KONCERTY

Dokumentacja projektowa PZSP2

Wersja 3

25.11.2024r.

Semestr 2024Z

Zespół nr 14 w składzie:
Marcin Bagnowski
Aleksandra Buczma
Krzysztof Gólcz
Zofia Jasina

Mentor zespołu: dr inż. Krystian Radlak
Właściciel tematu: mgr inż. Katarzyna Nałęcz-Charkiewicz

Spis treści

1 Wprowadzenie

1.1 Cel projektu (z dokumentu od właściciela)

Projekt ma na celu stworzenie aplikacji internetowej wspomagającej zarządzanie zespołami muzycznymi oraz orkiestrami. Użytkownikami aplikacji będą członkowie orkiestr/zespołów oraz ich dyrygenci/kapelmistrzowie/chórmistrzowie (konieczny podział uprawnień użytkowników).

1.2 Wstępna wizja projektu

Aplikacja do zarządzania zespołami muzycznymi i orkiestrami ma na celu wsparcie grup muzycznych. Jej główne zadanie to ułatwienie komunikacji wewnątrz zespołu, koordynacji prób, koncertów oraz dostęp do zasobów, takich jak partytury czy nagrania.

Kluczowym elementem aplikacji jest system zarządzania uprawnieniami, który pozwala na przypisanie ról użytkowników zgodnie z ich pozycją w zespole. Dyrygenci, kapelmistrzowie i chórmistrzowie będą mieli pełne uprawnienia, umożliwiające im zarządzanie składem zespołu, zasobami muzycznymi, harmonogramem wydarzeń oraz ogłoszeniami. Będą mogli organizować np.: próby ogólne, jak również te dedykowane dla poszczególnych podgrup, takich jak instrumenty dęte, smyczkowe, czy chóry. Członkowie zespołów będą mieć dostęp do materiałów muzycznych przypisanych do ich roli oraz szczegółów dotyczących terminów prób i koncertów. Koordynatorzy z kolei, w odróżnieniu od zwykłych użytkowników, będą mogli zarządzać komunikacją poprzez dodawanie ogłoszeń.

Aplikacja zapewnia dystrybucję materiałów muzycznych, w której dyrygenci będą mogli udostępniać partytury, nagrania i notatki, dostosowane do potrzeb poszczególnych grup instrumentalnych oraz indywidualnych muzyków. Dzięki temu członkowie zespołów będą mieli stały dostęp do aktualnych zasobów, co umożliwi im samodzielne przygotowanie się przed próbami. System powiadomień przypomni o nadchodzących wydarzeniach, nowych materiałach czy zmianach w harmonogramie, umożliwiając bieżące śledzenie wszelkich aktualizacji.

Komunikacja w ramach aplikacji odbywać się będzie poprzez system ogłoszeń wewnętrznych, umożliwiający szybki kontakt między członkami zespołu a dyrygentem. Będzie to szczególnie przydatne w sytuacjach wymagających szybkich zmian lub dostosowania materiałów do potrzeb konkretnego koncertu.

Aplikacja będzie dostępna przez przeglądarkę internetową - przystosowana także pod kątem urządzeń mobilnych, dzięki czemu użytkownicy będą mogli korzystać z niej zarówno na komputerach, jak i smartfonach.

2 Metodologia wytwarzania

Podstawowym narzędziem pracy zespołowej jest repozytorium kodu na wydziałowym gitlabie:

- https://gitlab-stud.elka.pw.edu.pl/kgolcz/pzsp2-megalodony

Będzie ono wykorzystywane do przechowywania plików projektowych i wspólnej pracy nad nimi, a także do definiowania i przydzielania zadań. Przydzielanie zadań odbywać się będzie na bieżąco, biorąc pod uwagę aktualne możliwości oraz umiejętności członków zespołu.

Komunikacja między członkami zespołu odbywa się przy pomocy narzędzi takich jak Messenger oraz Discord. Do kontaktu z mentorem, klientem oraz ekspertami używane są także platformy Teams i Leon.

Platforma Miro służy jako przestrzeń robocza do tworzenia modeli i diagramów.

W celu określenia wrodzonych predyspozycji członków zespołu, przeprowadzony został test Belbina. Wynika z niego, że w zespole występują zróżnicowane role (3x Completer Finisher, 2x Shaper, Implementer, Monitor Evaluator, Plant, 2x Coordinator, Team Worker). Dotkliwym brakiem może okazać się brak naturalnego Resource Investigator-a.

Zgodnie z modelem kaskadowym pracy nad projektem, implementacja rozwiązania poprzedzona jest planowaniem, analizą wymagań oraz modelowaniem, a nastąpi po niej faza testowania.

3 Analiza wymagań

3.1 Wymagania użytkownika i biznesowe

- wymagania biznesowe:

Klient potrzebuje aplikacji internetowej wspomagającej zarządzanie zespołami muzycznymi oraz orkiestrami, aby ułatwić komunikację oraz dystrybucję zasobów między członkami.

- wymagania użytkowe:

Wymagania użytkownika, dostarczone przez klienta, prezentuje tabela 1.

Priorytet	Rola	Wymaganie
High	Kierownik grupy	Zarządzanie grupą (nazwa, opis, informacje dodatkowe takie jak link do strony internetowej czy profilów w mediach społecznościowych).
High	Kierownik grupy	Zarządzanie użytkownikami, w tym tworzenie grup i podgrup, zapraszanie nowych użytkowników do grup/podgrup.
High	Kierownik grupy	Wysyłanie ogłoszeń: do całej grupy, do podgrupy, do wybranych członków.
High	Kierownik grupy	Możliwość integracji ogłoszeń z mailem, co powoduje wysłanie ogłoszenia także na adresy e-mail członków grupy/podgrupy.
High	Kierownik grupy	Zarządzanie utworami (tytuł, opis, pliki z nutami, w tym możliwość dodawania plików per instrument, opcjonalne przypisanie do koncertu/koncertów).
High	Kierownik grupy	Planowanie koncertów i prób (tytuł, opis, repertuar, czas, miejsce, wybór całej grupy/podgrupy, bądź poszczególnych użytkowników).
High	Zwykły użytkownik	Oglądanie planu prób.
High	Zwykły użytkownik	Oglądanie planu koncertów.
High	Zwykły użytkownik	Przeglądanie repertuaru i pobieranie nut.
High	Zwykły użytkownik	Przeglądanie kalendarza wydarzeń.
Medium	Zwykły użytkownik	Przeszukiwanie repertuaru wg różnych kryteriów: nazwa utworu, koncert, do którego utwór jest przypisany, rodzaj instrumentu.
Low	-	Integracja kalendarza wydarzeń z Google Calendar.

Tab. 1. Wymagania użytkownika

- wymagania systemowe:

opisane przez wymagania funkcjonalne i niefunkcjonalne, patrz podpunkt 3.2.

- **aktorzy w projekcie:**

- główni: UŻYTKOWNIK (trzy poziomy uprawnień)
  - kierownik grupy (administrator, kapelmistrz, chórmistrz)
  - moderator (koordynator)
  - zwykły użytkownik (muzyk)
- drugorzędni: SYSTEMY ZEWNĘTRZNE
  - Google Sign-In API
  - Google Calendar
  - e-mail API
  - Google Drive API
  - Google Drive

3.2 Wymagania funkcjonalne i niefunkcjonalne

- wymagania funkcjonalne:

