AKADEMIA NAUK STOSOWANYCH
W NOWYM SĄCZU

Wydział Nauk Inżynieryjnych
Katedra Informatyki

DOKUMENTACJA PROJEKTOWA

Bazy danych

System zarządzania pieśniami - Cantabo Manager

Autor:
Szymon Pogwizd
Jakub Poręba

Prowadzący:
mgr inż. Nikodem Bulanda

Nowy Sącz 2023

Spis treści

1. Nazwa projektu

"System zarządzania pieśniami - Cantabo Manager"

Nazwa projektu składa się z dwóch części. Pierwsza część "System zarządzania pieśniami"w precyzyjny sposób opisuje funkcjonalność, jaką ma pełnic aplikacja. Druga część nazwy to właściwa nazwa aplikacji webowej. "Cantabo Manager"jest połączeniem dwóch słów: "Cantabo", co po łacinie oznacza "będę śpiewał", oraz "Manager", co po angielsku oznacza "zarządca". Nazwa ta nawiązuje do funkcjonalności systemu, którym będzie zarządzanie pieśniami religijnymi. Słowo "Cantabo"podkreśla aspekt muzyczny, który jest kluczowy dla tego projektu, a słowo "Manager"wskazuje na funkcjonalność systemu zarządzania. Jest to nazwa łatwa do zapamiętania i jednocześnie nawiązująca do istoty projektu.

2. Cel

Celem projektu jest stworzenie aplikacji webowej, która umożliwi użytkownikowi zarządzanie biblioteką pieśni w ramach instytucji religijnej. Użytkownik dzięki aplikacji ma uzyskać możliwość łatwego zarządzania pieśniami, playlistami oraz profilami. Ponadto każda pieśń może zawierać slajdy oraz być przypisana do kategorii. Swoje kategorie posiadać mogą również playlisty. Przypisywanie do kategorii umożliwi łatwe znalezienie pieśni lub gotowej playlisty na daną okazję lub okres liturgiczny. Profile umożliwią natomiast zapisanie wielu ustawień, które będzie można łatwo przełączyc zmieniając jedynie aktywny profil.

3. Zakres

3.1. Analiza wymagań

Główne wymagania dla systemu obejmują:

- Stworzenie systemu zarządzania pieśniami religijnymi.
- Zapewnienie wspólnej podstawowej bazy pieśni dla wszystkich użytkowników.
- Przypisywanie użytkowników do konkretnej grupy.
- Możliwość dostosowywania pieśni dla konkretnej parafii (w różnych parafiach mogą być lekko pozmieniane teksty do tej samej piosenki).
- Grupowanie w kategorie oraz zapisywanie playlist na wybrane okazje.
- Stworzenie interfejsu użytkownika umożliwiającego prostą i intuicyjną obsługę.

3.2. Wymagania funkcjonalne i niefunkcjonalne

3.2.1. Wymagania funkcjonalne

- Użytkownik powinien mieć możliwość zalogowania się do systemu przy użyciu swojej nazwy użytkownika i hasła (+)
- Użytkownik powinien mieć możliwość dostępu do listy pieśni. MVP (+)
- Użytkownik powinien mieć możliwość dostępu do listy playlist. MVP (+)
- Użytkownik powinien mieć możliwość dostępu do listy profili. (+)
- Użytkownik poinien mieć możliwość dostępu do ustawień konta. (+/-)
- Użytkownik powinien mieć możliość wysłania wiadomości do superadministratora. (+)
- Użytkownik powinien mieć możliwość dodanie, edycji oraz usunięcia pieśni. MVP (+)
- Użytkownik powinien mieć możliwość przypisania kategorii do pieśni. MVP (+)
- Użytkownik powinien mieć możliwość dodania, edycji oraz usunięcia kategorii pieśni. MVP (+)

- Użytkownik powinien mieć możliwość dodania, edycji oraz usunięciu slajdu z pieśni. MVP (+)
- Użytkownik powinien mieć możliowść korzystania z edytora slajdów. MVP (+)
- Użytkownik powinien mieć możliwość korzystania z edytora slajdów w którym powinien mieć możliwość wybrania: poziomu nagłówka, pogrubienia, kursywy, podkreślenia, przekreślenia, koloru tekstu, koloru tła tekstu, indeksu dolnego, indeksu górnego, cytatu, elementu wyróżniającego, listy numerowej, listy punktowej, zmniejsz wcięcia, zwiększ wcięcie, rodzaj wcięcia oraz usuń efekty z teskstu. (+)
- Użytkownik powinien mieć możliwość dodania, edycji oraz usunięcia playlist. MVP (+)
- Użytkownik powinien mieć możliwość przypisania kategorii do playlisty. MVP (+)
- Użytkownik powinien mieć możliwość dodania, edycji oraz usunięcia kategorii playlisty. MVP (+)
- Użytkownik powinien mieć możliwość wyświetlenia na liście jedynie pieśni przypisane do wybranej kategorii. MVP (+)
- Użytkownik powinien miec możliwość wyszukiwania piesni po tytule oraz zawartości. MVP (+/-)
- Użytkownik powinien mieć możliwość wyświetlenia na liście jedynie playlisty przypisane do wybranej kategorii. MVP (+)
- Użytkownik powinien miec możliwość wyszukiwania playlist po tytule. MVP (+)
- Użytkownik powinien mieć możliwość usunięcia zarówno całej pieśni jaki pojedynczych slajdów z playlisty. (+/-)
- Użytkownik powinien mieć możliwość zmiany kolejności pieśni w playliście. (-)
- Użytkownik powinien mieć możliwość dodania, edycji oraz usunięcia profilu. (+)
- Użytkownik podczas edycji oraz tworzenia profilu powinien mieć możliwość ustawiania: nazwy profilu, koloru tła slajdów, koloru czcionki slajdów, koloru wyłączonego ekranu, domyślnego wyrównania tekstu, maksymalną wielkość czcionki, wielkość marginesów, minimalną wielkość czcionki oraz wybrania czy: profil jest aktywny, wybrane utwory są sortowane według częstości użycia, pokazywany jest tytuł utworu na pierwszym slajdzie, wszytko pisane jest dużymi literami, przed każdym utworem znajduje się pusty slajd, kolory są odwrócone, lista slajdów podczas przeglądania

playlist jest rozwinięta. (+)

- Administartor powinien mieć możliwość wykonania wszystkich czynności jakie może wykonać użytkownik. MVP (+)
- Administrator powinien mieć możliwość dostępu do listy użytkowników. (+)
- Administrator powinien mieć możliwość dodania nowego użytkownika. (+)
- Administrator powinien mieć możliwość usunięcia użytkownika. (+)
- Administrator powinien mieć możliwość edycji użytkownika. (+)

