POLITECHNIKA KOSZALIŃSKA

WYDZIAŁ ELEKTRONIKI I INFORMATYKI

Kierunek: Informatyka

Specjalność: Technologie Internetowe i Mobilne

Jakub Achtelik

Numer albumu: U-20019

E-PSPK Aplikacja Samorządowa dla Politechniki Koszalińskiej

E-PSPK Local Government Application for Koszalin University of Technology

Praca inżynierska wykonana pod kierunkiem
dra inż. Rafała Wojszczyka

Koszalin 2026

Streszczenie

Celem pracy było zaprojektowanie oraz implementacja aplikacji do obsługi wydarzeń dla organizacji studenckich zrzeszonych w Forum Uczelni Technicznych (FUT). Funkcjonalność aplikacji pozwala na podstawowe operacje CRUD na wydarzeniach, zarządzanie harmonogramem, punktami na mapie, opisem, rejestrowanie oraz logowanie. Aplikacja zawiera moduł archiwizacji wydarzeń, składu osobowego prezydium FUT wraz z filtrowaniem. Zaimplementowano możliwość tworzenia posiedzeń, automatyczne tworzenie prezentacji multimedialnej na podstawie planu spotkania. Zaimplementowano także interaktywną mapę samorządów technicznych wraz z listą samorządów studenckich polskich uczelni technicznych, która zawiera podstawowe informacje takie jak nazwa uczelni, skład zarządu, miasto oraz logotyp.

Abstract

The objective of this project was to design and implement an event management application for student organizations associated with the Forum of Technical Universities (FUT). The application's functionality includes basic CRUD operations for events, schedule management, map points, and event descriptions, as well as registration and login features.

The application features an event archiving module and a directory of the FUT Presidium members with filtering capabilities. It also includes tools for creating meetings, automatically generating multimedia presentations based on meeting agendas. Additionally, an interactive map of technical student governments was implemented, alongside a list of student governments from Polish technical universities. This list provides essential information such as the university name, board members, city, and logo.

Słowa kluczowe:
aplikacja mobilna,
aplikacja internetowa,
samorząd studencki,
cyfryzacja procesów,
zarządzanie wydarzeniami,
elektroniczne posiedzenia

Keywords:
mobile application,
web application,
student government,
process digitization,
event management,
electronic meetings

1. Wprowadzenie 4
1.1. Samorządy Studenckie w Polsce 4
1.1.1. Parlament Studentów Rzeczypospolitej Polskiej – PSRP 5
1.1.2. Forum Uczelni Technicznych – FUT 6
1.1.3. Parlament Studentów Politechniki Koszalińskiej – PSPK 6
1.2. Cel pracy 8
2. Wymagania pracy 9
2.1. Podstawowe funkcjonalności 9
2.2. Narzędzia i technologie 10
2.2.1. Visual Studio Code 10
2.2.2. JetBrains Phpstorm 11
2.2.3 Git & GitHub 12
2.2.4 PHP 13
2.2.5 Laravel 14
2.2.6 Bootstrap 15
2.2.7 OpenStreetMap API 16
2.2.8 Docker 17
2.2.9 PWA – Progressive Web App 18
2.2.10 phpMyAdmin 19
3. Analiza istniejących rozwiązań 21
3.1. System eSesja 21
3.2. Platforma Meeting Application 22
3.3 Porównanie rozwiązań w kontekście celów pracy 23
4.Użytkowanie 24
4.1.Główny widok aplikacji 24
4.2.Widok samorządu 25
4.3 Widok ekranu wydarzeń 27
4.4. Widok interaktywnej mapy. 28
4.5. Widok ekranu logowania. 29
4.6. Moduł panelu administracyjnego i zarządzania systemem. 31
4.7. Moduł Katalogu Samorządów Studenckich 33
5. Projekt i implementacja 35
5.1. Architektura aplikacji MVC 35
5.2. Routing aplikacji 36
5.3. Database Seeders – mechanizm wypełniania bazy danych 37
5.4. Technologia PWA 37
5.5 Baza danych 39
5.6. Przypadki użycia 41
5.6.1.Implementacja modułu uwierzytelniania 44
5.6.2. Modelowanie danych – Struktura posiedzeń (Meeting) 45
5.6.3. Automatyzacja prezentacji 47
6. Podsumowanie 49
7. Dalsze kierunki rozwoju 50
7.1. System elektronicznego głosowania i kworum 50
7.2. Rozszerzenie automatyzacji i integracja z zewnętrznymi API 50
7.3. Zaawansowana analiza danych i powiadomienia Push 50
7.4. Internacjonalizacja i moduł wielojęzyczności 50
Bibliografia 51
Spis skrótów 52
Spis rysunków 53