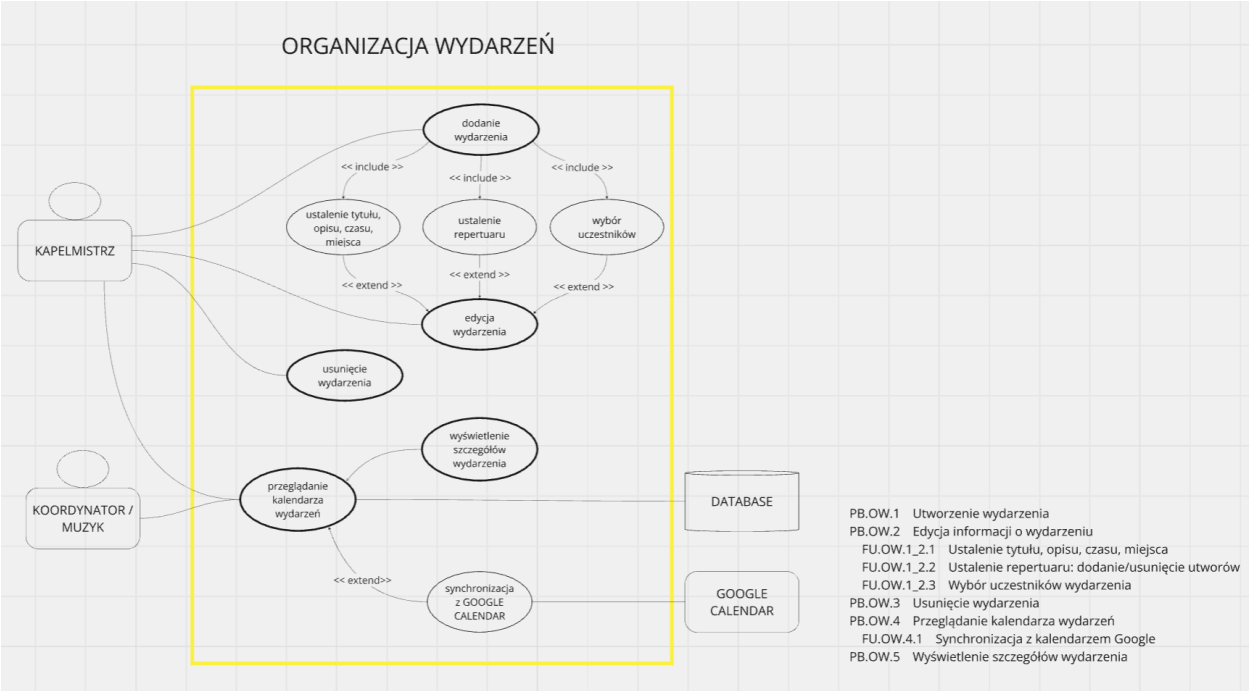
- Aplikacja powinna umożliwiać administratorom tworzenie grup i podgrup.
- Aplikacja powinna umożliwiać administratorom zarządzanie członkostwem użytkowników.
- Aplikacja powinna umożliwiać zarządzanie nazwą, opisem i dodatkowymi informacjami grup i podgrup (np. link do strony internetowej, profil w mediach społecznościowych).
- System powinien umożliwiać zapraszanie nowych użytkowników do grup i podgrup.
- Aplikacja powinna pozwolić administratorom i moderatorom na wysyłanie ogłoszeń do całej grupy, wybranej podgrupy lub pojedynczych członków.
- System powinien zapewnić wyświetlanie zamieszczonych ogłoszeń.
- System zapewni możliwość wysyłania powiadomień mailowych członkom grup/podgrup o nowym ogłoszeniu.
- System powinien umożliwiać dodawanie utworów muzycznych z nazwą, opisem, plikami nut oraz nagraniami do utworów.
- System powinien obsługiwać plik w formatach .mp4, .pdf, .m4a, .mid, .mus, .txt, .png, .musx, .jpg, .zip, .doc, .mp3, .wav, .wma, .tif.
- System powinien umożliwiać przypisanie pliku nut dla poszczególnych instrumentów oraz do poszczególnych użytkowników.
- System powinien umożliwiać przypisanie utworów do jednego lub kilku koncertów.
- Aplikacja powinna umożliwiać tworzenie wydarzeń takich jak koncerty i próby.
- Aplikacja zapewni zarządzanie wydarzeniami poprzez dodanie tytułu, opisu, repertuaru, daty, czasu i miejsca.
- System zapewni administratorowi możliwość wyboru uczestników wydarzenia (cała grupa, podgrupa lub poszczególni członkowie).
- System powinien udostępniać widok planu prób oraz planu koncertów.
- Aplikacja powinna umożliwiać członkom przeglądanie repertuaru wydarzenia.
- System zapewnia filtrowanie katalogu utworów według nazwy utworu, koncertu, do którego utwór jest przypisany, oraz instrumentu.
- Aplikacja powinna umożliwiać pobieranie przypisanych nut uprawnionym członkom zespołu.
- System powinien posiadać funkcję kalendarza, pozwalającą użytkownikom przeglądać wszystkie zaplanowane wydarzenia.
- System powinien umożliwiać synchronizację kalendarza wydarzeń z Google Calendar.

- wymagania niefunkcjonalne:

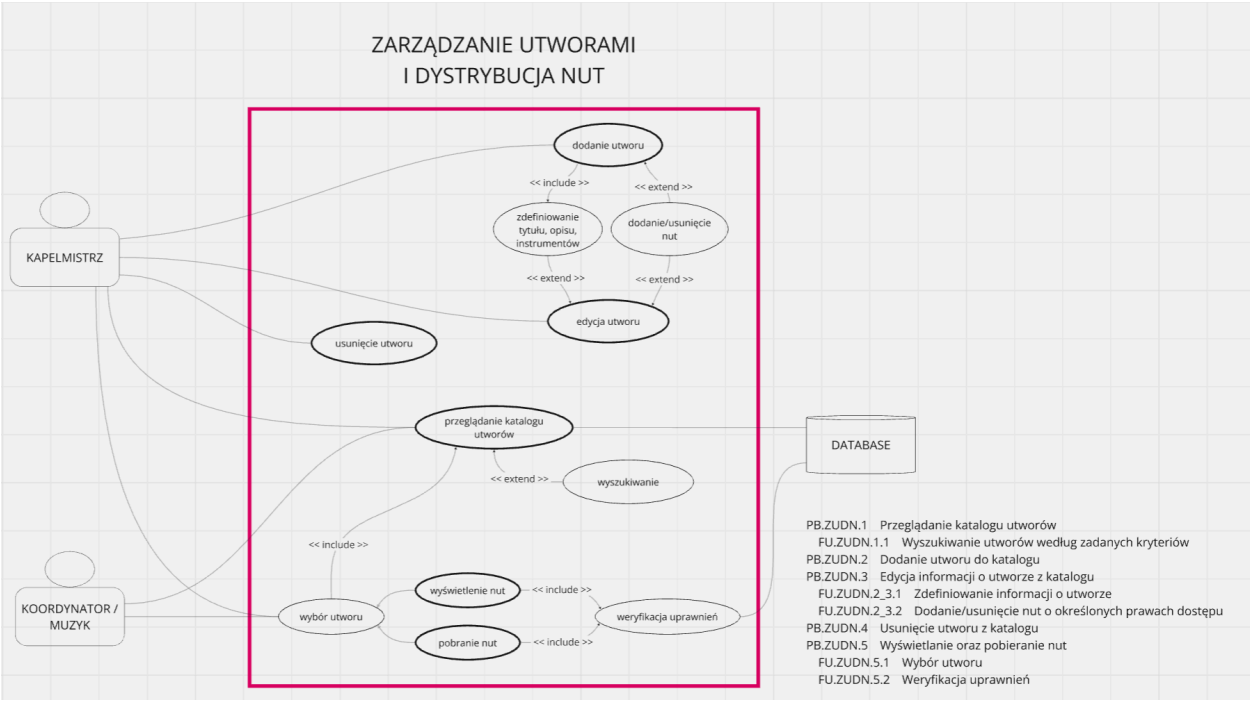
- Przenośność: Aplikacja powinna umożliwiać dostęp z różnych urządzeń i systemów operacyjnych, takich jak Windows, Linux, iOS i Android..
- Optymalizacja dla urządzeń mobilnych: Interfejs i wydajność aplikacji powinny być dostosowane do użytkowników korzystających z urządzeń mobilnych.
- Skalowalność: Aplikacja powinna wspierać rozbudowę i sprawne działanie dla 1000 użytkowników oraz kilku orkiestr (grup).
- Wydajność: Aplikacja powinna działać płynnie i bez zakłóceń nawet przy obciążeniu rzędu setek aktywnych użytkowników.
- Pojemność: Baza danych zapewni miejsce na pliki o łącznym rozmiarze do 15 GB (łącznie dla kilku grup, przy założeniu obsługi 5 grup każda po 2,6 GB średnio).
- Bezpieczeństwo: System powinien zapewniać ochronę danych użytkowników i być zgodny z obowiązującymi przepisami o ochronie danych osobowych.
- Wsparcie dla wielu języków: Interfejs aplikacji powinien być dostępny początkowo w języku polskim i angielskim. Z możliwością późniejszego łatwego rozszerzenia dla większej liczby języków.
- Dokumentacja: Aplikacja powinna posiadać pełną dokumentację techniczną oraz instrukcje instalacji i wdrożenia, umożliwiające łatwe wdrożenie i utrzymanie systemu.