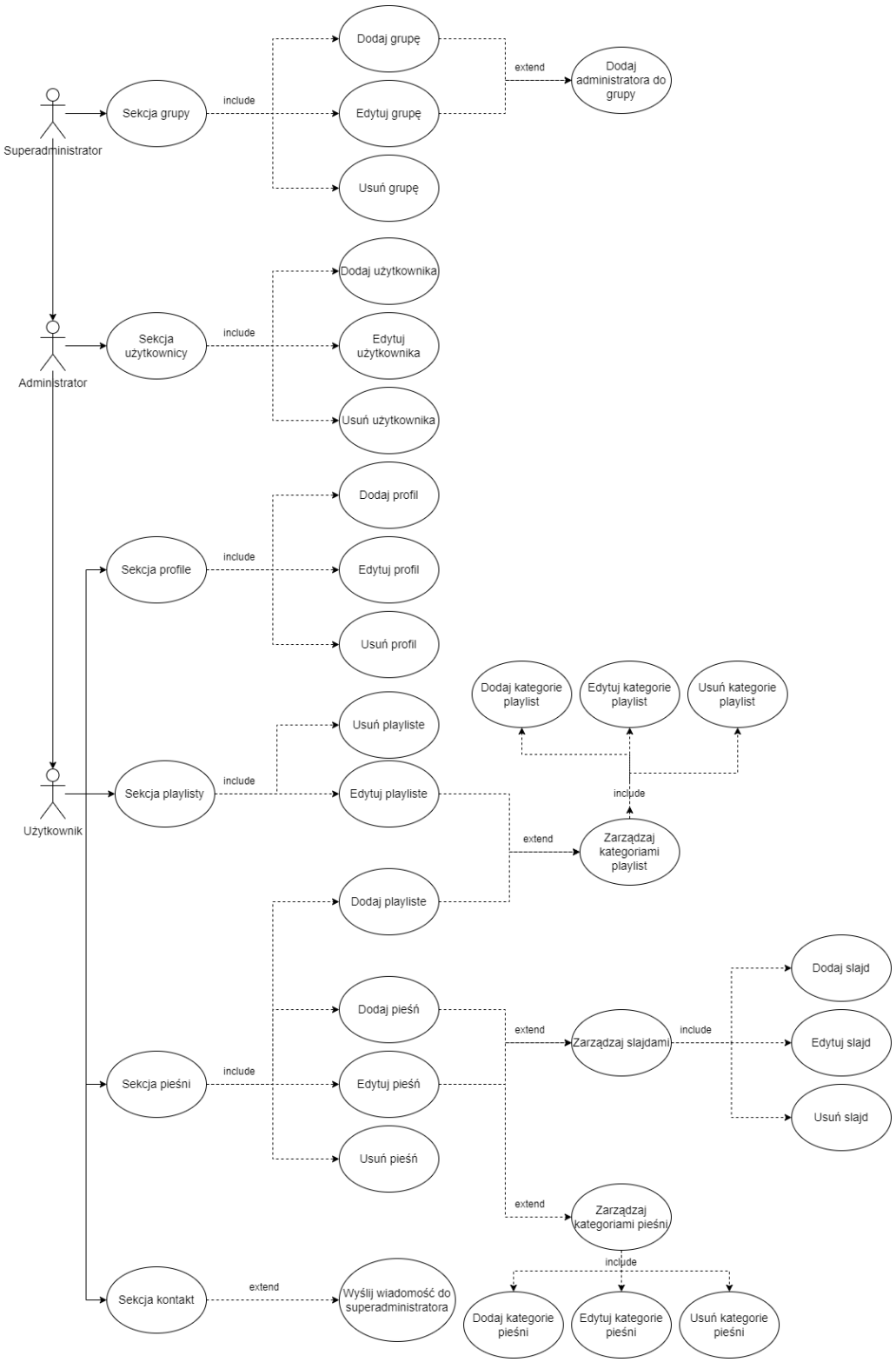
- Superadministartor powinien miec możliwość wykonania wszystkich czynności jakie może wykonać użytkownik oraz administrator poza opcją wysłania wiadomości do superadministartora. MVP (+)
- Superadministrator powinien mieć możliwość dostępu do listy grup. (+/-)
- Superadministrator powinien mieć możliwość dodania nowej grupy. (-)
- Superadministrator powinien mieć możliwiość usunięcia grupy. (-)
- Superadministrator powinien mieć możliwiość edycji grupy. (-)
- Superadministrator powinien mieć możliwość dodania administratora do grupy. (-)

3.2.2. Wymagania niefunkcjonalne

- Dostępność: aplikacja ma być dostępna 24/7 a o wszelkich przerwach w działaniu użytkownik ma być informowany.
- Dostępność: aplikacja ma działać we wszystkich popularnych przeglądarkach internetowych (Chrome, Firefox, Opera, Safari, Edge, Brave)
- Bezpieczeństwo: aplikacja ma zapewniać zabezpieczenie przed nieautoryzowanym dostępem.
- Bezpieczeństwo: aplikacja ma wyświetlać odpowiednie komunikaty oraz alerty w przypadku wprowadzenia nieodpowiednich lub niepoprawnych danych.
- Konserwacja: aplikacja ma posiadać możliwość łatwego utrzymania oraz aktualizacji.

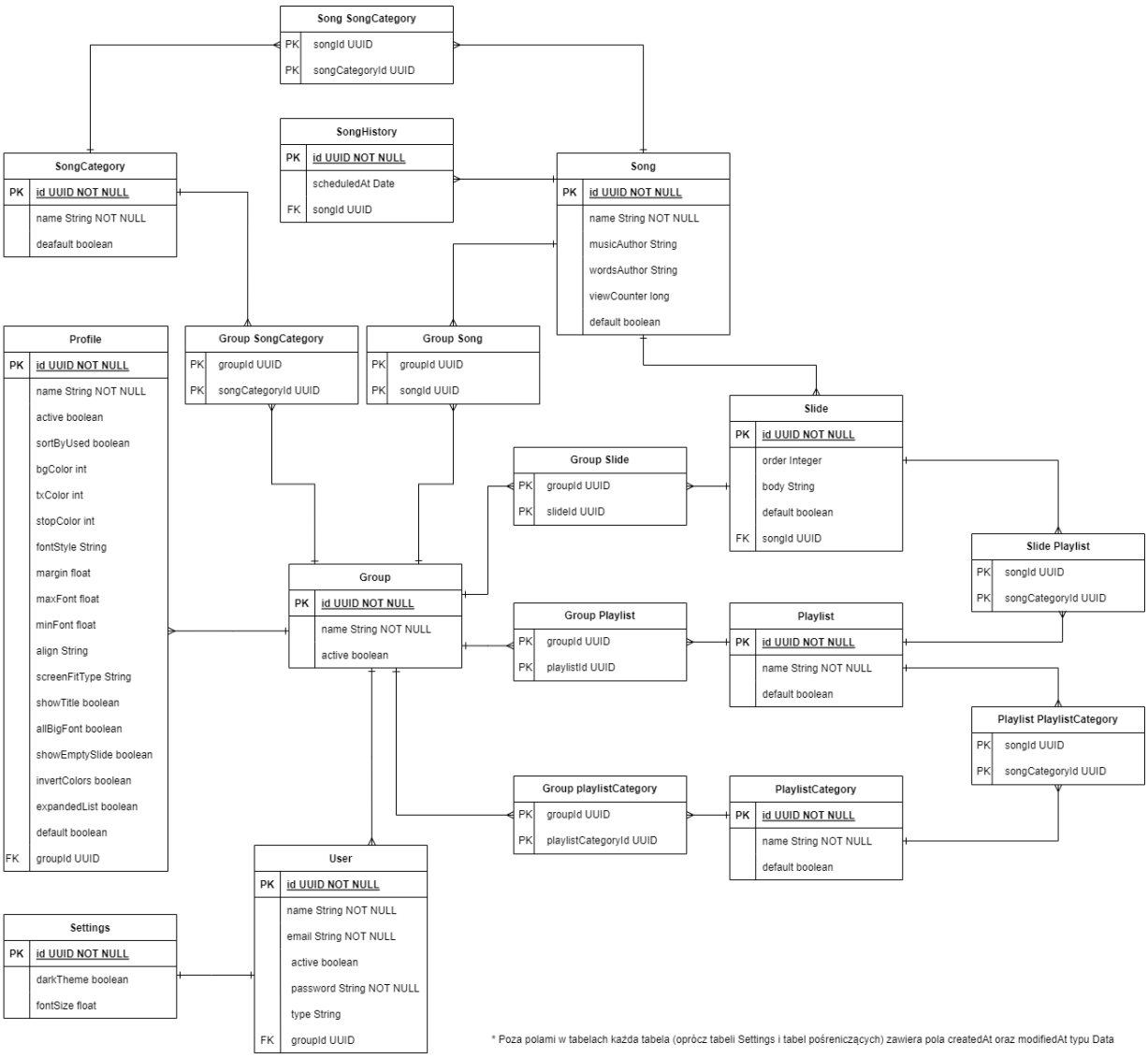
- Użyteczność: aplikacja ma być łatwa w obsłudze (intuicyjna).
- Wsparcie: aplikacja ma zapewnić możliwość zgłaszania błędów przez formularz kontaktowy.
- Wsparcie: użytkownik po zgłoszeniu błędu ma otrzymać przewidywany termin naprawy maksymalnie wciągu tygodnia od wpłynięcia zgłoszenia.
- Wsparcie: aplikacja ma posiadać testy jednostkowe wdrożonych funkcjonalności.
- Wydajność: aplikacja ma działać na terytorium Polski.
- Wydajność: aplikacja ma być w stanie obsłużyć wszystkich dodanych użytkowników bez pogorszenia wydajności.
- Wdrożenie: aplikacja ma być w pełni ukończona oraz wdrożona przed 24.06.2023 przy czym zalecany termin zakończenia prac nad aplikacją to 03.06.2023 aby zapewnić czas na nieprzewidziane okoliczności oraz opóźnienia.