1. Wprowadzenie

1.1. Samorządy Studenckie w Polsce

Samorząd studencki stanowi podstawową formę reprezentacji studentów na uczelniach wyższych w Polsce. Jest to organizacja działająca w ramach struktury uczelni, której głównym zadaniem jest reprezentowanie interesów studentów wobec władz uczelni, instytucji publicznych oraz innych podmiotów związanych z funkcjonowaniem szkolnictwa wyższego. Funkcjonowanie samorządów studenckich reguluje ustawa z dnia 20 lipca 2018 r. – Prawo o szkolnictwie wyższym i nauce, która określa ich kompetencje, sposób działania oraz zakres odpowiedzialności.

Samorząd studencki tworzą wszyscy studenci danej uczelni, natomiast jego organami są przedstawiciele wybrani w drodze wyborów. Struktura organizacyjna samorządu może różnić się w zależności od uczelni, jednak zazwyczaj obejmuje takie jednostki jak parlament studentów, zarząd samorządu, komisję rewizyjną oraz przedstawicieli w organach kolegialnych uczelni. Dzięki temu studenci mają możliwość aktywnego uczestniczenia w procesie podejmowania decyzji dotyczących funkcjonowania uczelni, w tym m.in. w sprawach programów studiów, organizacji życia akademickiego czy warunków studiowania.

Do podstawowych zadań samorządu studenckiego należy również organizacja wydarzeń integracyjnych, konferencji, szkoleń oraz inicjatyw naukowych i kulturalnych skierowanych do społeczności akademickiej. Samorządy studenckie często współpracują także z organizacjami studenckimi, kołami naukowymi oraz instytucjami zewnętrznymi, co sprzyja rozwojowi aktywności społecznej i organizacyjnej studentów.

Na poziomie ogólnopolskim samorządy studenckie są zrzeszone w organizacjach reprezentujących środowisko akademickie w skali kraju. Najważniejszą z nich jest Parlament Studentów Rzeczypospolitej Polskiej, który pełni funkcję ogólnokrajowej reprezentacji studentów. W ramach jego struktur działają również organizacje branżowe skupiające uczelnie o podobnym profilu, takie jak Forum Uczelni Technicznych. Na poziomie poszczególnych uczelni funkcjonują natomiast lokalne organy samorządowe, których przykładem jest Parlament Studentów Politechniki Koszalińskiej.

Struktura ta tworzy wielopoziomowy system reprezentacji studentów, obejmujący zarówno poziom uczelniany, jak i ogólnokrajowy. Dzięki temu możliwe jest skuteczne reprezentowanie interesów studentów oraz rozwijanie współpracy pomiędzy środowiskami akademickimi różnych uczelni w Polsce.

1.1.1. Parlament Studentów Rzeczypospolitej Polskiej – PSRP

Parlament Studentów Rzeczypospolitej Polskiej (PSRP) jest ogólnopolską organizacją reprezentującą wszystkich studentów uczelni wyższych w Polsce. W jego skład wchodzą przedstawiciele samorządów studenckich zarówno uczelni publicznych, jak i niepublicznych. PSRP pełni funkcję oficjalnej reprezentacji środowiska studenckiego wobec organów administracji publicznej, w szczególności Ministerstwa Nauki i Szkolnictwa Wyższego, a także innych instytucji państwowych odpowiedzialnych za kształtowanie polityki edukacyjnej. Działalność organizacji regulowana jest przez ustawę Prawo o szkolnictwie wyższym i nauce, która określa zakres kompetencji samorządu studenckiego oraz jego rolę w systemie szkolnictwa wyższego w Polsce.

Do głównych zadań Parlamentu Studentów RP należy reprezentowanie interesów studentów na szczeblu krajowym, opiniowanie projektów aktów prawnych dotyczących szkolnictwa wyższego, a także wspieranie działalności samorządów studenckich poprzez organizację szkoleń, konferencji oraz inicjatyw integrujących środowisko akademickie. Organizacja prowadzi również działania mające na celu rozwój kompetencji liderów studenckich oraz wzmacnianie współpracy pomiędzy samorządami uczelni w całym kraju. PSRP stanowi ważną platformę wymiany doświadczeń oraz dobrych praktyk w zakresie zarządzania organizacjami studenckimi.

