Dokument Wymagań Produktu (PRD) – HardbanRecords Lab

1.0 Wprowadzenie i Wizja

1.1. Streszczenie Wykonawcze

HardbanRecords Lab to wyspecjalizowana platforma wydawnicza, zaprojektowana w celu wzmocnienia pozycji niezależnych muzyków i autorów. Dostarcza zunifikowany, kompleksowy ekosystem do tworzenia, dystrybucji, monetyzacji i analizy treści cyfrowych, eliminując tradycyjnych pośredników. Platforma umożliwia twórcom pełną kontrolę nad ich dziełami, od momentu powstania pomysłu aż po globalną dystrybucję i sprzedaż bezpośrednią. Niniejszy dokument określa wymagania dla Minimalnie Opłacalnego Produktu (MVP) platformy, koncentrując się strategicznie na dwóch kluczowych filarach: Wydawnictwie Muzycznym (Music Publishing) oraz Wydawnictwie Cyfrowym (Digital Book Publishing). Celowe pominięcie modułu e-learningowego [1] jest decyzją strategiczną, mającą na celu zapewnienie najwyższej jakości i dopracowania podstawowej propozycji wartości dla docelowych grup użytkowników.

1.2. Wizja Strategiczna

Wizją HardbanRecords Lab jest stanie się domyślną platformą dla niezależnych twórców, którzy cenią sobie autonomię, pełną kontrolę nad swoją własnością intelektualną oraz dostęp do nowoczesnych, zautomatyzowanych narzędzi. Filozofia produktu opiera się na haśle: „Zero pośredników, pełna kontrola kreatora”.[1] Platforma ma na celu zdemokratyzowanie procesu wydawniczego, czyniąc go dostępnym, przejrzystym i dochodowym dla każdego artysty i autora, niezależnie od skali jego

działalności. Dążymy do zbudowania ekosystemu, w którym technologia służy kreatywności, a nie ją ogranicza, automatyzując złożone procesy techniczne i pozwalając twórcom skupić się na tym, co najważniejsze – tworzeniu.

1.3. Cele Produktu i Mierniki Sukcesu

Aby zrealizować naszą wizję, zdefiniowano trzy główne cele strategiczne dla produktu, wraz z kluczowymi wskaźnikami efektywności (KPI), które pozwolą mierzyć postępy w ich osiąganiu.

- **Cel 1: Wzmocnienie Procesu Tworzenia i Dystrybucji:** Umożliwienie twórcom płynnego i intuicyjnego przepływu pracy, od przesłania gotowego materiału (utworu muzycznego, manuskryptu książki) aż po jego skuteczną dystrybucję na kluczowe globalne platformy.
- **Cel 2: Maksymalizacja Przychodów Twórcy:** Dostarczenie narzędzi do sprzedaży bezpośredniej (direct-to-fan) oraz zapewnienie przejrzystego i sprawiedliwego systemu rozliczania tantiem i przychodów z różnych źródeł.
- **Cel 3: Uproszczenie Barier Technicznych:** Zintegrowanie złożonych procesów, takich jak generowanie metadanych, konwersja formatów plików, przygotowywanie pakietów dystrybucyjnych i automatyzacja marketingu, w ramach prostego i przyjaznego interfejsu użytkownika.

Poniższa tabela przedstawia kluczowe wskaźniki sukcesu (KPI) i ich cele na pierwsze 12 miesięcy działalności platformy, bazując na wstępnych założeniach.[1]

Wskaźnik	Cel w ciągu 12 miesięcy
Liczba aktywnych twórców	5 000
Liczba opublikowanych produktów (albumów/książek)	2 500
Przychody przetworzone dla twórców	250 000 PLN
Wskaźnik konwersji (rejestracja -> publikacja)	> 15%
Średni czas od rejestracji do pierwszej	< 7 dni

publikacji	

2.0 Grupa Docelowa i Podróże Użytkownika

2.1. Persony Docelowe

Aby zapewnić, że produkt odpowiada na realne potrzeby rynku, zdefiniowano dwie kluczowe persony reprezentujące główne segmenty docelowych użytkowników.[1]

- **Persona 1: Anna, Niezależna Autorka (43 lata)**
  - **Kim jest:** Anna pisze i wydaje e-booki poradnikowe skierowane do kobiet po 40. roku życia. Posiada gotowe treści w formacie DOCX i korzysta z narzędzi takich jak Canva do tworzenia prostych okładek.
  - **Potrzeby i cele:** Potrzebuje prostego, zautomatyzowanego sposobu na sformatowanie swojej książki do standardów e-bookowych (EPUB, MOBI), wygenerowanie profesjonalnej okładki i dystrybucję na kluczowe platformy, zwłaszcza Amazon KDP. Chce również sprzedawać swoje książki bezpośrednio swojej społeczności, którą buduje poprzez newsletter.
  - **Bariery (Pain Points):** Złożoność techniczna procesu wydawniczego, brak czasu na naukę skomplikowanego oprogramowania, trudności z marketingiem i promocją.
- **Persona 2: Tomasz, Producent Muzyczny (35 lat)**
  - **Kim jest:** Tomasz tworzy w domowym studiu muzykę z gatunków ambient i techno. Posiada gotowe utwory w formacie WAV i jest zaznajomiony z podstawami technologii, ale jego czas jest ograniczony.
  - **Potrzeby i cele:** Szuka darmowego lub niskokosztowego sposobu na dystrybucję swojej muzyki na główne serwisy streamingowe, takie jak Spotify i Apple Music. Chce również sprzedawać swoje pakiety sampli (sample packs) i beaty bezpośrednio innym producentom.
  - **Bariery (Pain Points):** Wysokie koszty i skomplikowane modele biznesowe istniejących dystrybutorów, czasochłonne przygotowywanie metadanych i pakietów dystrybucyjnych, brak narzędzi do łatwego zarządzania swoim