3.3 Przypadki użycia

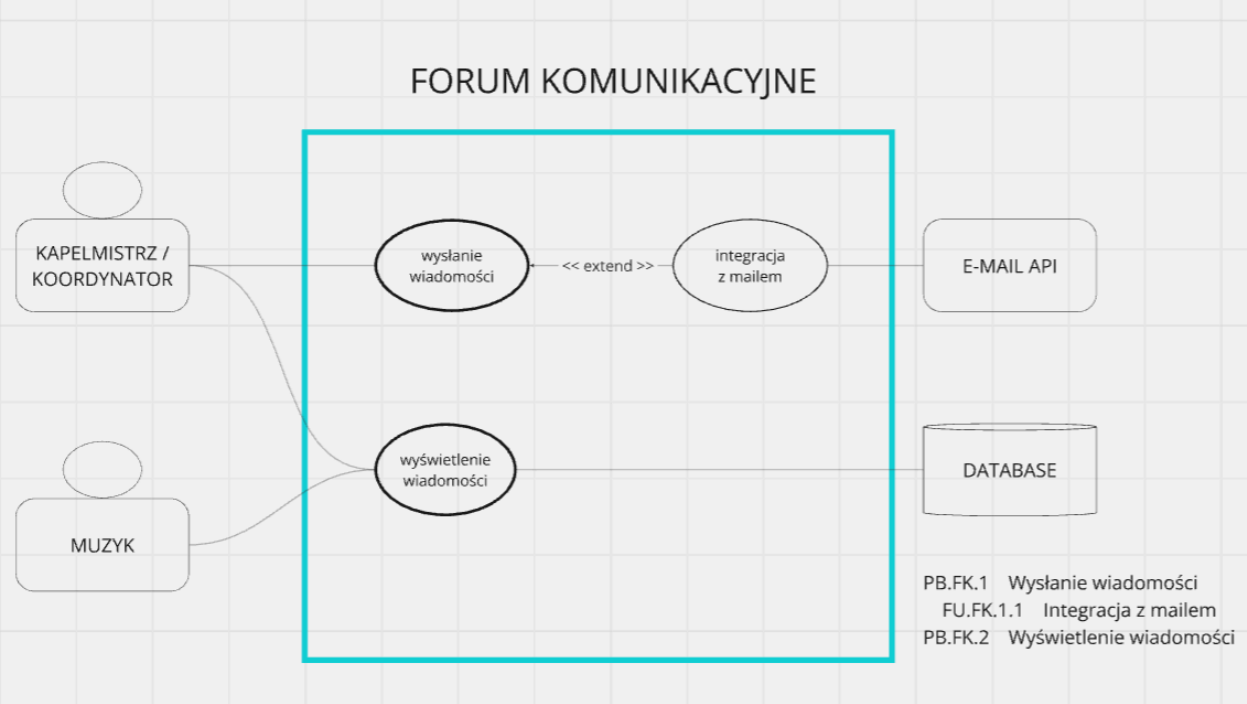
Najważniejsze przypadki użycia zamodelowane zostały na diagramach. Dotyczą one organizacji wydarzeń (zdj. 1.), zarządzania utworami i dystrybucji nut (zdj. 2.), forum komunikacyjnego (zdj. 3.) oraz zarządzania grupami (zdj. 4.).



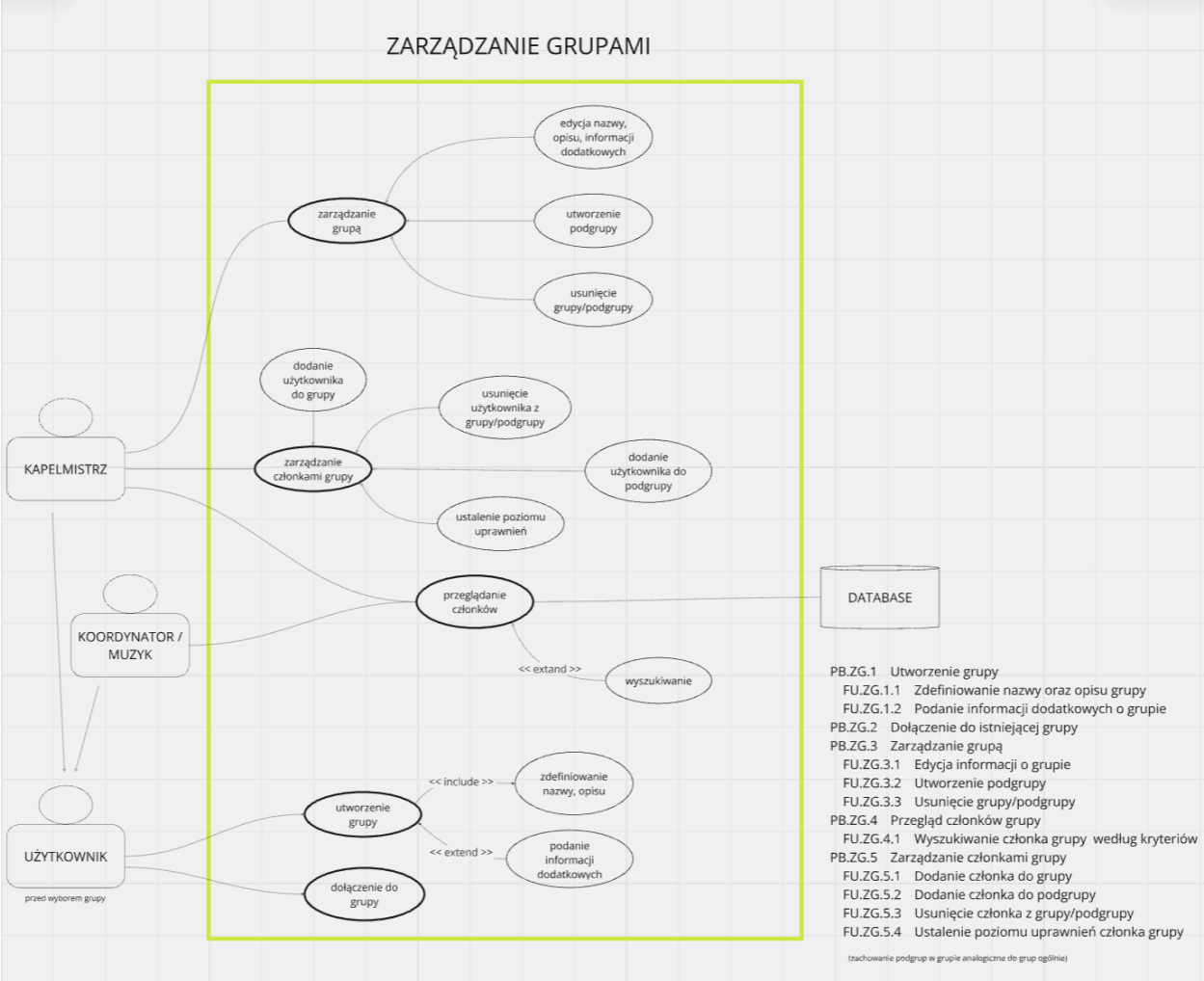
Zdj. 1. Diagram przypadków użycia - organizacja wydarzeń



Zdj. 2. Diagram przypadków użycia - zarządzanie utworami i dystrybucja nut



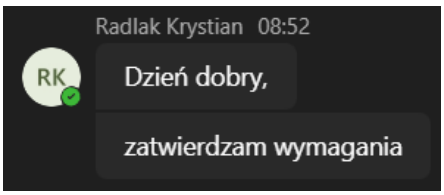
Zdj. 3. Diagram przypadków użycia - forum komunikacyjne