3.3. Diagram przypadków użycia



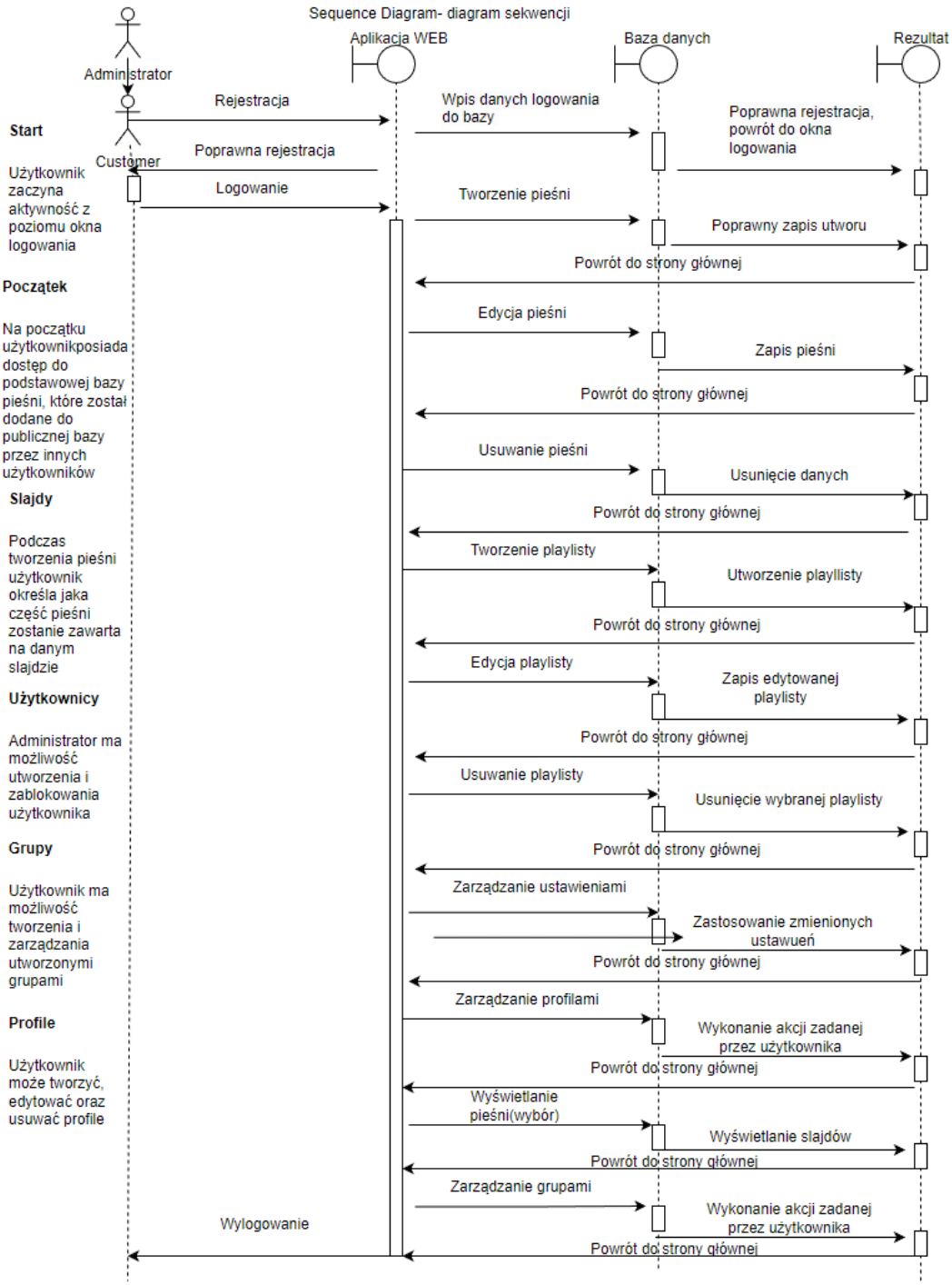
Rys. 3.1. Diagram przypadków użycia

3.4. Diagram ERD



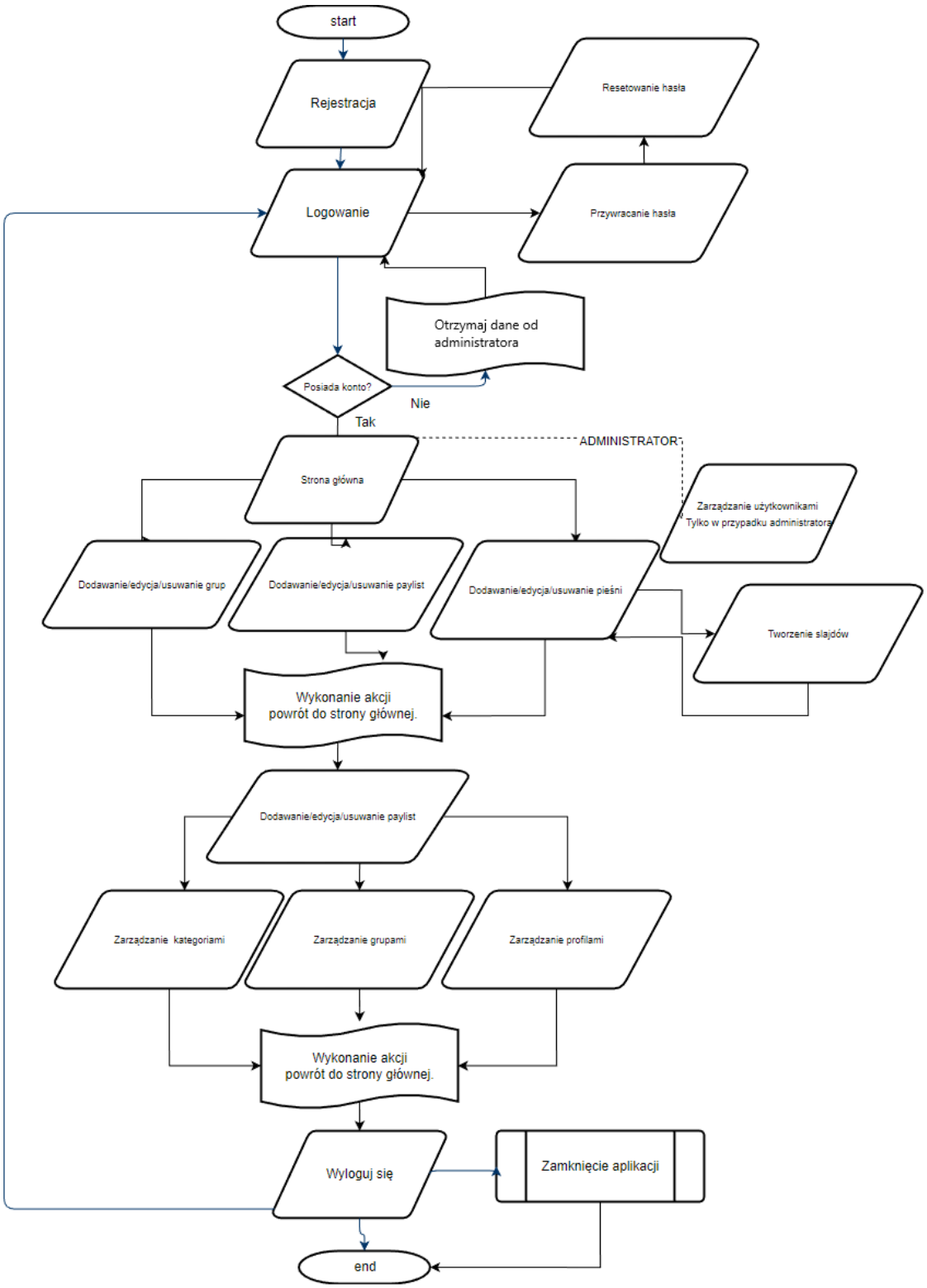
Rys. 3.2. Diagram ERD

3.5. Diagram sekwencji



Rys. 3.3. Diagram sekwencji

3.6. Diagram aktywności



Rys. 3.4. Diagram aktywności

3.7. Dobór technologii

- JAVA - jeden z najpopularniejszych języków programowania, który jest szeroko stosowany w tworzeniu aplikacji webowych. Jave oferuje wiele zalet, takich jak wydajność, bezpieczeństwo, skalowalność i wsparcie społeczności.
- Spring Framework - jeden z najpopularniejszych frameworków do tworzenia aplikacji webowych w Javie. Ułatwia tworzenie skalowalnych i wydajnych aplikacji webowych, Spring Framework oferuje wiele funkcji, takich jak wstrzykiwanie zależności, obsługa transakcji, obsługa żądań HTTP, bezpieczeństwo itp.
- JUnit - framework to testowania jednostkowego w Java. Używanie JUnit pozwala na łatwa i automatyczne testowanie poszczególnych modułów aplikacji i zapewnia, że są one wydajne i poprawne.
- Gradle - narzędzie to automatyzacji budowy aplikacji, które pozwala na łatwe zarządzanie zależnościami i konfiguracją projektu. Gradle oferuje wiele funkcji ułatwiających budowanie i testowania aplikacji oraz zapewnia ich poprawne wdrożenie.
- Figma - narzędzie do projektowania interfejsów użytkownika, które umożliwia łatwe tworzenie prototypów, projektowanie UI / UX i współpracę w czasuie rzeczywistym.
- React - framework dla języka JavaScript, który umożliwia interaktywnych interfejsów użytkownika. Jest to szybka, skalowalna i efektywana technologia, która pozwala na łatwe tworzenie interaktywnych i dynamicznych stron internetowych. React jest również kompatybilny z wieloma innymi bibliotekami i narzędziami, co umożliwia łatwe integrowanie go z innymi częściami aplikacji.
- Bootstrap - framework front-endowy do tworzenia responsywnych stron internetowych. Bootstrap zapewnia gotowe komponenty interfejsu użytkownika, takie jak formularze, przyciski itp., które można łatwo dostosować do swoich potrzeb. Dzięki temu można zaoszczędzić czas i zapewnić spójny wygląd aplikacji.
- HTML - język znaczników, który umożliwia tworzenie struktury i zawartości strony internetowej. Jest to podstawowa technologia dla każdej aplikacji intenetowej.
- CSS - język stylów, który umożliwia dodanie wyglądu i stylu do strony internetowej. Dzięki CSS można dostosować kolory, czcionki, rozmiary, tła, marginesy i wiele innych elementów, aby stworzyć spójny i estetyczny wygląd aplikacji.
- JavaScript - język skryptowym, który umożliwia dodanie interaktywności i dynamiczności do strony internetowej. JavaScript pozwala na tworzenie zaawansowanych interfejsów użytkownika, animacji, efektów wizualnych itp.

- PostgreSQL - relacyjna baza danych, która jest kompatybilna z Java i Spring. PostgreSQL oferuje wiele zalet m.in. takich jak duża wydajność, skalowalność, wsparcie dla transakcji i wiele innych.
- Serwer własny - serwer niezbędny jest do hostowania aplikacji webowej i umożliwia dostęp przez przeglądarkę internetową. Serwr własny pozwala na większą kontrolę nad aplikacją.

4. Scenariusze

Tytuł	Dodawanie grupy
Numer	Scenariusz 1
Aktorzy	Superadministrator
Stan wejścia	Aktor znajduje się w menu głównym
Przebieg	1. Aktor klika w menu przycisk z napisem "Grupy" 2. Aktor klika przycisk z ikonką plusa 3. Aktor uzupełnia dane grupy 4. Aktor zapisuje zmiany
Wynik	Zostaje stworzony nowa grupa
Scenariusz alternatywny	4.a Aktor wychodzi bez zapisywania 4.b Aktor chce utworzyć grupę o nazwie która już istnieje - pojawia się alert