katalogiem.

2.2. Kluczowe Podróże Użytkownika

Poniższe scenariusze opisują, w jaki sposób zdefiniowane persony będą wchodzić w interakcję z platformą HardbanRecords Lab, aby osiągnąć swoje cele. Te podróże łączą wymagania funkcjonalne z realnymi przypadkami użycia, ilustrując wartość dostarczaną przez platformę.[1]

- **Podróż 1: Anna publikuje swojego pierwszego e-booka**
  1. **Rejestracja i Wdrożenie:** Anna zakłada darmowe konto na platformie, wybierając rolę „Autorka książek" (book_author). Po zalogowaniu zostaje przekierowana do swojego panelu, gdzie przejrzysty interfejs prowadzi ją przez kolejne kroki.
  2. **Import Treści:** Anna klika przycisk „Dodaj nową książkę" i importuje swój manuskrypt w formacie DOCX bezpośrednio do systemu.[1]
  3. **Tworzenie i Formatowanie:** Platforma automatycznie konwertuje tekst. Anna korzysta z wbudowanego narzędzia AI do generowania okładki, wpisując prosty opis tekstowy (text prompt).[1] Następnie uzupełnia podstawowe metadane, takie jak tytuł, opis i numer ISBN.
  4. **Dystrybucja:** W sekcji dystrybucji Anna zaznacza interesujące ją kanały, w tym Amazon KDP i Draft2Digital.[1] Jednym kliknięciem inicjuje proces eksportu, a system generuje pakiety (pliki e-book w formatach EPUB, MOBI, PDF oraz pliki z metadanymi) gotowe do wgrania na wybrane platformy.[1]
  5. **Monetyzacja i Analiza:** Równocześnie z dystrybucją, system automatycznie tworzy stronę produktu w zintegrowanym sklepie WooCommerce, umożliwiając Annie sprzedaż bezpośrednią.[1] W swoim panelu Anna może śledzić statystyki sprzedaży i pobrań ze wszystkich kanałów.
- **Podróż 2: Tomasz wydaje nowy singiel**
  1. **Rejestracja i Wdrożenie:** Tomasz rejestruje się na platformie jako „Twórca muzyki" (music_creator). Po zalogowaniu trafia do panelu muzycznego.
  2. **Przesyłanie Materiałów:** Tomasz tworzy nowe „wydanie muzyczne" i przesyła swój utwór w formacie WAV oraz grafikę okładki. Pliki są bezpiecznie przechowywane na serwerze Amazon S3.[1]
  3. **Zarządzanie Metadanymi:** W prostym formularzu uzupełnia kluczowe metadane: tytuł utworu, gatunek, a system automatycznie generuje dla niego kod ISRC.

4. **Automatyzacja Dystrybucji:** Tomasz chce skorzystać z dystrybutora RouteNote. W panelu HardbanRecords Lab klika przycisk „Eksportuj dla RouteNote". System automatycznie tworzy archiwum ZIP zawierające plik audio oraz plik CSV z metadanymi, w pełni zgodny ze standardem wymaganym przez RouteNote.[1] Proces, który wcześniej zajmował mu kilkadziesiąt minut, teraz trwa kilka sekund.
5. **Promocja i Analiza:** Platforma generuje dla niego inteligentny link promocyjny (smartlink) z opcją pre-save. Po premierze, w swoim panelu, Tomasz może monitorować statystyki odsłuchań i przychody (w oparciu o przyszłą integrację z raportami dystrybutorów).[1]

3.0 Główne Moduły Platformy: Wymagania Funkcjonalne

Ta sekcja definiuje konkretne wymagania funkcjonalne dla kluczowych modułów platformy, syntetyzując ogólne cele z dokumentów strategicznych [1] ze szczegółowymi punktami końcowymi API i logiką biznesową opisaną w dokumentacji technicznej.[1]

3.1. Zunifikowany Panel Twórcy

Panel główny jest spersonalizowanym centrum dowodzenia dla każdego użytkownika. Jego zawartość i dostępne opcje są dynamicznie dostosowywane w oparciu o przypisaną użytkownikowi rolę (music_creator, book_author, admin), co jest kluczowym założeniem architektury opartej na rolach.[1]

- **FR-DASH-01:** Po zalogowaniu użytkownik musi być automatycznie przekierowany do swojego spersonalizowanego panelu.
- **FR-DASH-02:** Panel musi wyświetlać widżety podsumowujące kluczowe statystyki (np. sprzedaż, odsłuchania), listę ostatnio dodanych projektów oraz przyciski szybkiego dostępu do głównych akcji (np. „Dodaj książkę", „Dodaj utwór").[1]
- **FR-DASH-03:** Nawigacja główna musi dynamicznie ukrywać lub pokazywać linki do modułów (Wydawnictwo Muzyczne, Wydawnictwo Cyfrowe) w zależności od roli użytkownika.