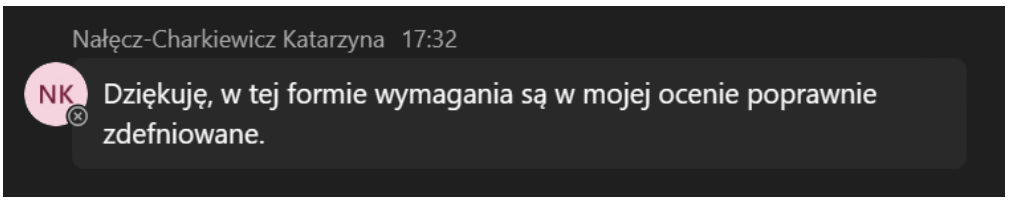


Zdj. 4. Diagram przypadków użycia - zarządzanie grupami

3.4 Potwierdzenie zgodności wymagań

Zatwierdzenie wymagań przez mentora oraz właściciela tematu



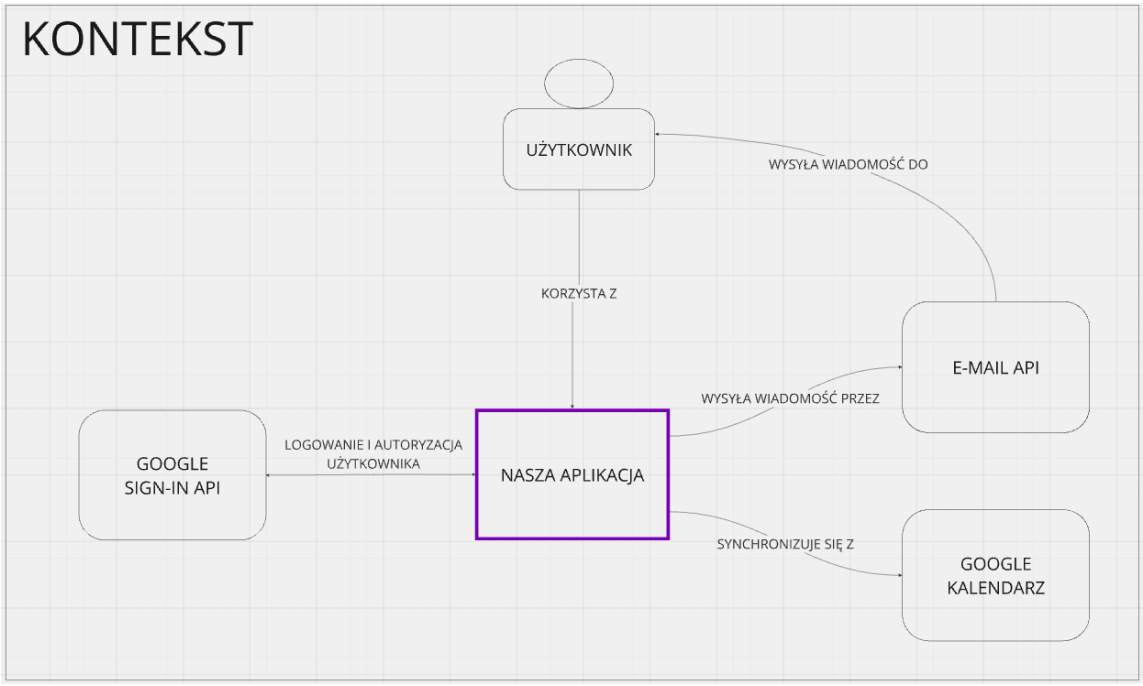


4 Definicja architektury

- Plan struktury systemu: c4

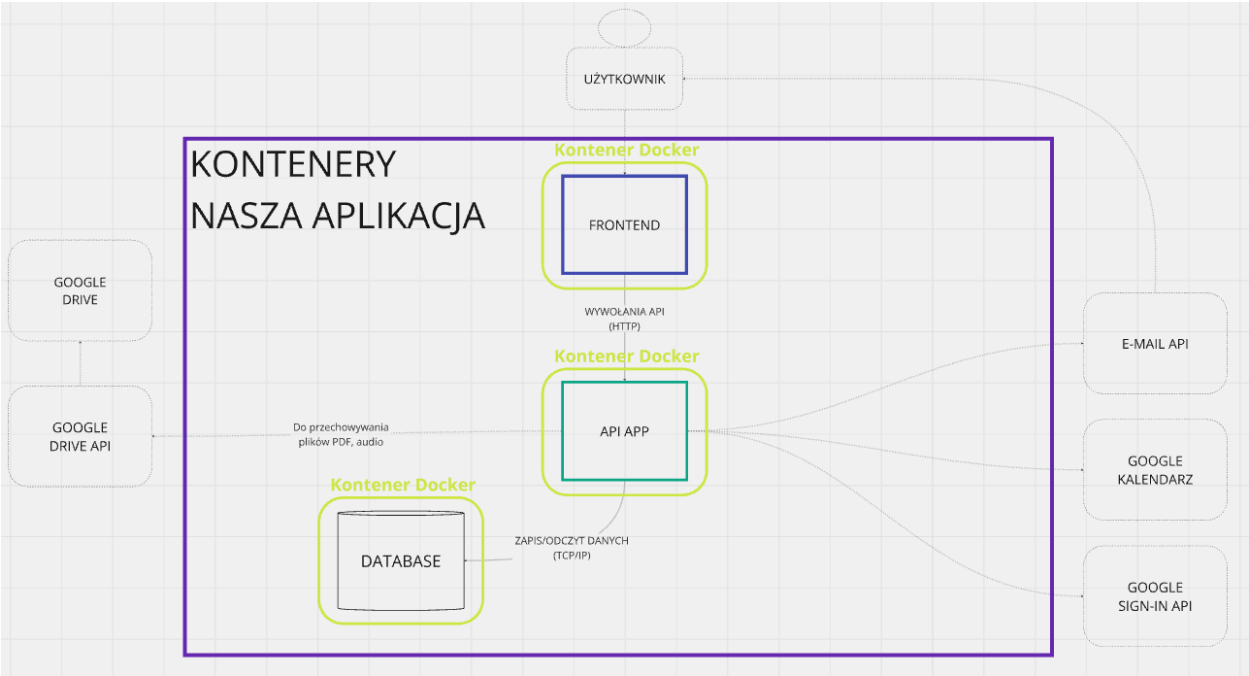
Link do Miro z całym modelem C4.

- Warstwa C1 - Kontekst (zdj 5.) - zewnętrzny kontekst opisu naszej aplikacji.
    - Użytkownik łączy się z naszą aplikacją z wykorzystaniem swojego konta google, które jest kluczowe w dalszym działaniu aplikacji
    - Użytkownik przy procesie logowania zostaje zautoryzowany przez naszą aplikację korzystając z Google Sign-In Api
    - Do przesyłania maili na konto Google (pocztę Gmail) użytkownika aplikacja korzysta z E-Mail Api
    - Aplikacja synchronizuje się z kalendarzem Google korzystając z Google Calendar Api



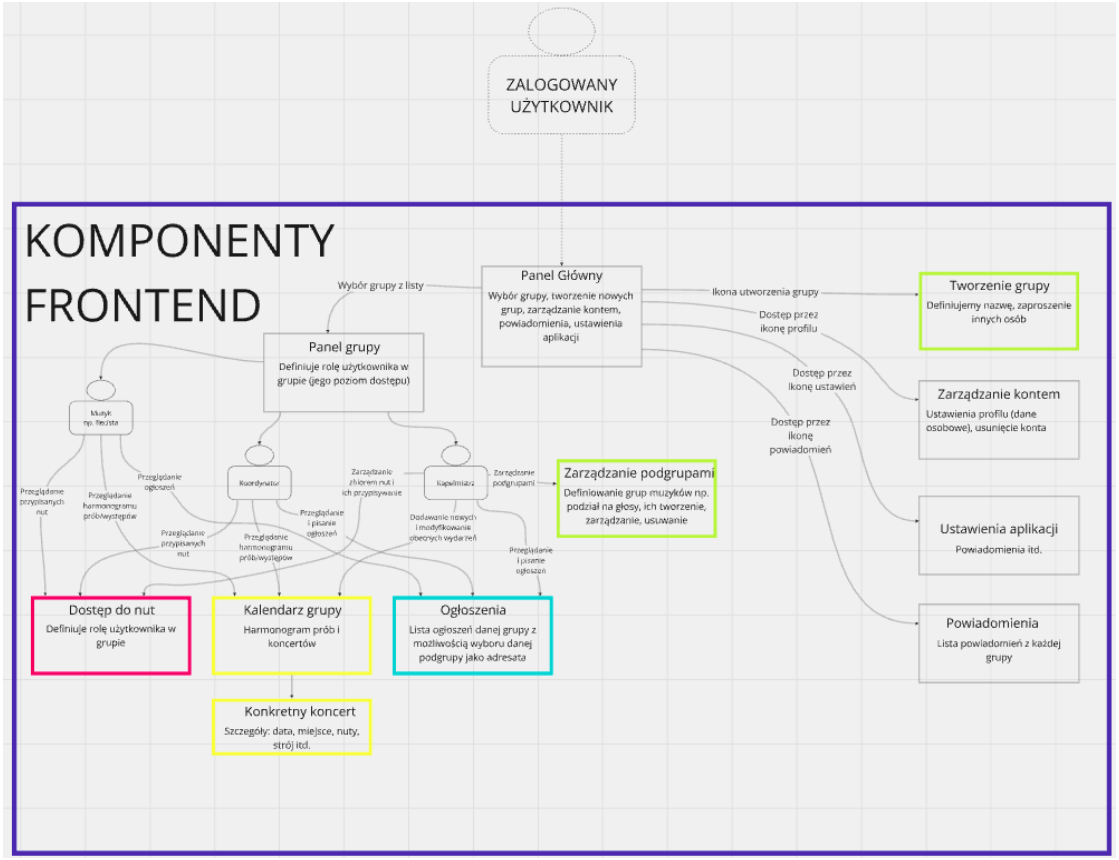
Zdj. 5. pierwsza warstwa modelu C4

- Warstwa C2 (Zdj. 6) - Kontenery - struktura wewnętrzna naszej aplikacji. Wykorzystamy **szablon architektoniczny architektury trójwarstwowej**. Każdy z trzech kluczowych komponentów aplikacji: Frontend, API (serwer), Baza danych znajduje się w oddzielnym kontenerze Docker. Te porozumiewają się między sobą z wykorzystaniem protokołu HTTP. Co warto zaznaczyć nasza aplikacja ma dwie bazy danych z czego pierwsza jest bazą relacyjną - przechowuje dane systemu, ale bezpośrednio nie przechowuje plików, a jedynie informacje o nich. Druga baza to chmura Google Drive, na której przetrzymywane są wszystkie pliki. Połączenie z nią jest realizowane za pośrednictwem Google Drive Api. Zatem nasza baza przechowuje klucz ID do plików trzymanych w chmurze Google Drive.