Tab. 4.1. Scenariusz 1 - Tworzenie grupy

Tytuł	Edycja grupy
Numer	Scenariusz 2
Aktorzy	Superadministrator
Stan wejścia	Aktor znajduje się w menu głównym
Przebieg	1. Aktor klika w menu przycisk z napisem "Grupy" 2. Aktor klika przycisk z ikonką pióra na wybranej grupie 3. Aktor edytuje dane grupy 4. Aktor zapisuje zmiany
Wynik	Dane grupy zostają zmienione
Scenariusz alternatywny	4.a Aktor wychodzi bez zapisywania 4.b Aktor chce zmienić nazwę grupy na nazwe która już istnieje - pojawia się alert

Tab. 4.2. Scenariusz 2 - Edycja grupy

Tytuł	Usuwanie grupy
Numer	Scenariusz 3
Aktorzy	Superadministrator
Stan wejścia	Aktor znajduje się w menu głównym
Przebieg	1. Aktor klika w menu przycisk z napisem "Grupy" 2. Aktor klika przycisk z ikonką kosza na wybranej grupie 3. Aktor potwierdza usunięcie grupy
Wynik	Grupa zostaje usunięta
Scenariusz alternatywny	3.a Aktor anuluje usunięcie

Tab. 4.3. Scenariusz 3 - Usuwanie grupy

Tytuł	Dodanie administratora do grupy
Numer	Scenariusz 4
Aktorzy	Superadministrator
Stan wejścia	Aktor znajduje się w menu głównym
Przebieg	1. Aktor klika w menu przycisk z napisem "Grupy" 2. Aktor klika przycisk z ikonką pióra na wybranej grupie 3. Aktor klika przycisk z napisem "Dodaj administratora" 4. Aktor wybiera z listy użytkownika który ma zostać administratorem grupy 5. Aktor zapisuje zmiany
Wynik	Administrator zostaje dodany do grupy
Scenariusz alternatywny	4.a Aktor wychodzi bez zapisywania

Tab. 4.4. Scenariusz 4 - Dodanie administratora do grupy

Tytuł	Dodawanie użytkownika
Numer	Scenariusz 5
Aktorzy	Superadministrator, Administrator
Stan wejścia	Aktor znajduje się w menu głównym
Przebieg	1. Aktor klika w menu przycisk z napisem "Użytkownicy" 2. Aktor klika przycisk z ikonką plusa 3. Aktor uzupełnia dane użytkownika 4. Aktor zapisuje zmiany
Wynik	Zostaje stworzony nowy użytkownik
Scenariusz alternatywny	4.a Aktor wychodzi bez zapisywania 4.b Aktor chce utworzyć użytkownika o nazwie która już istnieje - pojawia się alert

Tab. 4.5. Scenariusz 5 - Dodawanie użytkownika

Tytuł	Usuwanie użytkownika
Numer	Scenariusz 6
Aktorzy	Superadministrator, Administrator
Stan wejścia	Aktor znajduje się w menu głównym
Przebieg	1. Aktor klika w menu przycisk z napisem "Użytkownicy" 2. Aktor klika przycisk z ikonką kosza na wybranym użytkowniku 3. Aktor potwierdza usunięcie użytkownika
Wynik	Użytkownik zostaje usunięty
Scenariusz alternatywny	3.a Aktor anuluje usunięcie

Tab. 4.6. Scenariusz 6 - Usunięcie użytkownika

Tytuł	Edycja użytkownika
Numer	Scenariusz 7
Aktorzy	Superadministrator, Administrator
Stan wejścia	Aktor znajduje się w menu głównym
Przebieg	1. Aktor klika w menu przycisk z napisem "Użytkownicy" 2. Aktor klika przycisk z piórem na wybranym użytkowniku 3. Aktor zmienia dane użytkownika 4. Aktor zapisuje zmiany
Wynik	Dane użytkownika zostają zmienione
Scenariusz alternatywny	4.a Aktor wychodzi bez zapisywania 4.b Aktor chce zmienić nazwę użytkonika na już istniejącą - pojawia się alert

Tab. 4.7. Scenariusz 7 - Edycja użytkownika

Tytuł	Dodawanie profilu
Numer	Scenariusz 8
Aktorzy	Superadministrator, Administrator, Użytkownik
Stan wejścia	Aktor znajduje się w menu głównym
Przebieg	1. Aktor klika w menu przycisk z napisem "Profile" 2. Aktor klika przycisk z ikonką plusa 3. Aktor uzupełnia dane profilu 4. Aktor zapisuje zmiany
Wynik	Zostaje stworzony nowy profil
Scenariusz alternatywny	4.a Aktor wychodzi bez zapisywania 4.b Aktor chce utworzyć profil o nazwie która już istnieje - pojawia się alert