3.2. Moduł Wydawnictwa Muzycznego (Music Publishing)

- **FR-MUS-01: Zarządzanie Wydaniami:** Użytkownicy muszą mieć możliwość tworzenia, przeglądania, aktualizowania i usuwania swoich wydań muzycznych (singli, EP, albumów). Funkcjonalność ta odpowiada bezpośrednio zdefiniowanym punktom końcowym CRUD API: POST /music/, GET /music/list, GET /music/{id}, PUT /music/{id}, DELETE /music/{id}.[1]
- **FR-MUS-02: Przesyłanie Zasobów:** System musi akceptować pliki audio w wysokiej jakości (preferowany format WAV, akceptowany również MP3) oraz pliki graficzne na okładki. Wszystkie pliki muszą być bezpiecznie przechowywane w dedykowanym zasobniku (bucket) Amazon S3.[1]
- **FR-MUS-03: Zarządzanie Metadanymi:** Interfejs użytkownika musi umożliwiać wprowadzenie wszystkich niezbędnych metadanych, w tym tytułu, wykonawcy, gatunku, kodów ISRC/UPC oraz informacji o podziale tantiem (split royalties).
- **FR-MUS-04: Zautomatyzowane Pakowanie Dystrybucyjne:** System musi udostępniać funkcję jednego kliknięcia (one-click), która wygeneruje i udostępni do pobrania pakiet dystrybucyjny w formacie ZIP. Pakiet ten musi zawierać pliki audio oraz plik metadanych CSV, sformatowany zgodnie ze specyfikacją dystrybutora RouteNote. Funkcjonalność ta jest realizowana przez punkt końcowy POST /music/{id}/export-route-note.[1]
- **FR-MUS-05: Analityka:** Panel użytkownika musi wyświetlać statystyki dla poszczególnych wydań. W fazie MVP będą to dane wprowadzane ręcznie lub dane zastępcze, z docelową integracją z raportami z serwisów streamingowych poprzez punkt końcowy GET /music/{id}/stats.[1]

3.3. Moduł Wydawnictwa Cyfrowego (Digital Book Publishing)

- **FR-BOOK-01: Zarządzanie Projektami Książkowymi:** Użytkownicy muszą mieć możliwość tworzenia, przeglądania, aktualizowania i usuwania swoich projektów książkowych. Funkcjonalność ta jest obsługiwana przez punkty końcowe CRUD API: POST /books/, GET /books/list, GET /books/{id}, PUT /books/{id}, DELETE /books/{id}.[1]
- **FR-BOOK-02: Import Treści:** System musi pozwalać użytkownikom na import istniejących manuskryptów w popularnych formatach, takich jak EPUB, PDF i

DOCX. Logika ta jest realizowana przez punkt końcowy POST /books/{id}/import.[1]

- **FR-BOOK-03: Generowanie Okładek z Wykorzystaniem AI:** Platforma musi zintegrować się z zewnętrznym serwisem generowania obrazów AI (np. poprzez API Hugging Face), aby umożliwić użytkownikom tworzenie unikalnych okładek na podstawie opisów tekstowych. Funkcja ta będzie wywoływana przez punkt końcowy POST /books/{id}/generate-cover.[1]
- **FR-BOOK-04: Eksport do Wielu Formatów:** System musi być w stanie konwertować treść książki do standardowych formatów e-booków: PDF, EPUB i MOBI. Proces ten będzie wykorzystywał biblioteki takie jak WeasyPrint i ebooklib i będzie uruchamiany przez punkt końcowy POST /books/{id}/export.[1]
- **FR-BOOK-05: Dystrybucja i Metadane:** System musi generować pliki metadanych (w formatach CSV, JSON) kompatybilne z wymaganiami głównych dystrybutorów, takich jak Amazon KDP i Draft2Digital, aby uprościć proces publikacji.[1]
- **FR-BOOK-06: Analityka:** Panel autora musi prezentować statystyki dotyczące jego książek, w tym liczbę pobrań i dane o sprzedaży, pobierane przez punkt końcowy GET /books/{id}/stats.[1]

4.0 Integracje Ekosystemowe i Automatyzacja

Sukces platformy zależy nie tylko od jej wewnętrznych funkcji, ale także od jej zdolności do płynnej integracji z szerszym ekosystemem narzędzi i usług, z których korzystają twórcy.