Rys. 1.1. Logo Parlamentu Studentów Rzeczypospolitej Polskiej

źródło: [https://psuek.pl/o-nas/psrp-parlament-studentow-rzeczpospolitej-polskiej]

1.1.2. Forum Uczelni Technicznych – FUT

Forum Uczelni Technicznych (FUT) jest jedną z komisji branżowych działających w strukturze Parlamentu Studentów Rzeczypospolitej Polskiej. Organizacja ta zrzesza samorządy studenckie uczelni technicznych w Polsce, umożliwiając współpracę pomiędzy środowiskami akademickimi o podobnym profilu kształcenia. Głównym celem FUT jest reprezentowanie interesów studentów uczelni technicznych oraz podejmowanie inicjatyw związanych z rozwojem edukacji technicznej i inżynierskiej w Polsce.

W ramach swojej działalności Forum organizuje liczne wydarzenia, takie jak zjazdy, konferencje, szkolenia czy Sesje Robocze przedstawicieli samorządów studenckich. Wydarzenia te umożliwiają wymianę doświadczeń pomiędzy uczelniami, a także wspólne wypracowywanie rozwiązań dotyczących problemów środowiska akademickiego. Politechnika Koszalińska od czasu podpisania porozumienia w 2006 roku aktywnie uczestniczy w strukturach Forum, biorąc udział w jego inicjatywach oraz wydarzeniach integrujących społeczność studentów uczelni technicznych.

Rys. 1.2. Logo Forum Uczelni Technicznych

źródło: [https://fut.edu.pl/]

1.1.3. Parlament Studentów Politechniki Koszalińskiej – PSPK

Parlament Studentów Politechniki Koszalińskiej (PSPK) jest organem samorządu studenckiego reprezentującym wszystkich studentów Politechniki Koszalińskiej. Jego działalność polega na reprezentowaniu interesów studentów wobec władz uczelni, a także na aktywnym uczestnictwie w procesie podejmowania decyzji dotyczących funkcjonowania uczelni. Parlament działa na podstawie ustawy Prawo o szkolnictwie wyższym i nauce oraz wewnętrznych regulaminów obowiązujących na uczelni, które określają jego strukturę organizacyjną, kompetencje oraz zakres działania.

Do głównych zadań Parlamentu Studentów Politechniki Koszalińskiej należy reprezentowanie społeczności studenckiej w organach kolegialnych uczelni, takich jak senat czy rady wydziałów. Przedstawiciele samorządu studenckiego uczestniczą w procesach opiniowania programów studiów, zmian w regulaminach oraz innych decyzji mających wpływ na jakość kształcenia i warunki studiowania. Dzięki temu studenci mają realny wpływ na funkcjonowanie uczelni oraz możliwość zgłaszania swoich potrzeb i postulatów.

Istotnym elementem działalności PSPK jest również organizowanie wydarzeń skierowanych do studentów, takich jak szkolenia, konferencje, wydarzenia integracyjne oraz inicjatywy kulturalne i naukowe. Samorząd studencki współpracuje także z organizacjami studenckimi, kołami naukowymi oraz instytucjami zewnętrznymi, wspierając rozwój aktywności akademickiej i społecznej studentów.

Parlament Studentów Politechniki Koszalińskiej bierze również udział w inicjatywach ogólnopolskich poprzez współpracę z Parlamentem Studentów Rzeczypospolitej Polskiej oraz organizacjami branżowymi, takimi jak Forum Uczelni Technicznych. Umożliwia to wymianę doświadczeń pomiędzy samorządami studenckimi różnych uczelni oraz udział w wydarzeniach o zasięgu ogólnopolskim. Dzięki temu studenci Politechniki Koszalińskiej mogą aktywnie uczestniczyć w działaniach środowiska akademickiego nie tylko na poziomie uczelni, ale także w skali całego kraju.

Rys. 1.3. Logo Parlamentu Studentów Politechniki Koszalińskiej

źródło: [https://pspk.tu.koszalin.pl/informator/]

1.2. Cel pracy

Głównym celem pracy jest zaprojektowanie oraz implementacja wielomodułowej aplikacji internetowej w technologii PWA (Progressive Web App), służącej do kompleksowej obsługi wydarzeń i cyfryzacji procesów organizacyjnych dla samorządów studenckich zrzeszonych w Forum Uczelni Technicznych.

Zakres pracy

Zakres zrealizowanych w ramach pracy zadań obejmuje:

Projekt architektury i bazy danych: Zaprojektowanie struktury systemu w oparciu o wzorzec architektoniczny MVC (Model-View-Controller) z wykorzystaniem frameworka Laravel oraz zamodelowanie bazy danych.