Tab. 4.8. Scenariusz 8 - Dodawanie profilu

Tytuł	Edycja profilu
Numer	Scenariusz 9
Aktorzy	Superadministrator, Administrator, Użytkownik
Stan wejścia	Aktor znajduje się w menu głównym
Przebieg	1. Aktor klika w menu przycisk z napisem "Profile" 2. Aktor klika przycisk z piórem na wybranym profilu 3. Aktor zmienia dane profilu 4. Aktor zapisuje zmiany
Wynik	Dane profilu zostają zmienione
Scenariusz alternatywny	4.a Aktor wychodzi bez zapisywania 4.b Aktor chce zmienić nazwę profilu na już istniejącą - pojawia się alert

Tab. 4.9. Scenariusz 9 - Edycja profilu

Tytuł	Usuwanie profilu
Numer	Scenariusz 10
Aktorzy	Superadministrator, Administrator, Użytkownik
Stan wejścia	Aktor znajduje się w menu głównym
Przebieg	1. Aktor klika w menu przycisk z napisem "Profile" 2. Aktor klika przycisk z ikonką kosza na wybranym profilu 3. Aktor potwierdza usunięcie profilu
Wynik	Profil zostaje usunięty
Scenariusz alternatywny	3.a Aktor anuluje usunięcie

Tab. 4.10. Scenariusz 10 - Usunięcie profilu

Tytuł	Edycja playlisty
Numer	Scenariusz 11
Aktorzy	Superadministrator, Administrator, Użytkownik
Stan wejścia	Aktor znajduje się w menu głównym
Przebieg	1. Aktor klika w menu przycisk z napisem "Playlisty" 2. Aktor klika przycisk z piórem na wybranej playliście 3. Aktor zmienia dane playlisty 4. Aktor zapisuje zmiany
Wynik	Dane playlisty zostają zmienione
Scenariusz alternatywny	4.a Aktor wychodzi bez zapisywania 4.b Aktor chce zmienić nazwę playlisty na już istniejącą - pojawia się alert

Tab. 4.11. Scenariusz 11 - Edycja playlisty

Tytuł	Usuwanie playlisty
Numer	Scenariusz 12
Aktorzy	Superadministrator, Administrator, Użytkownik
Stan wejścia	Aktor znajduje się w menu głównym
Przebieg	1. Aktor klika w menu przycisk z napisem "Playlisty" 2. Aktor klika przycisk z ikonką kosza na wybranej playliście 3. Aktor potwierdza usunięcie playlisty
Wynik	Playlista zostaje usunięta
Scenariusz alternatywny	3.a Aktor anuluje usunięcie

Tab. 4.12. Scenariusz 12 - Usunięcie playlisty

Tytuł	Wysłanie wiadomości do superadministratora
Numer	Scenariusz 13
Aktorzy	Administrator, Użytkownik
Stan wejścia	Aktor znajduje się w menu głównym
Przebieg	1. Aktor klika w menu przycisk z napisem "Kontakt" 2. Aktor uzupełnia dane formularza 3. Aktor klika przycisk z napisem "Wyślij"
Wynik	Wiadomość zostaje wysłana
Scenariusz alternatywny	2.a Aktor podaje niepoprawne dane - pojawia się komunikat 3.a Aktor nie uzupełnił wymaganych pól - pojawia się alert

Tab. 4.13. Scenariusz 13 - Wysłanie wiadomości do superadministratora

Tytuł	Dodawanie slajdu
Numer	Scenariusz 14
Aktorzy	Superadministrator, Administrator, Użytkownik
Stan wejścia	Aktor znajduje się w menu głównym
Przebieg	1. Aktor klika w menu przycisk z napisem "Pieśni" 2. Aktor klika przycisk z ikonką pióra na wybranej pieśni 3. Aktor uzupełnia dane slajdu w edytorze 4. Aktor klika przycisk z napisem »" 5. Aktor zapisuje zmiany
Wynik	Zostaje dodany nowy slajd
Scenariusz alternatywny	5.a Aktor wychodzi bez zapisywania

Tab. 4.14. Scenariusz 14 - Dodawanie slajdu

Tytuł	Edycja slajdu
Numer	Scenariusz 15
Aktorzy	Superadministrator, Administrator, Użytkownik
Stan wejścia	Aktor znajduje się w menu głównym
Przebieg	1. Aktor klika w menu przycisk z napisem "Pieśni" 2. Aktor klika przycisk z piórem na wybranej pieśni 3. Aktor klika na wybrany slide 4. Aktor zmienia w edytorze dane slajdu 5. Aktor klika przycisk z napisem »" 6. Aktor zapisuje zmiany
Wynik	Zawartość slajdu zostaje zmieniona
Scenariusz alternatywny	6.a Aktor wychodzi bez zapisywania

Tab. 4.15. Scenariusz 15 - Edycja slajdu

Tytuł	Wyświetlenie pieśni według kategorii
Numer	Scenariusz 16
Aktorzy	Superadministrator, Administrator, Użytkownik
Stan wejścia	Aktor znajduje się w menu głównym
Przebieg	1. Aktor klika w menu przycisk z napisem "Pieśni" 2. Aktor wybiera z rozwijanej listy kategorie
Wynik	Na liście pojawiają się pieśni dla wybranej kategorii
Scenariusz alternatywny	-

Tab. 4.16. Scenariusz 16 - Wyświetlenie pieśni według kategorii

Tytuł	Wyszukanie pieśni
Numer	Scenariusz 17
Aktorzy	Superadministrator, Administrator, Użytkownik
Stan wejścia	Aktor znajduje się w menu głównym
Przebieg	1. Aktor klika w menu przycisk z napisem "Pieśni" 2. Aktor wpisuje w polu wyszukiwania fragment tytułu lub slajdu
Wynik	Na liście pojawiają się pieśni pasujące do wprowadzonego fragmentu
Scenariusz alternatywny	-

Tab. 4.17. Scenariusz 17 - Wyszukanie pieśni

Tytuł	Dodawanie playlisty
Numer	Scenariusz 18
Aktorzy	Superadministrator, Administrator, Użytkownik
Stan wejścia	Aktor znajduje się w menu głównym
Przebieg	1. Aktor klika w menu przycisk z napisem "Pieśni" 2. Aktor klika na pieśni które chce dodać do playlisty 3. Aktor uzupełnia tytuł playlisty 4. Aktor wybiera kategorie playlisty »" 5. Aktor zapisuje zmiany
Wynik	Zostaje utworzona nowa playlista
Scenariusz alternatywny	5.a Aktor wychodzi bez zapisywania

Tab. 4.18. Scenariusz 18 - Dodawanie playlisty

Tytuł	Dodawanie kategorii do pieśni
Numer	Scenariusz 19
Aktorzy	Superadministrator, Administrator, Użytkownik
Stan wejścia	Aktor znajduje się w menu głównym
Przebieg	1. Aktor klika w menu przycisk z napisem "Pieśni" 2. Aktor klika na przycisk z ikonką pióra na wybranej pieśni 3. Aktor klika na pole wyboru kategorii 4. Aktor zaznacza wybrane kategorie »" 5. Aktor zapisuje zmiany
Wynik	Do pieśni zostają przypisane kategorie
Scenariusz alternatywny	5.a Aktor wychodzi bez zapisywania

Tab. 4.19. Scenariusz 19 - Dodawanie kategorii do pieśni

Tytuł	Dodawanie pieśni
Numer	Scenariusz 20
Aktorzy	Superadministrator, Administrator, Użytkownik
Stan wejścia	Aktor znajduje się w menu głównym
Przebieg	1. Aktor klika w menu przycisk z napisem "Pieśni" 2. Aktor klika na przycisk z ikonką plusa 3. Aktor uzupełnia dane pieśni 4. Aktor zapisuje zmiany
Wynik	Zostaje utworzona nowa pieśń
Scenariusz alternatywny	4.a Aktor wychodzi bez zapisywania 4b. Aktor chce stworzyć pieśń z nazwą która już istniejącą - pojawia się alert

Tab. 4.20. Scenariusz 20 - Dodawanie pieśni

5. Estymacja czasowa

Łączny czas przeznaczoy na realizacje projektu: 3 miesiące (13 tygodni)