4.1. Sieć Partnerów Dystrybucyjnych

Platforma ma na celu ułatwienie dystrybucji treści do szerokiej gamy partnerów. Poniższa tabela określa zakres wsparcia dla poszczególnych kanałów w fazie MVP oraz w dalszym rozwoju, bazując na liście partnerów zidentyfikowanej w materiałach strategicznych.[1]

Partner Dystrybucyjny	Typ Treści	Status MVP

RouteNote	Muzyka	**Cel MVP** (poprzez eksport ZIP+CSV)
Amazon KDP	Książki	**Cel MVP** (poprzez eksport pakietu)
Draft2Digital	Książki	**Cel MVP** (poprzez eksport pakietu)
Spotify, Apple Music, etc.	Muzyka	Integracja w przyszłości (przez API)
Google Play Books, Apple Books	Książki	Integracja w przyszłości (przez API)
Pozostali (Smashwords, Kobo, etc.)	Książki	Integracja w przyszłości

4.2. E-commerce i Sprzedaż Bezpośrednia (WooCommerce)

Integracja ze sklepem internetowym jest kluczowa dla umożliwienia twórcom monetyzacji ich pracy bezpośrednio. Wybór WooCommerce jako platformy e-commerce jest podyktowany jego elastycznością i głęboką integracją z ekosystemem WordPress.[1]

- **FR-ECOMM-01:** Kiedy użytkownik oznaczy książkę lub wydanie muzyczne jako „Opublikowane do sprzedaży", system musi uruchomić zautomatyzowany (lub półautomatyzowany w fazie MVP) proces, który utworzy odpowiedni produkt cyfrowy w połączonym sklepie WooCommerce. Proces ten musi przenieść kluczowe dane, takie jak tytuł, opis, cena i obraz okładki.[1]

4.3. Marketing i Automatyzacja Przepływów Pracy

Automatyzacja powtarzalnych zadań marketingowych i komunikacyjnych jest

kluczowym elementem propozycji wartości platformy.

- **FR-AUTO-01:** Platforma musi integrować się z MailerLite, umożliwiając twórcom umieszczanie formularzy zapisu na newsletter na swoich stronach produktowych i budowanie listy mailingowej.[1]
- **FR-AUTO-02:** System powinien być zaprojektowany z myślą o przyszłej integracji z narzędziami do automatyzacji, takimi jak n8n lub Zapier. Umożliwi to tworzenie zaawansowanych przepływów pracy, np. automatyczne wysłanie kampanii e-mail do subskrybentów twórcy w momencie opublikowania nowego utworu.[1]

5.0 Wymagania Niefunkcjonalne (NFRs)

Wymagania niefunkcjonalne definiują standardy jakościowe i operacyjne systemu, które są równie ważne jak jego funkcje.

- **NFR-01: Wydajność:** Czas odpowiedzi API dla typowych zapytań (np. pobranie listy projektów) nie powinien przekraczać 500 ms. Czas ładowania stron front-endowych w panelu użytkownika powinien dążyć do wartości poniżej 2 sekund.
- **NFR-02: Skalowalność:** Architektura backendu musi być zaprojektowana w sposób umożliwiający skalowanie horyzontalne (dodawanie kolejnych instancji usług) w odpowiedzi na rosnące obciążenie. Jest to główna przesłanka stojąca za wyborem architektury opartej na mikroserwisach, która ewoluowała z początkowych koncepcji.[1]
- **NFR-03: Bezpieczeństwo:** Cały ruch sieciowy do i z platformy musi być szyfrowany przy użyciu protokołu HTTPS. System musi być zabezpieczony przed powszechnymi atakami internetowymi (np. SQL Injection, XSS). Hasła użytkowników muszą być przechowywane w bazie danych w postaci bezpiecznie zahashowanej. Implementacja uwierzytelniania opartego na tokenach JWT musi być zgodna z najlepszymi praktykami bezpieczeństwa, w tym zarządzaniem kluczami tajnymi.[1]
- **NFR-04: Niezawodność:** Platforma powinna dążyć do osiągnięcia dostępności na poziomie 99,9%. Obowiązkowe jest wdrożenie zautomatyzowanych, regularnych kopii zapasowych bazy danych (Amazon RDS) oraz plików przechowywanych w Amazon S3, z jasno zdefiniowaną procedurą odzyskiwania danych po awarii.[1]
- **NFR-05: Zgodność (Compliance):** Platforma musi być zgodna z ogólnym

rozporządzeniem o ochronie danych (RODO/GDPR), zapewniając użytkownikom kontrolę nad ich danymi osobowymi. Musi również zawierać mechanizmy umożliwiające zarządzanie własnością intelektualną i zgłaszanie naruszeń praw autorskich (procedura DMCA).