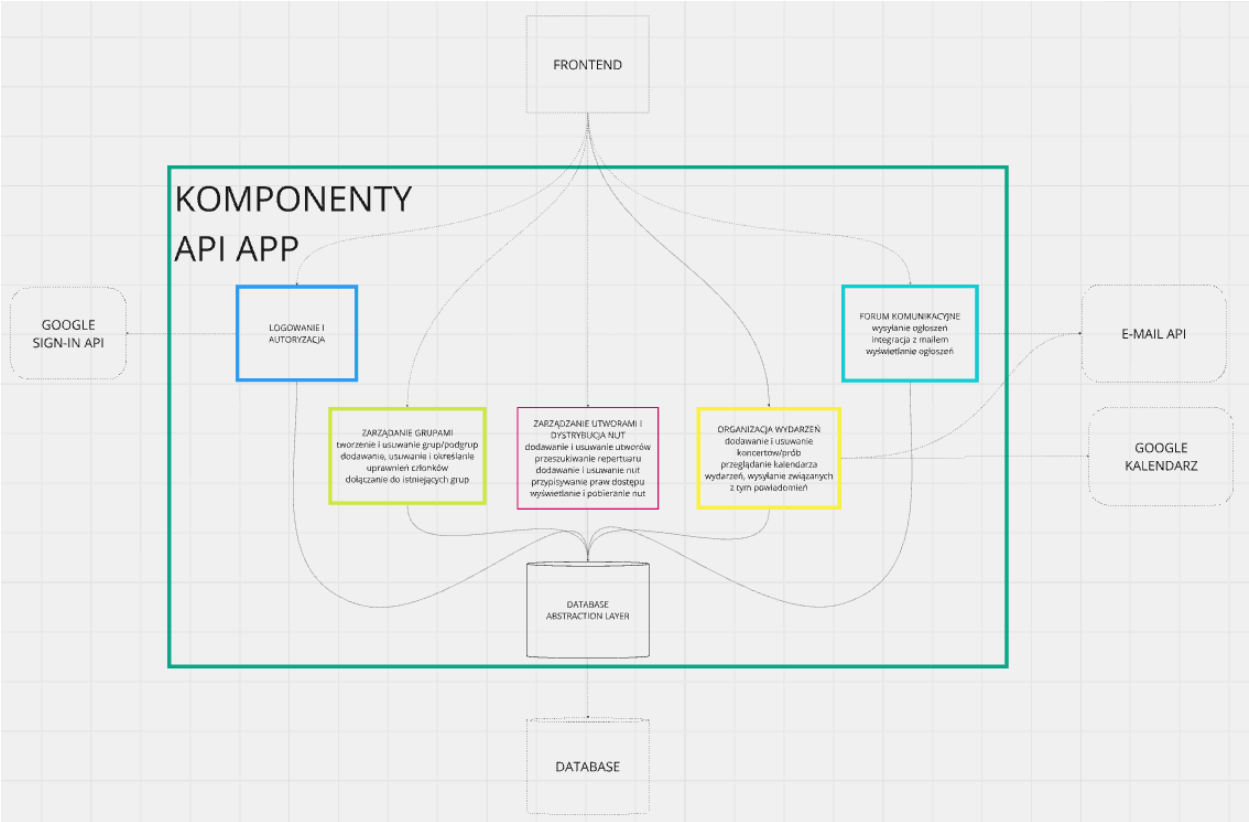
Zdj. 6. druga warstwa modelu C4

- Warstwa C3 - Komponenty:
    - Frontend (Zdj 7.) - służy do bezpośredniej interakcji z użytkownikiem. Wykonywane polecenia przez użytkownika wysyłane są do drugiego komponentu - API App, z którym się bezpośrednio komunikuje. Bazowym oknem jest tutaj Panel Główny, który pozwala na szybką orientację w systemie i zawiera odnośniki do poszczególnych funkcji:
        - Panel Grupy - sprawdza rolę użytkownika w grupie (Muzyk, Koordynator, Kapelmistrz) po czym udostępnia okna dotyczące konkretnej grupy:
            - Dostęp do nut - lista przypisanych nut oraz możliwość zarządzania przydziałem nut
            - Kalendarz grupy - wyświetla harmonogram prób i koncertów
            - Ogłoszenia - lista ogłoszeń grupy
            - Zarządzanie podgrupami - pozwala wyświetlać obecne grupy muzyków, nimi zarządzać oraz tworzyć nowe
        - Tworzenie nowej grupy - pozwala na utworzenie nowej grupy, zdefiniowanie jej nazwy oraz zaproszenie osób do grupy
        - Zarządzanie kontem - służy do wyświetlania i modyfikacji swoich danych w systemie oraz do usunięcia konta
        - Ustawienia aplikacji - pozwala na zmianę ustawień np. dotyczących wysyłania powiadomień
        - Powiadomienia - lista wszystkich powiadomień ze wszystkich grup oraz tych technicznych z aplikacji



Zdj. 7. komponent Frontend z trzeciej warstwy modelu C4

- API APP (backend) (Zdj 8.) - odpowiada za techniczną realizację funkcjonalności systemu oraz wykonuje odpowiednie akcje w odpowiedzi na polecenia użytkownika. Łączy się bezpośrednio z frontendem protokołem HTTP oraz wykonuje odpowiednie polecenia do baz danych w zależności od wybranej akcji. Polecenia do baz danych są wykonywane przy użyciu warstwy abstrakcji (Data Abstraction Layer). Główne funkcjonalności to:
  - Logowanie i autoryzacja - weryfikuje użytkownika korzystając z Google Sign-In Api
  - Zarządzanie grupami - obsługuje mechanizm zarządzania grupami oraz ich tworzenia i usuwania, również odpowiada za zarządzanie poziomami dostępu poszczególnych użytkowników w grupie
  - Zarządzanie utworami i dystrybucja nut - realizuje dodawanie i usuwanie nut z systemu, przypisywanie praw dostępu do nut, wyświetlanie i pobieranie nut
  - Organizacja wydarzeń - dodawanie, usuwanie i modyfikacja wydarzeń, przeglądanie kalendarza, aktualizuje o wprowadzone modyfikacje kalendarz google użytkownika oraz wysyła mu stosowne powiadomienie mailowo i w aplikacji
  - Forum komunikacyjne - realizuje wysyłanie ogłoszeń w aplikacji oraz na maila, wyświetla ogłoszenia

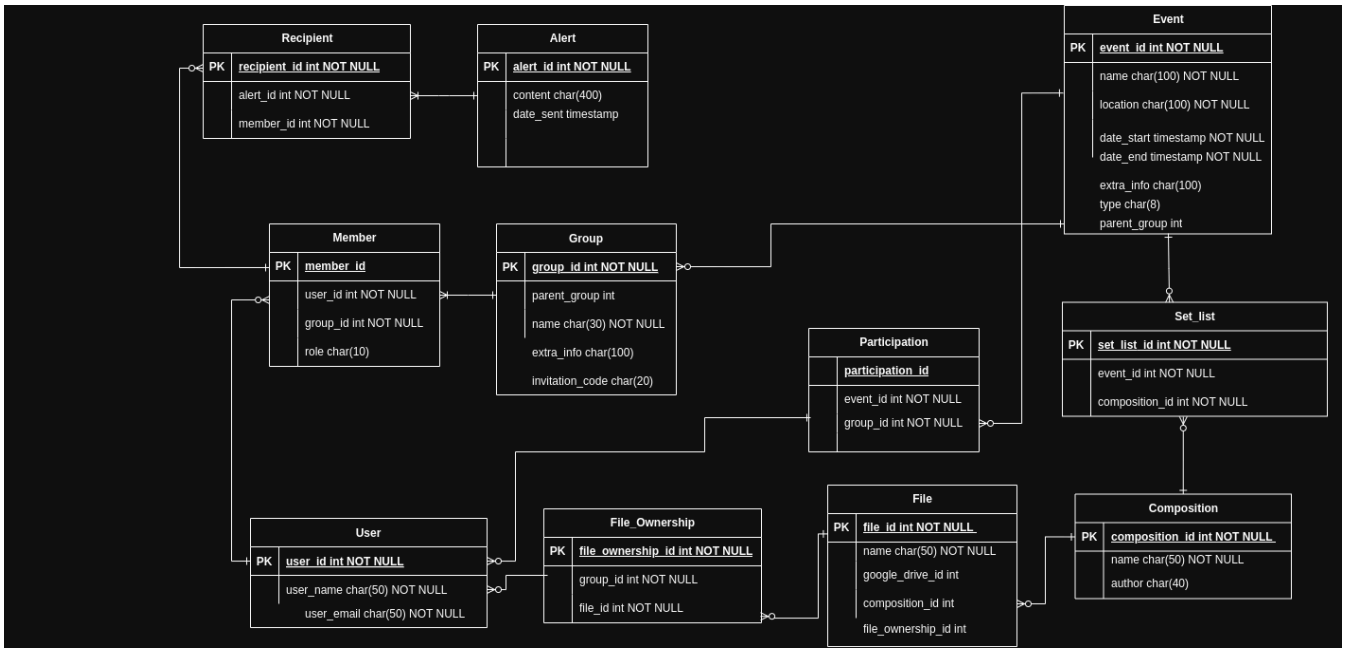


Zdj. 8 komponent API APP (serwer) z trzeciej warstwy modelu C4

- Baza danych - składa się z dwóch komponentów:
  - Baza relacyjna - przechowuje wszystkie informacje o systemie (grupy, użytkownicy, powiadomienia, informacje o plikach, wydarzenia) w odpowiednich tabelach, również przechowuje klucze ID plików z chmury Google Drive
  - Chmura Google Drive - przechowuje wszystkie pliki, do których można się odwołać przy użyciu ID konkretnego pliku, dostęp przy użyciu Google Drive Api

5 Dane trwałe

5.1 Model logiczny danych



Zdj. 9. Logiczny model relacyjnej bazy danych

5.2 Przetwarzanie i przechowywanie danych

- Relacyjna baza danych: PostgreSQL

- Dostęp do bazy danych postgresql: SQLAlchemy

Baza danych obsługiwana jest z poziomu aplikacji - nie wykorzystujemy funkcji/procedur z samej bazy.