1. Stworzenie dokumentacji projektowej (3 tygodnie)

2. Stworzenie szablonu aplikacji web (1 tydzień)

3. Napisanie backendu wraz z testami jednostkowymi (3-4 tygodnie)
   - Implementacja struktury backendu oraz konfiguraja środowiska (2 dni)
   - Implementacja warstwy danych - stworzenie klas encji (DAO), obiektów transferu danych (DTO) oraz repozytoriów. (7 dni)
   - Implementacja warstwy serwisów - stworzenie klas serwisów, które będą wykorzystywać repozytoria i logikę biznesową. (7 dni)
   - Implementacja warstwy kontrolerów - stworzenie klas kontrolerów REST API, które będą obsługiwać zapytania HTTP i wywoływać odpowiednie metody serwisów. (7 dni)
   - Implementacja testów jednostkowych - pisanie testów dla każdej z klas w powyższych warstwach w celu sprawdzenia ich poprawności i zgodności z wymaganiami. Pisanie testów jednostkowych będzie wykonywane na bieżąco z implementacją warstwy danych, serwisów i kontrolerów. (razem z powyższymi)
   - Testowanie i ewentualne poprawki (2 dni)

4. Napisanie frontendu (2-3 tygodnie)
   - Implementacja struktury frontendu oraz konfiguracja środowiska (2 dni)
   - Implementacja komponentów interfejsu użytkownika (10 dni)
   - Integracja z backendem (2 dni)
   - Testowanie i ewentualne poprawki (3 dni)

5. Tesy integracyjne (1 tydzień)
   - Testowanie poszczególnych funkcjonalności (logowanie, dodanie elementu itp.)
   - Testowanie integracji między frontendem a backendem
   - Testowanie wydajnościowe (np. obciążenie serwera)
   - Testowanie bezpieczeństwa (np. ataki typu SQL injection, XSS)

6. Wdrożenie oraz testowanie manualne i naprawa potencjalnych błędów (2 tygodnie)

6. Implementacja

- Repozytorium GitHub
- Szablon aplikacji

W ramach implementacji projektu można wyróżnić kilka głównych etapów. Do najważniejszych należą:

1. Stworzenie szablonu aplikacji - ten etap obejmuje zaprojektowanie i stworzenie podstawowej struktury aplikacji, która będzie stanowiła bazę dla kolejnych funkcjonalności.

   Stworzenie szablonu aplikacji będzie realizowane przy użyciu narzędzia Figma. Szablon zostanie stworzony z elementów, które zostaną dostosowane do potrzeb aplikacji.

   Szablon aplikacji powinien być responsywny, ale głównie dedykowany dla przeglądarek. Jego styl powinien być zgodny z aktualnymi trendami, prosty w obsłudze i estetyczny.

   Ważne jest, aby szablon był łatwy w wdrożeniu zarówno w frontendzie, jak i w połączeniu z backendem.

   Projekt graficzny aplikacji powinien zawierać następujące strony:

   - Panel logowania
   - Odzyskiwanie hasła
   - Błąd 404
   - Strona główna
   - Piesni
   - Tworzenie pieśni
   - Playlisty
   - Tworzenie playlist
   - Kategorie
   - Profile
   - Ustawienia
   - Kontakt
   - Użytkownicy
   - Grupy

2. Implementacja backendu - to proces tworzenia logiki biznesowej i interakcji z bazą danych. Backend jest oparty na języku Java, a w trakcie jego tworzenia pisane są również testy jednostkowe, które pozwalają na sprawdzenie poprawności działania poszczególnych elementów systemu.

   Do tworzenia backendu wykorzystany zostanie również Spring Framework.

   Najważniejsze elementy backendu:

   - klasy DAO
   - klasy DTO
   - klasy repozytoriów
   - klasy serwisów
   - klasy kontrolerów
   - testy jednostkowe

   Wszystkie te klasy oraz testy muszą być wykonane dla każdej z tabel.

3. Implementacja frontendu - w tym etapie tworzona jest warstwa prezentacji, czyli interfejs użytkownika. Frontend jest głównie oparty na technologii React, a jego zadaniem jest umożliwienie użytkownikom korzystanie z aplikacji w sposób intuicyjny i efektywny.

   Poza Reactem może tutaj spotkać również HTML, CSS oraz JavaScript.

   Frontend będzie napisany na podstawie stworzonego szablonu.

   Najważniejsze komponenty jakie pojawią się w frontendzie:

   - formularze
   - przyciski
   - nawigacja
   - listy
   - pola "Select"
   - pola "Text Field"
   - pole "Textarea"
   - ikonki
   - edytor
   - lista "Drap And Drop"

- lista "Transfer List"
- przyciski "Checkbox"
- przyciski "Switch"
- suwaki "Slider"
- narzędzie do wybierania kolorów
- zdjęcia

Po zaimplemenentowaniu frontendu następuje jego integracja z backendem oraz przeprowadzenie testów.

4. Napisanie testów integracyjnych - ten etap ma na celu sprawdzenie, czy poszczególne komponenty systemu działają ze sobą prawidłowo i poprawnie przekazują informacje między sobą.

   W tym etapie warto rówież przeprowadzić testy wydajnościowe oraz testy bezpieczeństwa.

5. Przeprowadzenie testów manualnych - mają one na celu sprawdzenie, czy system działa zgodnie z oczekiwaniami użytkowników i czy interakcja z nim jest intuicyjna i łatwa.

6. Wdrożenie aplikacji - ostatnim etapem projektu jest umieszczenie aplikacji na serwerze i uruchomienie go w środowisku produkcyjnym.

7. Testy i ich wyniki

7.1. Rodzaje Testów

W ramach realizacji projektu "System Zarządzania Pieśniami"wykorzystano następujące rodzaje testów:

7.1.1. Testy Jednostkowe (Unit Tests)

Testy jednostkowe były stosowane do sprawdzania poprawności funkcji i metod w kodzie backendowym napisanym w języku Java. Testy jednostkowe koncentrowały się na izolowanych częściach kodu, aby upewnić się, że każda funkcja działa poprawnie w różnych scenariuszach. Łącznie wykonano 158 testów jednostkowych.

7.1.2. Testy Manualne

Poza testami jednostkowymi, przeprowadzono również testy manualne. Testy te były przeprowadzane na bieżąco podczas rozwijania projektu. Polegały one na interakcji z aplikacją w sposób, w jaki typowy użytkownik by z niej korzystał, co pozwoliło na ocenę i zrozumienie, jak różne elementy systemu współdziałają ze sobą. Wykorzystując manualne testowanie, sprawdzono poprawność działania interfejsu użytkownika (front-endu) stworzonego przy użyciu technologii React, a także poprawność operacji na bazie danych PostgreSQL.

7.2. Pokrycie Kodu Testami

Statystyki pokrycia dla wszystkich klas wyglądają następująco:

- Pokrycie Klas: 93.2% (109 z 117 klas zostało przetestowanych)
- Pokrycie Metod: 76.2% (433 z 568 metod zostało przetestowanych)
- Pokrycie Linii Kodu: 74.9% (1235 z 1648 linii kodu zostało przetestowanych (Kod zawiera o wiele więcej linii te tutaj są jedynie tymi które narzędzie do analizy uznało za warte przetestowania))

W analizie możemy zauważyć następujące pokrycie kodu dla wybaanych pakietów:

- pl.cantabo: Pokrycie klas, metod i linii kodu wynosi odpowiednio 100%, 50% i 50%.
- pl.cantabo.configuration: Wszystkie klasy, metody i linie kodu w tym pakiecie są pokryte testami (100%).

- pl.cantabo.controller: Pokrycie klas, metod i linii kodu wynosi odpowiednio 90.9%, 54.7% i 41.4%.

- pl.cantabo.database.group: Pokrycie klas, metod i linii kodu wynosi odpowiednio 100%, 76.7% i 82.8%.

Podobne statystyki powtarzają się dla pozostałych pakietów.

Wyjątkowo wysokie pokrycie testami osiągnęły pakiety pl.cantabo.initializer oraz pl.cantabo.validator.email i pl.cantabo.validator.password, gdzie wszystkie klasy, metody i linie kodu są pokryte testami (100%).

Podsumowując, statystyki pokrycia kodu testami są stosunkowo wysokie dla większości pakietów.