Plan Techniczny (Technical Blueprint) – HardbanRecords Lab

1.0 Przegląd Architektury Systemu

1.1. Filozofia Architektoniczna

Platforma HardbanRecords Lab wykorzystuje nowoczesną, odseparowaną architekturę opartą na wzorcu mikroserwisów. Frontend, zrealizowany w oparciu o WordPress i Elementor Pro, pełni rolę klienta UI, który komunikuje się poprzez API REST z zestawem niezależnych usług backendowych zbudowanych w technologii FastAPI (Python). Taki projekt stanowi świadomą ewolucję od wcześniejszych, bardziej monolitycznych koncepcji [1], a jego głównym celem jest priorytetyzacja skalowalności, łatwości utrzymania oraz możliwości niezależnego wdrażania poszczególnych komponentów biznesowych.[1] Ta separacja pozwala na elastyczny rozwój modułów muzycznego i książkowego, a także na przyszłą rozbudowę systemu bez wpływu na istniejące funkcjonalności.

1.2. Wysokopoziomowy Diagram Architektury

Poniższy diagram ilustruje przepływ danych i interakcje pomiędzy kluczowymi komponentami systemu:

Fragment kodu

```
graph TD
    A[Użytkownik] --> B{Przeglądarka Internetowa};
    B --> C;
    C --> D{Nginx (Reverse Proxy)};
    D -- ścieżka /auth/* --> E;
    D -- ścieżka /music/* --> F;
    D -- ścieżka /books/* --> G;
    E --> H;
    F --> H;
    G --> H;
    F --> I;
    G --> I;
```

1.3. Stos Technologiczny

Poniżej znajduje się ostateczna lista technologii, które będą wykorzystane do budowy i wdrożenia platformy, zgodnie z najbardziej aktualnymi planami technicznymi.[1]

- **Frontend:** WordPress, Elementor Pro
- **Usługi Backendowe:** Python 3.11, FastAPI, Uvicorn
- **Baza Danych:** PostgreSQL (hostowana na Amazon RDS)
- **Przechowywanie Plików:** Amazon S3
- **Infrastruktura:** Amazon Web Services (EC2, RDS, S3)
- **Odwrotne Proxy (Reverse Proxy):** Nginx
- **Middleware do Automatyzacji:** n8n (self-hosted)

2.0 Architektura Mikroserwisów Backendu (FastAPI)

Architektura backendu jest sercem platformy. Została zaprojektowana jako zbiór

trzech odrębnych, ale współpracujących ze sobą mikroserwisów. Taka dekompozycja jest bezpośrednią realizacją ewolucji projektu w kierunku fizycznej separacji modułów, co ma kluczowe znaczenie dla skalowalności i izolacji błędów.[1]

2.1. Struktura Katalogów

W celu ułatwienia zarządzania kodem, współdzielenia wspólnych komponentów i niezależnego wdrażania, proponuje się następującą strukturę monorepozytorium, inspirowaną dyskusjami na temat separacji modułów [1]:

```
backend/
├── common/
│   ├── auth.py        # Współdzielona logika walidacji JWT, zależności RBAC
│   ├── database.py    # Konfiguracja sesji bazy danych
│   └── models.py      # Współdzielone modele SQLAlchemy (np. User, Role)
├── auth_app/
│   └── main.py        # Główna aplikacja serwisu uwierzytelniania
├── music_app/
│   ├── main.py        # Główna aplikacja serwisu muzycznego
│   ├── routers.py
│   └── schemas.py
└── books_app/
    ├── main.py        # Główna aplikacja serwisu książkowego
    ├── routers.py
    └── schemas.py
```

2.2. Centralny Serwis Uwierzytelniania (auth_app)

Ten serwis jest fundamentem bezpieczeństwa całej platformy. Jego jedyną odpowiedzialnością jest zarządzanie tożsamością użytkowników.

- **Odpowiedzialność:** Obsługa rejestracji użytkowników, proces logowania oraz wydawanie i odświeżanie tokenów JSON Web Tokens (JWT). Działa jako jedyne źródło prawdy (Single Source of Truth) w kwestii tożsamości.
- **Struktura JWT:** Ładunek (payload) wydawanego tokena JWT będzie zawierał następujące kluczowe pola: sub (ID użytkownika), role (np. admin, music_creator, book_author), exp (czas wygaśnięcia) oraz iat (czas wydania).
- **Bezpieczeństwo:** Klucz tajny (JWT_AUTH_SECRET_KEY) używany do podpisywania tokenów jest krytycznym elementem konfiguracji. Musi być zarządzany jako bezpieczna zmienna środowiskowa i współdzielony z serwisami music_app i books_app, aby mogły one poprawnie weryfikować autentyczność otrzymanych tokenów. Bez tego współdzielonego sekretu, mechanizm kontroli dostępu opartej na rolach (RBAC) nie będzie działał.[1]

Specyfikacja API Serwisu Uwierzytelniania:

Metoda	Ścieżka	Opis	Ciało Żądania (Request Body)	Odpowiedź (Success 200)
POST	/auth/register	Rejestruje nowego użytkownika.	UserCreate (email, hasło)	Token (access_token, token_type)
POST	/auth/login	Loguje użytkownika i zwraca token JWT.	OAuth2PasswordRequestForm	Token (access_token, token_type)
POST	/auth/refresh	Odświeża wygasły token dostępu.	RefreshToken	Token (access_token, token_type)
GET	/users/me	Zwraca dane zalogowanego użytkownika.	Brak	UserPublic (id, email, rola)

2.3. Serwis Wydawnictwa Muzycznego (music_app)

- **Odpowiedzialność:** Zarządzanie całą logiką biznesową związaną z wydaniami

muzycznymi, od tworzenia metadanych po generowanie pakietów dystrybucyjnych.