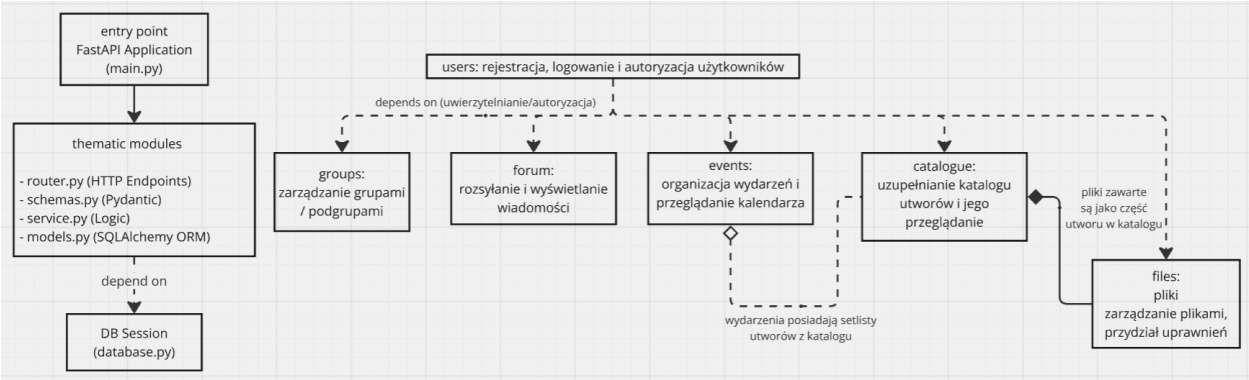
- Dodatkowa baza danych przeznaczona do obsługi plików: Postanowiliśmy użyć chmury Google Drive do przechowywania plików. Polegać to będzie na tym, że w bazie relacyjnej wspomnianej wcześniej będzie tabela File. W tej tabeli będzie pole google_drive_id, które generowane jest przy każdym dodaniu pliku do Google Drive, przy użyciu Google Drive API. Dzięki temu będziemy mogli wysyłać zapytanie z naszego serwera do naszej bazy danych, skąd otrzymamy google_drive_id, które zostanie użyte przez nasz serwer do wysłania zapytania do Google Drive API i na przykład pobrania pliku o danym id.

6 Specyfikacja analityczna i projektowa

- https://gitlab-stud.elka.pw.edu.pl/kgolcz/pzsp2-megalodony

- Metody realizacji:

- AngularJS/TypeScript - Frontend
- Python FastAPI - Backend
- VScode, środowiska wirtualne (.venv)
- Środowisko wdrażania - Docker



Zdj. 10. Model struktury aplikacji

- Statystyki:

- przybliżona liczba i rozmiar plików źródłowych:

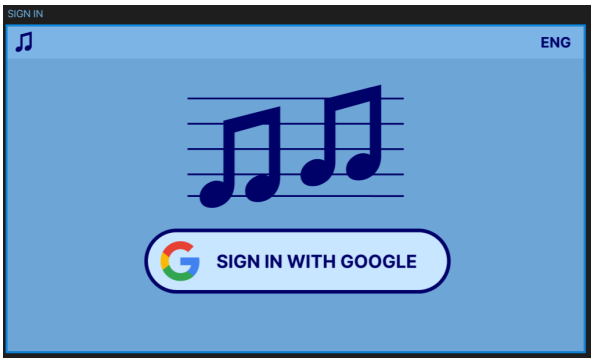
  backend: 40 (100KB) frontend: 100 (600KB)

- Linie kodu: ok. 6000
- Aktualna liczba testów jednostkowych: 30
- Aktualne pokrycie kodu przez testy: 79%

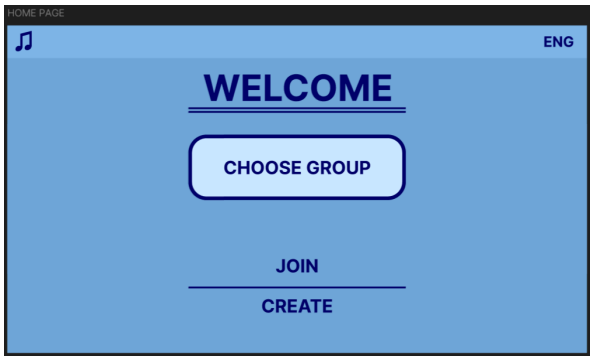
7 Projekt standardu interfejsu użytkownika

Interfejs użytkownika wstępnie zaprojektowany został z wykorzystaniem aplikacji figma.

Pierwsze, co użytkownik widzi, to logo aplikacji oraz możliwość zalogowania przez Google (zdj. 11.). Po udanym zalogowaniu ma możliwość wyboru grupy, do której już należy, bądź też dołączenia do lub utworzenia nowej (zdj. 12.). Przyciski te otwierają na stronie nakładki, oferujące wymienione funkcjonalności (zdj. 13.).



Zdj. 11. Ekran logowania

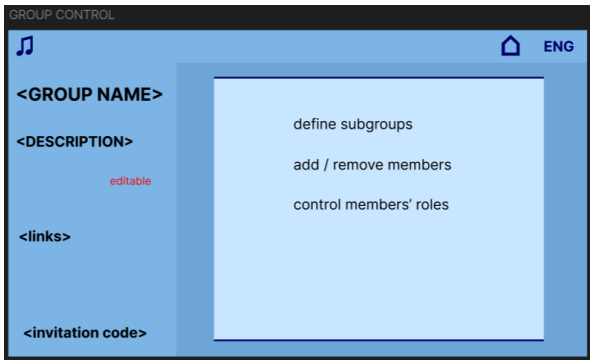


Zdj. 12. Ekran powitalny z wyborem grupy

Zdj. 13. Lista grup do wyboru, okienko dołączania, formularz utworzenia grupy

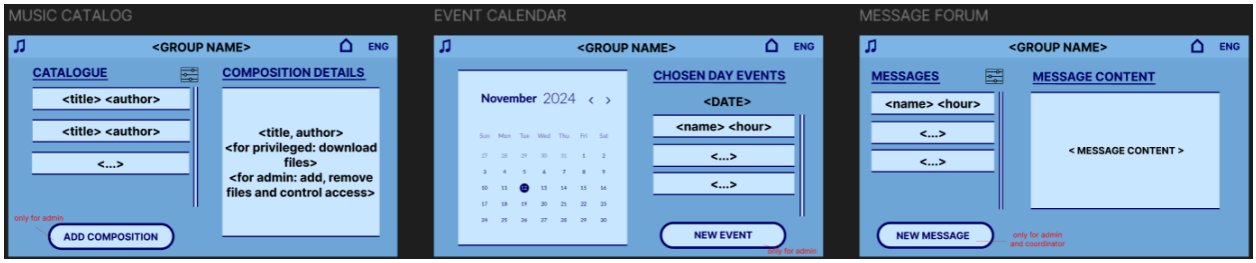
Po wybraniu grupy, użytkownik ma dostęp do kolejnego ekranu powitalnego (zdj. 14.), gdzie może zobaczyć dostępne mu informacje o grupie oraz wybrać, co chce zobaczyć dalej. Strona kontroli grupy (zdj. 15.) dostępna jest jedynie dla uprawnionych członków. Administratorzy mogą zobaczyć na niej szczegółowe informacje - jak kod zaproszeniowy, listę członków oraz podgrup - oraz nimi zarządzać.

Zdj. 14. Ekran powitalny grupy

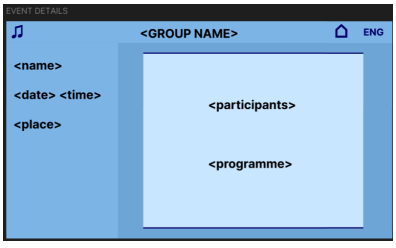


Zdj. 15. Strona kontroli grupy

Katalog muzyczny, kalendarz wydarzeń oraz forum komunikacyjne (zdj. 16.) dostępne są dla wszystkich, jednak uprawnieni członkowie mają na tych stronach większe możliwości - jak dodawanie utworów czy wydarzeń oraz pisanie ogłoszeń.



Zdj. 16. Katalog muzyczny, kalendarz wydarzeń, forum komunikacyjne



Z kalendarza wydarzeń przejść można także do strony ze szczegółami wybranego wydarzenia (zdj. 17.).