Current scope: all classes

Overall Coverage Summary

Package	Class, %	Method, %	Line, %
all classes	93,2% (109/117)	76,2% (433/568)	74,9% (1235/1648)

Coverage Breakdown

Package	Class, %	Method, %	Line, %
pl.cantabo	100% (1/1)	50% (1/2)	50% (1/2)
pl.cantabo.auditor	100% (1/1)	20% (1/5)	20% (1/5)
pl.cantabo.configuration	100% (1/1)	100% (3/3)	100% (3/3)
pl.cantabo.controller	90,9% (10/11)	54,7% (35/64)	41,4% (77/186)
pl.cantabo.database.group	100% (6/6)	76,7% (23/30)	82,8% (48/58)
pl.cantabo.database.playlist	100% (6/6)	75,9% (22/29)	76,6% (49/64)
pl.cantabo.database.playlist.playlistCategory	100% (6/6)	81,8% (18/22)	82,5% (33/40)
pl.cantabo.database.profile	100% (6/6)	97,6% (83/85)	97% (161/166)
pl.cantabo.database.settings	100% (6/6)	87,5% (21/24)	86,4% (38/44)
pl.cantabo.database.slide	100% (6/6)	82,8% (24/29)	85,2% (46/54)
pl.cantabo.database.song	100% (6/6)	78,4% (29/37)	83,8% (62/74)
pl.cantabo.database.song.songCategory	100% (6/6)	86,4% (19/22)	85% (34/40)
pl.cantabo.database.song.songHistory	20% (1/5)	8,3% (1/12)	6,7% (1/15)
pl.cantabo.database.user	100% (7/7)	97,7% (43/44)	96,4% (81/84)
pl.cantabo.initializer	100% (11/11)	100% (22/22)	100% (404/404)
pl.cantabo.security	75% (3/4)	47,4% (9/19)	69,8% (30/43)
pl.cantabo.service	100% (9/9)	69,8% (44/63)	45,3% (82/181)
pl.cantabo.utils	100% (1/1)	50% (1/2)	50% (1/2)
pl.cantabo.utils.exception	100% (1/1)	50% (1/2)	50% (1/2)
pl.cantabo.utils.jwt	60% (3/5)	25% (4/16)	15,1% (8/53)
pl.cantabo.utils.password	100% (1/1)	100% (2/2)	100% (3/3)
pl.cantabo.validator	100% (7/7)	68,2% (15/22)	48,1% (50/104)
pl.cantabo.validator.email	100% (2/2)	100% (6/6)	100% (11/11)
pl.cantabo.validator.password	100% (2/2)	100% (6/6)	100% (10/10)

generated on 2023-06-01 07:12

Rys. 7.1. Pokrycie kodu testami

7.3. Analiza metryk

7.3.1. Złożoność Cyklomatyczna

Złożoność cyklomatyczna jest miarą ilości ścieżek wykonania, które są dostępne w danym bloku kodu. Można to również interpretować jako miarę ilości decyzji, które muszą być podjęte podczas wykonania kodu. Ta miara jest używana do oceny złożoności kodu. Analiza złożoności cyklomatycznej kodu źródłowego pokazała następujące wyniki:

Średnia Złożoność Cyklomatyczna

- Moduł cantabo.main: Średnia złożoność cyklomatyczna wynosi 1.417. To oznacza, że średnio w każdym bloku kodu w tym module jest 1.417 decyzji, które mogą wpływać na ścieżkę wykonania.

- Moduł cantabo.test: Średnia złożoność cyklomatyczna wynosi 1.006. To oznacza, że średnio w każdym bloku kodu w tym module jest 1.006 decyzji, które mogą wpływać na ścieżkę wykonania.

Cały projekt: Średnia złożoność cyklomatyczna wynosi 1.210. To oznacza, że średnio w każdym bloku kodu w całym projekcie jest 1.210 decyzji, które mogą wpływać na ścieżkę wykonania.

Całkowita Złożoność Cyklomatyczna

- Moduł cantabo.main: Całkowita złożoność cyklomatyczna wynosi 248. Oznacza to, że w całym tym module jest 248 decyzji, które mogą wpływać na ścieżkę wykonania kodu.

- Moduł cantabo.test: Całkowita złożoność cyklomatyczna wynosi 179. Oznacza to, że w całym tym module jest 179 decyzji, które mogą wpływać na ścieżkę wykonania kodu.

- Cały projekt: Całkowita złożoność cyklomatyczna wynosi 427. Oznacza to, że w całym projekcie jest 427 decyzji, które mogą wpływać na ścieżkę wykonania kodu.

Podsumowując, złożoność cyklomatyczna w tym projekcie jest stosunkowo niska, co sugeruje, że kod jest dobrze zaprojektowany i łatwy do zrozumienia. To jest pozytywny wskaźnik dla przyszłego utrzymania i rozbudowy systemu. Niemniej jednak, zawsze warto dążyć do minimalizacji złożoności cyklomatycznej, aby ułatwić czytanie i testowanie kodu.

7.3.2. Metryki klas i metod

Analiza metryk klasy i metod dała następujące wyniki:

Metryki Klasy:

- Średnia Złożoność Obiektowa (OCavg): 1.12 (średnia liczba decyzji, które muszą być podjęte podczas czytania kodu klasy)
- Maksymalna Złożoność Obiektowa (OCmax): 1.32 (największa liczba decyzji, które muszą być podjęte podczas czytania kodu jakiejkolwiek klasy)
- Średnia Suma Wag Metod Klasy (WMC): 2.58 (średnia liczba różnych ścieżek, które mogą być przebywane przez kod w klasie)

Metryki Metod:

- Całkowita złożoność poznawcza (CogC): 83 (całkowita liczba decyzji, które muszą być podjęte podczas czytania kodu wszystkich metod)
- Całkowita złożoność wywołania krawędzi (ev(G)): 370 (całkowita liczba krawędzi, które prowadzą do każdej metody)
- Całkowita złożoność wejścia krawędzi (in(G)): 419 (całkowita liczba krawędzi, które prowadzą z każdej metody)
- Całkowita złożoność cyklomatyczna (v(G)): 427 (całkowita liczba różnych ścieżek, które mogą być przebywane przez kod we wszystkich metodach)
- Średnia złożoność poznawcza (CogC): 0.24 (średnia liczba decyzji, które muszą być podjęte podczas czytania kodu metody)
- Średnia złożoność wywołania krawędzi (ev(G)): 1.05 (średnia liczba krawędzi, które prowadzą do metody)
- Średnia złożoność wejścia krawędzi (in(G)): 1.19 (średnia liczba krawędzi, które prowadzą z metody)
- Średnia złożoność cyklomatyczna (v(G)): 1.21 (średnia liczba różnych ścieżek, które mogą być przebywane przez kod w metodzie)

Podsumowując, te statystyki sugerują, że kod jest dobrze zorganizowany, z umiarkowaną złożonością obiektową i metod. Dobre wyniki w tych metrykach mogą świadczyć o wysokiej jakości kodu, łatwości utrzymania i niskim ryzyku błędów.

7.4. Proces Testowania

1. **Testowanie manualne i poprawki na bieżąco**: Podczas procesu tworzenia projektu, każda nowa funkcja była systematycznie testowana manualnie. Dzięki temu, jakiekolwiek problemy, które wystąpiły, były naprawiane na bieżąco. Gdy naprawa nie była możliwa od razu, problem był dodawany do listy zadań, aby być później rozwiązanym. Ten proces pozwolił na utrzymanie projektu na najwyższym możliwym poziomie jakości w miarę jego rozwijania.

2. **Testowanie przez osoby trzecie**: Po zakończeniu prac nad projektem, aplikacja została przetestowana przez osobę niezwiązaną z projektem. Pozwoliło to na zidentyfikowanie potencjalnych problemów, które mogły zostać przeoczone podczas testowania przez zespół projektowy, oraz na zapewnienie, że aplikacja jest przyjazna dla użytkowników i intuicyjna w obsłudze.

3. **Testy kompatybilności przeglądarki**: Testowanie było przeprowadzane na różnych przeglądarkach, aby upewnić się, że aplikacja działa poprawnie niezależnie od używanej przeglądarki. Dzięki temu mogliśmy zapewnić kompatybilność aplikacji z najpopularniejszymi przeglądarkami i zapewnić, że różni użytkownicy będą mieli takie samo doświadczenia.