- **Autoryzacja:** Każdy chroniony punkt końcowy w tym serwisie będzie wykorzystywał zależność (dependency) FastAPI, która weryfikuje przychodzący token JWT (używając współdzielonego klucza tajnego) i sprawdza, czy rola użytkownika to music_creator lub admin.[1]

Specyfikacja API Serwisu Muzycznego:

Metoda	Ścieżka	Opis	Wymagana Rola
POST	/music/	Tworzy nowe wydanie muzyczne.	music_creator, admin
GET	/music/list	Zwraca listę wydań zalogowanego użytkownika.	music_creator, admin
GET	/music/{id}	Zwraca szczegóły konkretnego wydania.	music_creator, admin
PUT	/music/{id}	Aktualizuje metadane wydania.	music_creator, admin
DELETE	/music/{id}	Usuwa wydanie muzyczne.	music_creator, admin
GET	/music/{id}/stats	Zwraca statystyki dla wydania.	music_creator, admin
POST	/music/{id}/export-route-note	Generuje pakiet ZIP+CSV dla RouteNote.	music_creator, admin

2.4. Serwis Wydawnictwa Cyfrowego (books_app)

- **Odpowiedzialność:** Zarządzanie całą logiką biznesową związaną z publikacją

książek, w tym importem, konwersją formatów i integracją z AI.
- **Autoryzacja:** Analogicznie do serwisu muzycznego, każdy chroniony punkt końcowy będzie zabezpieczony przez zależność weryfikującą token JWT i rolę (book_author lub admin).

Specyfikacja API Serwisu Książkowego:

Metoda	Ścieżka	Opis	Wymagana Rola
POST	/books/	Tworzy nowy projekt książkowy.	book_author, admin
GET	/books/list	Zwraca listę książek zalogowanego użytkownika.	book_author, admin
GET	/books/{id}	Zwraca szczegóły konkretnej książki.	book_author, admin
PUT	/books/{id}	Aktualizuje metadane książki.	book_author, admin
DELETE	/books/{id}	Usuwa projekt książkowy.	book_author, admin
POST	/books/{id}/import	Importuje manuskrypt z pliku.	book_author, admin
POST	/books/{id}/generate-cover	Generuje okładkę przy użyciu AI.	book_author, admin
POST	/books/{id}/export	Eksportuje książkę do formatów PDF/EPUB/MOBI.	book_author, admin
GET	/books/{id}/stats	Zwraca statystyki dla książki.	book_author, admin

3.0 Architektura Bazy Danych (PostgreSQL)

3.1. Filozofia Projektu Schematu

Schemat bazy danych został zaprojektowany w oparciu o znormalizowany model relacyjny, aby zapewnić integralność i spójność danych. Klucze obce (foreign keys) będą rygorystycznie egzekwować relacje pomiędzy użytkownikami, ich rolami oraz treściami, które tworzą. Taki projekt jest fundamentem dla stabilnego działania całego systemu.

3.2. Tabela: Podstawowy Schemat Bazy Danych

Poniższa tabela definiuje strukturę kluczowych tabel w bazie danych PostgreSQL. Jest to fundament, na którym zostaną zbudowane modele SQLAlchemy we wszystkich serwisach backendowych.

Tabela	Kolumna	Typ Danych	Opis
roles	id	SERIAL PRIMARY KEY	Unikalny identyfikator roli.
	name	VARCHAR(50) UNIQUE NOT NULL	Nazwa roli (np. 'admin', 'music_creator', 'book_author').
users	id	SERIAL PRIMARY KEY	Unikalny identyfikator użytkownika.
	email	VARCHAR(255) UNIQUE NOT NULL	Adres e-mail użytkownika (służy jako login).
	hashed_password	VARCHAR(255) NOT	Zahashowane hasło

		NULL	użytkownika.
	role_id	INTEGER REFERENCES roles(id)	Klucz obcy do tabeli roles.
	created_at	TIMESTAMP WITH TIME ZONE	Data i czas utworzenia konta.
music_releases	id	SERIAL PRIMARY KEY	Unikalny identyfikator wydania muzycznego.
	user_id	INTEGER REFERENCES users(id)	Klucz obcy do tabeli users, wskazujący właściciela.
	title	VARCHAR(255) NOT NULL	Tytuł wydania.
	metadata_json	JSONB	Pozostałe metadane (artysta, gatunek, etc.) w formacie JSON.
	s3_audio_key	VARCHAR(1024)	Klucz do obiektu pliku audio w Amazon S3.
	s3_cover_art_key	VARCHAR(1024)	Klucz do obiektu pliku okładki w Amazon S3.
	status	VARCHAR(50)	Status wydania (np. 'draft', 'published').
books	id	SERIAL PRIMARY KEY	Unikalny identyfikator książki.
	user_id	INTEGER REFERENCES users(id)	Klucz obcy do tabeli users, wskazujący właściciela.