Zdj. 17. Ekran wydarzenia

8 Specyfikacja testów

Testy jednostkowe:

backend: pytest

frontend: przygotowane środowisko testowe jasmine + karma

Testy akceptacyjne:

Wszystkie testy akceptacyjne opierają się na pliku populate.sql, który dodaje przykładowe dane do bazy danych. Aby zainicjować bazę danych oraz wykonać na niej plik populate.sql należy uruchomić skrypt db_init.sh. Scenariusze testów akceptacyjnych:

1. Logowanie się do istniejącego konta oraz wylogowywanie:
- Uruchom aplikację w przeglądarce (localhost:4200)
- Naciśnij ikonę Google znajdującą się na “przycisku zaloguj się przez Google”
    - Wyświetli się nowe okno logowania za pomocą konta Google
- Zaloguj się za pomocą konta o e-mailu: megalodony.pzsp2@gmail.com
    - Wyświetli się ekran główny aplikacji
- naciśnij przycisk x w prawym górnym rogu
    - Wyświetli się znowu ekran logowania
2. Logowanie się jako nowy użytkownik
- Uruchom aplikację w przeglądarce (localhost:4200)
- Naciśnij ikonę Google znajdującą się na “przycisku zaloguj się przez Google”
    - Wyświetli się nowe okno logowania za pomocą konta Google
- Zaloguj się za pomocą prywatnego konta:
    - Wyświetli okno z polem na wpisanie swojej nazwy użytkownika
- Wpisz nową nazwę użytkownika i zatwierdź przyciskiem
    - Wyświetli się ekran główny aplikacji
- naciśnij przycisk x w prawym górnym rogu
    - Wyświetli się znowu ekran logowania
- Ponownie naciśnij ikonę Google znajdującą się na “przycisku zaloguj się przez Google”
- Zaloguj się przy pomocy tego samego konta
    - Wyświetli się strona główna aplikacji bez potrzeby ustawiania nazwy użytkownika
3. Wyświetlanie grup użytkownika
- Po zalogowaniu się za pomocą konta o e-mailu: megalodony.pzsp2@gmail.com
- Naciśnij przycisk “Wybierz grupę”
    - Wyświetli się panel z listą grup (Grupa “Grajkowie” oraz podgrupa: “Grajkowie:Strunowe”)
4. Tworzenie grupy
- Po zalogowaniu się za pomocą konta o e-mailu: megalodony.pzsp2@gmail.com
- Naciśnij przycisk “Utwórz”
    - Wyświetli się panel proszący o podanie nazwy oraz opisu grupy
- Wpisz “Nowa grupa” w polu “nazwa” oraz “przykładowy opis” w polu “Opis” oraz zatwierdź przyciskiem

    - Pod polem “Opis” wyświetli się potwierdzenie “Group Nowa grupa created successfully.”
- Naciśnij poza polem
- Naciśnij przycisk “Wybierz grupę”
    - Wyświetli się panel z listą grup (Grupa “Grajkowie” oraz podgrupa: “Grajkowie:Strunowe”, Utworzona grupa “Nowa grupa”)

5. Dołączanie do grupy

- Po zalogowaniu się za pomocą konta o e-mailu: megalodony.pzsp2@gmail.com
- Naciśnij przycisk dołącz
    - Wyświetli się panel z tytułem “Podaj kod grupy”
- Wpisz w polu “123”
    - Pod polem wyświetli się informacja zwrotna: “Group joined successfully.”
- Naciśnij poza panelem
- Naciśnij przycisk “Wybierz grupę”
    - Wyświetli się panel z listą grup (Grupa “Grajkowie” oraz podgrupa: “Grajkowie:Strunowe”, Grupa do której dołączyliśmy “Kumple”)

6. Wyświetlanie ekranu ustawień grupy

- Po zalogowaniu się za pomocą konta o e-mailu: megalodony.pzsp2@gmail.com
- Naciśnij przycisk “Wybierz grupę”
    - Wyświetli się panel z listą grup (Grupa “Grajkowie” oraz podgrupa: “Grajkowie:Strunowe”)
- Naciśnij na grupę “Grajkowie”
    - Wyświetli się ekran grupy
    - Po lewej stronie widać informacje o grupie, kolejno: nazwa grupy (Grajkowie), dodatkowe informacje o grupie (Polub nas na instagramie LINK), rola użytkownika (Kapelmistrz), kod zaproszenia do grupy (ASD12) oraz przycisk do zatwierdzenia zmian w grupie
    - W centralnej części ekranu wyświetlą się przyciski: “Katalog utworów”, “Kalendarz wydarzeń”, “Forum wiadomości”, “Ustawienia grupy”
- Naciśnij przycisk “Ustawienia grupy”
    - Wyświetli się ekran zarządzania grupą
    - W górnej części ekranu po lewej stronie wyświetli się lista użytkowników: adresy email, ich aktualne role grupie (można je zmieniać przez rozwinięcie pola roli) wraz z przyciskiem “-” umożliwiającym usuwanie użytkownika z grupy. Lista użytkowników: (megalodony.pzsp2@gmail.com z rolą “Kapelmistrz”, f@gmail.com z rolą “Muzyk”, k@gmail.com z rolą “Koordynator”, w@gmail.com z rolą “Muzyk”)
    - Po lewej stronie górnej części panelu jest moduł do zapraszania użytkowników z polem email oraz do filtrowania wyświetlania użytkowników według email lub roli
    - W dolnej części panelu po lewej stronie jest lista podgrup (Grajkowie:Strunowe) wraz z przyciskiem “-” umożliwiającym usuwanie podgrupy
    - W centralnej części panelu znajdują się członkowie podgrupy, którzy wyświetlają się w przypadku naciśnięcia podgrupy po lewej stronie (po naciśnięciu podgrupy Grajkowie:Strunowe wyświetlają się użytkownicy: megalodony.pzsp2@gmail.com, w@gmail.com)

    - Po prawej stronie znajduje się moduł tworzenia nowej podgrupy z polami “nazwa” i “opis” oraz przycisk zatwierdzania podgrupy

7. Edytowanie danych grupy nadrzędnej:

- Zaloguj się za pomocą konta o e-mailu: megalodony.pzsp2@gmail.com oraz wybierz grupę “Grajkowie” listy grup użytkownika oraz naciśnij przycisk “Ustawienia grupy”
- Naciśnij nazwę grupy (Grajkowie)
    - Pojawi się kursor w tekście “Grajkowie”
- Zmień nazwę na “Grajkowie i spółka”
- Naciśnij informacje dodatkowe grupy (Polub nas na instagramie LINK)
    - Pojawi się kursor w tekście “Polub nas na instagramie LINK”
- Zmień informacje na “Polub nas na instagramie NOWYLINK”
- Naciśnij przycisk zatwierdzający zmiany na dole ekranu
    - Wyświetli się informacja zwrotna: “Group info saved.”
    - Zmienione zostaną: nazwa grupy oraz informacje dodatkowe na temat grupy
- Naciśnij przycisk strzałki w prawym górnym rogu
    - Wyświetli się ekran grupy z zmienioną nazwą grupy oraz informacją na temat grupy

8. Edytowanie użytkowników grupy oraz filtrowanie użytkowników grupy

- Zaloguj się za pomocą konta o e-mailu: megalodony.pzsp2@gmail.com oraz wybierz grupę “Grajkowie” listy grup użytkownika oraz naciśnij przycisk “Ustawienia grupy”
- W górnej części ekranu po lewej stronie naciśnij na rolę użytkownika przy polu f@gmail.com
    - Wyświetlą się role na jakie możemy zmienić
- Wybierz rolę “Koordynator”
    - Zmieni się rola przy f@gmail.com na “Koordynator”
- Naciśnij przycisk “-” w rzędzie odpowiadającym w@gmail.com
    - Użytkownik w@gmail.com zniknie z listy
- W górnej części po prawej stronie przy polu “e-mail” wpisz literę “f”
    - W liście w lewej górnej części wyświetli się tylko użytkownik f@gmail.com
- Usuń wpisaną literę “f”, zmień pole e-mail na “ROLA” oraz wpisz w polu “Koordynator”
    - W liście w lewej górnej części wyświetli się zarówno użytkownik f@gmail.com jak i k@gmail.com

9. Zarządzanie podgrupami

- Zaloguj się za pomocą konta o e-mailu: megalodony.pzsp2@gmail.com oraz wybierz grupę “Grajkowie” listy grup użytkownika oraz naciśnij przycisk “Ustawienia grupy”
- W lewej dolnej stronie ekranu zarządzania naciśnij na podgrupę “Grajkowie:Strunowe”
    - Wyświetli się lista użytkowników należących do tej podgrupy (megalodony.pzsp2@gmail.com, w@gmail.com)
- Naciśnij przycisk “-” obok użytkownika w@gmail.com
    - Użytkownik w@gmail.com zniknie z listy
- W prawej dolnej stronie ekranu zarządzania uzupełnij pole nazwa tekstem “podgrupa” oraz opis tekstem “nowa podgrupa” i naciśnij przycisk “utwórz podgrupę”
    - Po lewej stronie w liście podgrup pojawi się nowa o nazwie “podgrupa”
- Naciśnij na nowoutworzoną podgrupę “podgrupa”
    - W centralnej, dolnej części ekranu wyświetli się lista członków nowej podgrupy (użytkownik megalodony.pzsp2@gmail.com, który utworzył podgrupę)

- Mając wciąż zaznaczoną podgrupę “podgrupa” naciśnij na przycisk przy roli użytkownika w górnej części panelu odpowiadającą użytkownikowi k@gmail.com
    - Lista użytkowników należących do podgrupy zmieni się na: megalodony.pzsp2@gmail.com, k@gmail.com

Testy integracyjne:

Aby włączyć testy integracyjne należy wykonać kolejne komendy:

1. Dodaj aktualny token (TEST_TOKEN) do pliku .env Token otrzymasz logując się z kontem google do aplikacji. Przed zalogowaniem naciśnij przycisk F12 i po poprawnym zalogowaniu należy wejść w sekcję pakietów network i wybrać pakiet typu POST od google-sign-in z kodem 200. W zakładce response będzie wygenerowany token w polu app_token.
2. Jeżeli środowisko było wcześniej uruchamiane, należy usunąć wcześniej zbudowany obraz:

docker image rm pzsp2-megalodony-backend:latest

3. Uruchom środowisko testowe z aplikacją przy pomocy:

docker-compose -f docker-compose.test.yml up

4. Zainicjuj bazę danych:

cd database

./db_init.sh

5. Uruchom testy znajdujące się wewnątrz kontenera

docker exec -it pzsp2-megalodony-backend-1 pytest tests/ --cov=src

Testy te uruchomią się wraz z wyliczeniem pokrycia.