4. **Testy równoległe/multisesyjne**: Testowaliśmy także możliwość jednoczesnego uruchamiania wielu sesji aplikacji.

7.5. Podsumowanie

Podczas realizacji projektu przeprowadzono różnego rodzaju testy, w tym testy jednostkowe i manualne. Te testy pozwoliły na sprawdzenie poprawności funkcji i metod w kodzie, a także na ocenę interakcji między różnymi elementami systemu.

Statystyki pokrycia kodu testami były stosunkowo wysokie dla większości pakietów, co wskazuje na solidne testowanie kodu. W szczególności, pokrycie klas wyniosło 93.2%, pokrycie metod 76.2%, a pokrycie linii kodu 74.9%.

Z analizy metryk wynika, że złożoność cyklomatyczna w projekcie jest stosunkowo niska, co sugeruje dobrze zaprojektowany i łatwy do zrozumienia kod. Wskaźniki złożoności obiektowej i metody również pokazują, że kod jest dobrze zorganizowany.

W trakcie procesu testowania, każda nowa funkcja była systematycznie testowana manualnie, a napotkane problemy były naprawiane na bieżąco. Dodatkowo, aplikacja została przetestowana przez osobę trzecią. Przeprowadzono również testy kompatybilności przeglądarki i testy równoległe/multisesyjne.

8. Podsumowanie i bilans

(MVP vs rzeczywistość)

8.1. Analiza efektywności pracy

Realizacja tego projektu była wyjątkowym wyzwaniem, które wymagało zarówno precyzyjnego planowania, jak i dostosowywania się do zmieniających się okoliczności.

Szymon:

Szymon poświęcił na początkowej fazie projektu, w marcu, 40 godzin na dokumentację i 20 godzin na tworzenie szablonu strony. Zakładając, że na realizację swojej części projektu potrzebuje około 20 godzin tygodniowo przez kwiecień i maj, czyli łącznie 180 godzin, był przygotowany na możliwość konieczności nadrabiania różnic wynikających z dynamicznego charakteru pracy zespołowej. Ostatecznie, Szymon poświęcił na część związaną z pisaniem kodu 183 godziny i 40 minut, realizując 74 comity na backendzie i 57 na froncie, co w sumie przekładało się na około 17000 linii kodu. Łączny czas poświęcony na realizaje projektu wyniósł więc 243 godziny i 40 minut. Pomimo wcześniejszego doświadczenia z Javą, Szymon musiał nauczyć się nowego języka, React, i pomimo tego wyzwania, udało mu się zrealizować wszystkie przypisane mu zadania, a także część zadań, które pierwotnie nie były do niego przypisane.

Łączny czas spędzony przy realizacji projektu: 243h 40m

Kuba:

Kuba początkowo szacował, że na realizację swojej części projektu potrzebuje około 160-170 godzin. W rzeczywistości poświęcił na projekt 162 godziny i 30 minut. Mimo braku wcześniejszego doświadczenia z Javą i React, udało mu się zrealizować znaczną część przypisanych mu zadań. Wykonał 23 comity na backendzie i 4 na froncie, dodając łącznie około 3300 linii kodu.

Łączny czas spędzony przy realizacji projektu: 160h 30m

8.1.1. Podsumowanie

Podsumowując, realizacja projektu była intensywnym procesem, który wymagał dużej ilości czasu i wysiłku. Łącznie, członkowie zespołu poświęcili na projekt 397 godzin i 10 minut, co pokazuje skalę zaangażowania w ten projekt. Udało się zrealizować większość zadań, a doświadczenia z tego projektu na pewno posłużą w przyszłości. Różnice w zakresie realizacji zadań pomiędzy członkami zespołu wynikały z różnych przewidywań i możliwości w kontekście zarządzania projektem.

8.2. Porównanie założeń z końcowym produktem:

W wymaganiach funkcjonalnych zostały dodane oznaczenia:

- + zrealizowane
- +/- zrealizowane po części
- - nie zrealizowane

8.2.1. Porównanie dla MVP:

Funkcje kluczowe, oznaczone jako MVP, zostały zrealizowane niemalże w całości, co jest dużym sukcesem. Przez cały zakres, od dostępu do list pieśni, kategorii i playlist, aż po zarządzanie nimi, wszystko funkcjonuje zgodnie z założeniami. Funkcjonalność wyszukiwania pieśni została zrealizowana częściowo, czyli umożliwia wyszukiwanie po tytule.

8.2.2. Porównanie dla wszystkich funkcji:

Większość z założonych funkcjonalności została zrealizowana z powodzeniem. Wszystko, od logowania się do systemu, dostępu do list pieśni, playlist, profili, aż po dostęp do sekcji kontaktowej, funkcjonuje zgodnie z planem. Dodatkowo, wszelkie funkcje związane z zarządzaniem pieśniami, kategoriami, slajdami, playlistami i profilami są w pełni funkcjonalne.

Nie zrealizowano pełnej funkcjonalności dostępu do ustawień konta oraz możliwości usunięcia pojedynczych slajdów z playlisty. Funkcja zmiany kolejności pieśni w playliście również nie została zaimplementowana. W przypadku Superadministratora, nie zostały zrealizowane funkcje dotyczące zarządzania grupami.

8.2.3. Dodatkowe funkcje:

Warto podkreślić, że w trakcie tworzenia produktu zidentyfikowano i zrealizowano dodatkowe funkcje, które początkowo nie były planowane, ale okazały się istotne dla użytkowników. Są to między innymi alerty, czyszczenie formularzy po kliknięciu przycisku, a także strona główna ze statystykami.

8.2.4. Podsumowanie

Podsumowując, projekt udało się zrealizować w zadowalającej mierze. Pierwotne cele i założenia były ambitne, ale przemyślane, co zaowocowało produktem, który spełnia wiele z

nich. Z sukcesem zrealizowano większość zaplanowanych funkcjonalności, w tym wszystkie kluczowe elementy oznaczone jako MVP. Są one podstawą dla użytkowników i gwarantują użyteczność systemu.

Jednakże, jak w każdym projekcie, napotkano na pewne wyzwania, które w niektórych przypadkach opóźniły lub utrudniły pełną realizację niektórych funkcji. Mimo tych drobnych przeszkód, projekt pozostał na właściwej ścieżce i większość funkcji została zaimplementowana zgodnie z planem.

Ponadto, w procesie tworzenia oprogramowania zidentyfikowano i zaimplementowano dodatkowe funkcje, które początkowo nie były planowane, ale okazały się istotne dla użytkowników.

Ogólnie rzecz biorąc, mimo niewielkiej liczby funkcji, które nie zostały zrealizowane lub są tylko częściowo zrealizowane, projekt można uznać za udany. Znaczna większość założonych funkcjonalności została zrealizowana, a zdolność do adaptacji i ewolucji projektu wskazuje na jego ogólną solidność i skuteczność. Z całą pewnością stanowi to mocną podstawę do dalszych działań, które mogą być podjęte w przyszłości.

9. Ważne liniki

- Repozytorium Backend
- Repozytorium Frontend
- Dokumentacja uruchomienia aplikacji

10. Kontakt

- Szymon Pogwizd szymonpogwizd12@gmail.com
- Jakub Poręba jakubporeba8@gmail.com

11. Literatura

Bibliografia zawiera artykuły czytane podczas tworzenia dokumentacji.

Bibliografia

[1] *Wymagania funkcjonalne aplikacji* (https://studiosoftware.pl/blog/wymagania-funkcjonalne-aplikacji-jak-zapanowac-nad-budzetem-i-przebiegiem-projektu/).

[2] *Czym są wymagania funkcjonalne* (https://visuresolutions.com/pl/blog/functional-requirements/).

[3] *Wymagania niefunkcjonalne aplikacji* (https://studiosoftware.pl/blog/wymagania-niefunkcjonalne-aplikacji-czym-sa-i-jak-je-poprawnie-opisac-w-specyfikacji-projektu/).

[4] *Wymagania niefunkcjonalne* (https://studiosoftware.pl/blog/wymagania-niefunkcjonalne-aplikacji-czym-sa-i-jak-je-poprawnie-opisac-w-specyfikacji-projektu/).

Spis rysunków

Spis tabel