	title	VARCHAR(255) NOT NULL	Tytuł książki.
	metadata_json	JSONB	Pozostałe metadane (autor, ISBN, etc.) w formacie JSON.
	s3_source_file_key	VARCHAR(1024)	Klucz do obiektu pliku źródłowego w Amazon S3.
	s3_cover_art_key	VARCHAR(1024)	Klucz do obiektu pliku okładki w Amazon S3.
	status	VARCHAR(50)	Status książki (np. 'draft', 'published').

4.0 Architektura Integracji Frontendu (WordPress + Elementor)

4.1. Wzorzec Komunikacji

Frontend będzie komunikował się z backendem FastAPI wyłącznie za pomocą asynchronicznych zapytań JavaScript, wykorzystując standardowy fetch API. Wszystkie zapytania do chronionych punktów końcowych API muszą zawierać nagłówek Authorization z tokenem JWT w formacie Bearer <token>.[1]

4.2. Zarządzanie Stanem po Stronie Klienta

- **Przechowywanie Tokena:** Token JWT otrzymany po pomyślnym zalogowaniu będzie przechowywany w localStorage przeglądarki. Umożliwi to persystencję sesji użytkownika po zamknięciu i ponownym otwarciu karty przeglądarki.[1]

- **Protokół Bezpieczeństwa:** Przechowywanie tokenów w localStorage naraża je na ryzyko kradzieży poprzez ataki Cross-Site Scripting (XSS). Aby zminimalizować to ryzyko, wdrożone zostaną następujące środki zaradcze, zgodnie z zaleceniami dotyczącymi bezpieczeństwa [1]:
    1. **Content Security Policy (CSP):** Na stronie WordPress zostanie wdrożona rygorystyczna polityka CSP, ograniczająca źródła, z których mogą być ładowane i wykonywane skrypty.
    2. **Sanityzacja Danych Wejściowych:** Wszystkie treści generowane przez użytkowników (np. w przyszłych komentarzach) będą starannie filtrowane i sanityzowane przed wyświetleniem.
    3. **Krótkożyjące Tokeny:** W przyszłości zaleca się wdrożenie mechanizmu krótkożyjących tokenów dostępu (np. 15 minut) i długożyjących tokenów odświeżających, co znacznie ograniczy okno czasowe, w którym skradziony token jest użyteczny.

4.3. Implementacja Dynamicznych Widżetów

Implementacja złożonych, interaktywnych funkcji, takich jak przesyłanie plików czy formularze, za pomocą widżetu „Kod HTML" w Elementorze, niesie ryzyko stworzenia trudnego w utrzymaniu kodu („spaghetti code").[1] Aby temu zapobiec, deweloperzy powinni stosować ustrukturyzowany wzorzec. Poniżej znajduje się szablon kodu dla typowego widżetu wyświetlającego listę danych z API.

JavaScript

```
<div id="music-list-container">
    <p>Ładowanie Twoich wydań...</p>
</div>

<script>
document.addEventListener('DOMContentLoaded', function() {
    const container = document.getElementById('music-list-container');
    const token = localStorage.getItem('access_token');

    if (!token) {
```

```
        container.innerHTML = '<p>Błąd: Nie jesteś zalogowany.</p>';
        // Opcjonalnie: przekierowanie do strony logowania
        // window.location.href = '/login';
        return;
    }

    fetch('https://your_api_domain.com/music/list', {
        method: 'GET',
        headers: {
            'Authorization': 'Bearer ' + token,
            'Content-Type': 'application/json'
        }
    })
    .then(response => {
        if (!response.ok) {
            throw new Error('Błąd sieci lub autoryzacji: ' + response.status);
        }
        return response.json();
    })
    .then(data => {
        if (data.length === 0) {
            container.innerHTML = '<p>Nie masz jeszcze żadnych wydań.</p>';
            return;
        }

        let html = '<ul>';
        data.forEach(release => {
            html += `<li>${release.title} - <i>(${release.status})</i></li>`;
        });
        html += '</ul>';
        container.innerHTML = html;
    })
    .catch(error => {
        console.error('Błąd podczas pobierania danych:', error);
        container.innerHTML = '<p>Wystąpił błąd podczas ładowania danych. Spróbuj ponownie później.</p>';
    });
});
</script>
```

5.0 Infrastruktura, Wdrożenie i Operacje

5.1. Alokacja Infrastruktury AWS

Infrastruktura zostanie oparta na usługach AWS, zgodnie z początkowymi założeniami projektu.[1]