Aktualne pokrycie testów: 79%

Przykładowe przypadki specjalne: Usunięcie kapelmistrza gdy nie ma innego w grupie, zmienienie roli kapelmistrza, gdy nie ma innego w grupie, dodanie do wydarzenia użytkownika osobiście, jak i przez grupę.

9 Wirtualizacja/konteneryzacja

Każda warstwa naszej aplikacji jest w osobnym kontenerze Dockerowym. Orkiestracja warstw odbywa się z użyciem docker-compose. Warstwa Frontendu oraz Backendu mają osobne Dockerfile do odpowiedniego zdefiniowania i uruchomienia obrazu. Dla warstwy danych skorzystaliśmy z gotowego obrazu postgres:13. Kontenery są połączone po sieci typu bridge.

10 Bezpieczeństwo

Wrażliwe dane aplikacji są przetrzymywane w osobnych plikach .env - zawiera informacje o połączeniu z bazą danych oraz z Google Sign-in API oraz w pliku service_account.json - zawiera dane do nawiązania połączenia z dyskiem google. Oba pliki są poufne i są przechowywane jedynie lokalnie przez administratora systemu.

Aplikacja korzysta z Google Sign-in API do logowania użytkownika. Po zalogowaniu generowany jest token google, który zawiera informacje o zalogowanym użytkowniku, a te są przekształcane w token JWT używany dalej przez aplikacje. Wszystkie funkcje wymagające autoryzacji użytkownika wykorzystują ten token do autoryzacji po stronie serwera. Bezpieczeństwo zapewnia korzystanie z zabezpieczonego API logowania Google, a pozyskane dane są szyfrowane do tokenu JWT. Taki system logowania sprawia, że w bazie danych są przechowywane jedynie maile użytkowników bez haseł.

Pliki są przechowywane w chmurze google drive do której dostęp jest możliwy tylko z administracyjnego konta google założonego na potrzebę aplikacji. Tylko to jedno konto ma bezpośredni dostęp do wszystkich plików. Użytkownicy aplikacji mają dostęp tylko pośredni do plików - przez aplikację, która łączy się z Dyskiem Google i pobiera jedynie te pliki do których użytkownik ma przyznany dostęp. Dostępem do plików dla użytkowników zarządza jedynie administrator grupy (rola Kapelmistrz). Użytkownik będący Kapelmistrzem w jednej grupie, nie widzi i nie może zarządzać dostępem do plików w pozostałych grupach w których go nie ma lub nie ma w nich roli Kapelmistrza.

11 Podręcznik użytkownika

Uruchomienie aplikacji

Uruchom aplikację w przeglądarce pod adresem htttp://localhost:4200

Wybór języka

W prawym górny rogu należy nacisnąć na napis EN lub PL i wybrać z listy wybrany język.

Logowanie do aplikacji

Naciśnij ikonę Google znajdującą się na “przycisku zaloguj się przez Google”. Jeżeli nie widzisz ikony Google na przycisku odśwież kartę przeglądarki. Następnie wybierz swoje konto Google z listy.

Wylogowywanie z aplikacji

Po zalogowaniu w prawym górnym rogu pojawi się znak x. Naciśnij go aby się wylogować. Pojawi się dodatkowe okienko, na którym trzeba potwierdzić chęć wylogowania się z aplikacji.

Otworzenie panelu grupy, której jest się już członkiem

Po zalogowaniu należy nacisnąć przycisk “wybierz grupę”, po czym nacisnąć wybraną grupę z listy.

Dołączenie do nowej grupy

Po zalogowaniu należy nacisnąć przycisk “dołącz”. Następnie wpisać kod dołączenia do grupy i nacisnąć przycisk “dołącz”.

Utworzenie nowej grupy

Po zalogowaniu należy nacisnąć przycisk “utwórz”. Następnie wpisać nazwę oraz opis tworzonej grupy. Po uzupełnieniu tych pól nacisnąć przycisk “utwórz”.

12 Podręcznik administratora

Instrukcja budowy systemu z kodu źródłowego

Po pobraniu kodu źródłowego z repozytorium
https://gitlab-stud.elka.pw.edu.pl/kgolcz/pzsp2-megalodony
należy dodać plik .env ze zmiennymi środowiskowymi do głównego folderu oraz plik service_account.json (do połączenia z dyskiem Google) do folderu backend.

Następnie należy wejść do głównego folderu i wykonać polecenie:

docker-compose up

To uruchomi wszystkie moduły aplikacji. Teraz trzeba wejść do folderu database i wykonać skrypt db_init.sh znajdujący się w katalogu /database, który zainicjuje bazę danych. Od tego momentu aplikacja jest dostępna pod adresem htttp://localhost:4200

Instrukcja zarządzania zasobami systemu

Każdy z modułów aplikacji można uruchamiać z użyciem docker-compose up <nazwa_modułu>. Domyślnie uruchamiane są wszystkie moduły (cała aplikacja). Natomiast chcąc zakończyć aplikacji działanie należy wykonać docker-compose down. Po uruchomieniu moduł loguje swoje działanie w terminalu. Poleceniem docker-compose logs <nazwa_modułu> można wyświetlić logi danego modułu. Chcąc zobaczyć pliki wewnątrz uruchomionego modułu trzeba wykonać polecenie docker exec -it <nazwa_modułu> sh. To uruchomi powłokę wewnątrz modułu.

13 Podsumowanie

Krytyczna analiza osiągniętych wyników:

- Mocne strony
  - Dobrze zdefiniowaliśmy potrzeby klienta i zaplanowaliśmy ich realizację poprzez odpowiednie wymagania użytkowe i systemowe w naszym projekcie.
  - Dobraliśmy i rozrysowaliśmy architekturę systemu odpowiednią pod naszą aplikację
  - Stworzyliśmy intuicyjny interfejs do korzystania z aplikacji.
  - Połączyliśmy aplikację z Google Sign-In API do autoryzacji i logowania użytkowników, co zapewnia bezpieczeństwo logowania oraz zwalnia z potrzeby przechowywania haseł użytkowników.
  - Połączyliśmy aplikację z Dyskiem Google do przechowywania plików, dzięki czemu udało się połączyć bazę relacyjną z chmurą i mieć możliwość wykonywania zapytań SQL oraz przechowywać do 15 GB plików dowolnego typu. Zapewnia to również bezpieczeństwo w dostępie do plików, ponieważ bezpośredni dostęp do chmury jest tylko z konta administratora systemu.
  - Wsparcie różnych języki, obecnie polski i angielski, możliwość dodania nowych.
- Słabe strony
  - Z powodu ograniczonego czasu, nie udało się zrealizować wszystkich przewidzianych funkcjonalności - brak integracji z kalendarzem google, ograniczone możliwości filtrowania katalogu utworów.
  - Ograniczone pokrycie kodu przez testy jednostkowe.

Możliwe kierunki rozwoju

- Rozwinięcie aplikacji o nie-zaimplementowane funkcjonalności.
- Dodanie wsparcia kolejnych języków.
- Przystosowanie interfejsu pod urządzenia mobilne.
- Integracja z mapami Google.

Zdj. 18. Wizytówka projektu

14 Bibliografia

Dokumentacja wykorzystywanych technologii, w szczególności:

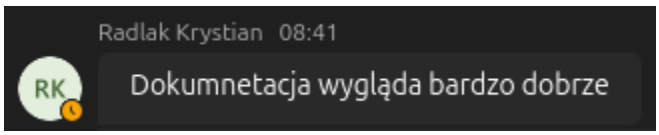
- https://docs.python.org/3/
- https://fastapi.tiangolo.com/
- https://www.typescriptlang.org/docs
- https://v17.angular.io/docs
- https://docs.docker.com/
- https://docs.pytest.org/en/stable/

Metodologia wytwarzania aplikacji:

- https://12factor.net/

Inspiracja istniejącymi rozwiązaniami:

- https://glissandoo.com/en



Zatwierdzam dokumentację.	…… …… Data i podpis Mentora