Implementację modułu zarządzania wydarzeniami: Stworzenie mechanizmów pozwalających na operacje CRUD (tworzenie, odczyt, aktualizacja, usuwanie) dla wydarzeń, zarządzanie harmonogramem, przypisywanie lokalizacji geograficznych oraz archiwizację minionych spotkań.

Implementację modułu obsługi posiedzeń i automatyzacji: Opracowanie narzędzia do tworzenia porządku obrad wraz z załącznikami oraz innowacyjnego mechanizmu automatycznego generowania prezentacji multimedialnych na podstawie planu spotkania.

Stworzenie interaktywnego Katalogu Samorządów Studenckich: Implementacja bazy wiedzy o samorządach uczelni technicznych w Polsce (zawierającej m.in. składy zarządów i logotypy) oraz zintegrowanie jej z interaktywną mapą przy użyciu OpenStreetMap API.

Zarządzanie użytkownikami: Realizacja bezpiecznego modułu uwierzytelniania i autoryzacji (logowanie, rejestracja, zarządzanie uprawnieniami).

Wdrożenie i konteneryzację (Środowisko deweloperskie): Wykorzystanie narzędzia Docker do konteneryzacji infrastruktury w celu zapewnienia przenośności i spójności środowiska uruchomieniowego.

Opracowanie responsywnego interfejsu (Frontend): Zbudowanie warstwy wizualnej aplikacji przy użyciu biblioteki Bootstrap, dostosowanej do urządzeń mobilnych i stacjonarnych.

2. Wymagania pracy

2.1. Podstawowe funkcjonalności

Aplikacja została zaprojektowana z myślą o studentach oraz członkach samorządów studenckich uczestniczących w wydarzeniach organizowanych przez nadrzędną jednostkę organizacyjną np. Forum Uczelni Technicznych. Głównym założeniem systemu jest usprawnienie zarządzania wydarzeniami, komunikacji pomiędzy uczestnikami oraz organizacji posiedzeń i spotkań. System ma na celu cyfryzację wybranych procesów organizacyjnych, które dotychczas często realizowane były w sposób manualny lub przy wykorzystaniu wielu niezależnych narzędzi. W dalszej części podrozdziału zostały omówione moduły aplikacji.

Zarządzanie wydarzeniami - Umożliwia on administratorom tworzenie nowych wydarzeń, ich edycję oraz usuwanie w przypadku zmian organizacyjnych. W ramach tego modułu możliwe jest określenie szczegółowych informacji dotyczących wydarzenia, takich jak nazwa, opis, harmonogram, miejsce organizacji czy dane kontaktowe organizatorów. Dodatkowo system pozwala na przypisywanie współrzędnych geograficznych do wydarzeń, co umożliwia ich wizualizację na interaktywnej mapie. Funkcjonalność ta ułatwia uczestnikom szybkie odnalezienie lokalizacji wydarzenia oraz planowanie udziału w spotkaniach.

Katalog samorządów studenckich - moduł ten pozwala na przeglądanie oraz wyszukiwanie informacji o poszczególnych samorządach studenckich działających na uczelniach technicznych w Polsce. Użytkownicy mogą filtrować dane według różnych kryteriów, takich jak nazwa uczelni, miasto czy słowa kluczowe. Dzięki temu możliwe jest szybkie odnalezienie informacji o danym samorządzie, jego strukturze organizacyjnej oraz podstawowych danych kontaktowych.

Obsługa posiedzeń - system umożliwia tworzenie porządku obrad poprzez dodawanie kolejnych punktów spotkania, które mogą zawierać opisy oraz dodatkowe materiały. Do poszczególnych punktów można również dołączać załączniki w formie odnośników do zewnętrznych zasobów internetowych, takich jak dokumenty czy prezentacje. Rozwiązanie to pozwala uczestnikom na szybki dostęp do wszystkich niezbędnych materiałów związanych z danym spotkaniem.

Automatyzacja prezentacji - mechanizm automatycznego generowania prezentacji multimedialnej na podstawie wcześniej przygotowanego planu posiedzenia. Funkcjonalność ta umożliwia automatyczne tworzenie slajdów zawierających kolejne punkty porządku obrad, co znacząco ułatwia prowadzenie spotkań oraz ogranicza czas potrzebny na przygotowanie materiałów prezentacyjnych.