- **EC2:** Instancja t2.micro z systemem operacyjnym Ubuntu 24.04. Grupy bezpieczeństwa (Security Groups) będą skonfigurowane tak, aby zezwalać na publiczny dostęp przychodzący na portach 80 (HTTP) i 443 (HTTPS) oraz na dostęp SSH (port 22) wyłącznie z zaufanych adresów IP.
- **RDS:** Instancja PostgreSQL db.t2.micro. Dostęp do bazy danych będzie ograniczony wyłącznie do grupy bezpieczeństwa instancji EC2. Publiczny dostęp do bazy danych będzie zablokowany.
- **S3:** Prywatny zasobnik (bucket) S3 będzie służył do przechowywania wszystkich mediów przesyłanych przez użytkowników (pliki audio, okładki, manuskrypty). Dostęp do plików będzie realizowany poprzez generowane przez FastAPI, tymczasowe, podpisane cyfrowo adresy URL (pre-signed URLs).

5.2. Wdrażanie i Zarządzanie Usługami

Wdrożenie architektury mikroserwisowej wymaga precyzyjnej konfiguracji routingu i zarządzania procesami.[1]

- **Nginx jako Odwrotne Proxy:** Nginx będzie nasłuchiwał na portach 80 i 443, obsługiwał terminację połączeń SSL i przekierowywał ruch na podstawie prefiksu ścieżki URL do odpowiedniego procesu Uvicorn działającego lokalnie.
- **Tabela: Konfiguracja Routingu w Nginx.** Poniższy blok konfiguracyjny jest kluczowy dla prawidłowego działania systemu i musi zostać zaimplementowany w konfiguracji serwera wirtualnego Nginx.[1]

Ścieżka URL	Docelowy Port Lokalny	Serwis Backendowy
/auth/	http://localhost:8000	auth_app
/music/	http://localhost:8001	music_app

/books/	http://localhost:8002	books_app

Przykładowa konfiguracja Nginx:

```
Nginx


server {
    listen 80;
    server_name your_api_domain.com;
    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl;
    server_name your_api_domain.com;

    ssl_certificate /path/to/fullchain.pem;
    ssl_certificate_key /path/to/privkey.pem;

    # Konfiguracja CORS
    add_header 'Access-Control-Allow-Origin' 'https://your_wordpress_domain.com' always;
    #... pozostałe nagłówki CORS

    location /auth/ {
        proxy_pass http://localhost:8000;
        proxy_set_header Host $host;
        #... pozostałe nagłówki proxy
    }

    location /music/ {
        proxy_pass http://localhost:8001;
        proxy_set_header Host $host;
        #... pozostałe nagłówki proxy
    }

    location /books/ {
```

```
        proxy_pass http://localhost:8002;
        proxy_set_header Host $host;
        #... pozostałe nagłówki proxy
    }
}
```

- **Systemd do Zarządzania Procesami:** Każda aplikacja FastAPI (auth_app, music_app, books_app) będzie zarządzana jako osobna usługa systemd. Zapewni to jej automatyczne uruchamianie przy starcie systemu oraz ponowne uruchomienie w przypadku awarii. Dla każdej aplikacji zostanie utworzony plik .service w katalogu /etc/systemd/system/, podobny do poniższego przykładu dla music_app.[1]

```
Ini, TOML
```

```
[Unit]
Description=FastAPI Music Application
After=network.target

User=ubuntu
Group=www-data
WorkingDirectory=/path/to/backend/
ExecStart=/path/to/venv/bin/uvicorn music_app.main:app --host 127.0.0.1 --port 8001
Restart=always

[Install]
WantedBy=multi-user.target
```

5.3. Wzmacnianie Bezpieczeństwa i Monitorowanie

- **Firewall:** Zapora sieciowa UFW zostanie skonfigurowana na serwerze EC2, aby dodatkowo egzekwować reguły zdefiniowane w grupach bezpieczeństwa na poziomie systemu operacyjnego.[1]

- **Monitorowanie:** Podstawowe monitorowanie działania aplikacji i systemu zostanie zrealizowane przy użyciu wbudowanych narzędzi, takich jak journalctl do przeglądania logów usług systemd oraz logrotate do zarządzania rotacją plików logów.[1]
- **Kopie Zapasowe i Odzyskiwanie Danych:** Zostaną skonfigurowane i zweryfikowane zautomatyzowane, codzienne migawki (snapshots) dla instancji RDS i EC2, oferowane przez AWS. Zostanie opracowana i udokumentowana procedura odzyskiwania danych z tych migawek w scenariuszu awaryjnym.[1]

5.4. Potok CI/CD (Rekomendacja na Przyszłość)

Chociaż w fazie MVP wdrożenia mogą być realizowane ręcznie, zaleca się zaplanowanie wdrożenia potoku ciągłej integracji i ciągłego dostarczania (CI/CD). Wykorzystanie narzędzi takich jak GitHub Actions do automatycznego uruchamiania testów i wdrażania zmian na instancję EC2 za pomocą SSH znacząco poprawi szybkość, niezawodność i bezpieczeństwo procesu deweloperskiego w przyszłości.[1]

Cytowane prace

1. Ewolucja Architektury HardbanRecords Lab.pdf