajdowała i uzupełniała związane z nim metadane.



## Spis literatury

- [1] „Libib”, <https://www.libib.com/>, Data dostępu 04.09.2024
- [2] „RateYourMusic”, <https://rateyourmusic.com/>, Data dostępu 04.09.2024
- [3] „MediaMonkey”, <https://www.mediamonkey.com/>, Data dostępu 04.09.2024
- [4] „Helium Music Manager”, <https://helium-music-manager.softonic.pl/>, Data dostępu 04.09.2024
- [5] „Get Music Bee”, <https://www.getmusicbee.com/>, Data dostępu 05.09.2024
- [6] „Foobar2000”, <https://www.foobar2000.org/>, Data dostępu 05.09.2024
- [7] „Java”, <https://www.java.com/pl/>, Data dostępu 20.11.2024
- [8] „Spring”, <https://spring.io/>, Data dostępu 20.11.2024
- [9] „Spring Security”, <https://spring.io/projects/spring-security>, Data dostępu 20.11.2024
- [10] „React”, <https://react.dev/>, Data dostępu 20.11.2024
- [11] „PostgreSQL”, <https://www.postgresql.org/>, Data dostępu 20.11.2024
- [12] „MusicBrainz”, <https://musicbrainz.org/>, Data dostępu 20.11.2024
- [13] „ID3v2”, <https://pl.wikipedia.org/wiki/ID3>, Data dostępu 20.11.2024
- [14] „Covert Art Archive”, <https://coverartarchive.org/>, Data dostępu 20.11.2024
- [15] „Zasady Nielsena”, <https://www.nngroup.com/articles/ten-usability-heuristics/>, Data dostępu 21.01.2025
- [16] „Material Design”, <https://m3.material.io/>, Data dostępu 21.01.2025