Autoryzacja i uwierzytelnianie - ten moduł umożliwia bezpieczną rejestrację nowych użytkowników oraz logowanie do systemu. Mechanizm ten zapewnia ochronę danych oraz kontrolę dostępu do poszczególnych funkcji aplikacji. Dzięki zastosowaniu odpowiednich metod uwierzytelniania możliwe jest ograniczenie dostępu do funkcji administracyjnych wyłącznie dla uprawnionych użytkowników, takich jak organizatorzy wydarzeń lub administratorzy systemu.

Zastosowanie powyższych funkcjonalności pozwala na stworzenie spójnego systemu wspierającego organizację wydarzeń oraz zarządzanie informacjami w środowisku samorządów studenckich uczelni technicznych. W dalszej części podrozdziału zostały omówione moduły aplikacji.

2.2. Narzędzia i technologie

2.2.1. Visual Studio Code

Visual Studio Code jest nowoczesnym, otwartoźródłowym edytorem kodu źródłowego opracowanym przez firmę Microsoft. Narzędzie to dostępne jest na najpopularniejsze systemy operacyjne, takie jak Windows, macOS oraz Linux, co pozwala na jego wykorzystanie w różnych środowiskach programistycznych. Dzięki swojej uniwersalności oraz dużej liczbie dostępnych rozszerzeń Visual Studio Code jest jednym z najczęściej wykorzystywanych edytorów kodu wśród programistów zajmujących się tworzeniem aplikacji webowych, mobilnych oraz systemów backendowych.

Jedną z największych zalet Visual Studio Code jest jego modułowa budowa oraz możliwość instalowania licznych wtyczek rozszerzających funkcjonalność środowiska programistycznego. Dzięki temu edytor może zostać dostosowany do konkretnych potrzeb użytkownika oraz specyfiki realizowanego projektu. Wtyczki umożliwiają między innymi obsługę wielu języków programowania, integrację z systemami kontroli wersji, automatyczne formatowanie kodu, analizę składniową oraz podpowiedzi kodu w czasie rzeczywistym. Funkcje te znacząco usprawniają proces tworzenia oprogramowania oraz zmniejszają ryzyko występowania błędów w kodzie.

Visual Studio Code zapewnia również rozbudowane wsparcie dla systemów kontroli wersji, w szczególności dla systemu Git. Dzięki wbudowanym narzędziom możliwe jest zarządzanie repozytorium bezpośrednio z poziomu edytora, w tym przeglądanie historii zmian, tworzenie commitów czy rozwiązywanie konfliktów w kodzie. Integracja z platformami takimi jak GitHub pozwala na wygodne zarządzanie projektem oraz współpracę pomiędzy członkami zespołu programistycznego.

Rys. 2.1. Logo Visual Studio Code

źródło: [https://apps.microsoft.com/detail/xp9khm4bk9fz7q?hl=en-US&gl=US]

2.2.2. JetBrains Phpstorm

PhpStorm jest zaawansowanym środowiskiem programistycznym typu IDE (Integrated Development Environment) opracowanym przez firmę JetBrains, przeznaczonym przede wszystkim do tworzenia aplikacji w języku PHP. Narzędzie to zostało zaprojektowane z myślą o programistach tworzących nowoczesne aplikacje webowe i oferuje szeroki zestaw funkcji wspierających proces projektowania, pisania oraz testowania kodu źródłowego. Dzięki rozbudowanym mechanizmom analizy kodu oraz integracji z wieloma technologiami webowymi PhpStorm stanowi jedno z najczęściej wykorzystywanych środowisk programistycznych w ekosystemie PHP.

Jedną z najważniejszych funkcjonalności PhpStorm jest inteligentna analiza kodu źródłowego. Środowisko na bieżąco analizuje strukturę projektu, wykrywając potencjalne błędy składniowe, nieużywane zmienne czy niepoprawne wywołania metod. Dzięki temu programista może szybko zidentyfikować problemy w kodzie jeszcze na etapie jego tworzenia. Dodatkowo środowisko oferuje mechanizm automatycznego uzupełniania kodu (ang. code completion), który podpowiada nazwy klas, metod, zmiennych oraz parametrów funkcji na podstawie kontekstu programistycznego.

Istotnym elementem PhpStorm są także narzędzia wspierające refaktoryzację kodu. Refaktoryzacja polega na modyfikowaniu struktury kodu w taki sposób, aby poprawić jego czytelność, organizację oraz jakość, bez zmiany jego funkcjonalności. Środowisko umożliwia między innymi automatyczne zmienianie nazw klas i metod, przenoszenie fragmentów kodu czy reorganizację struktury plików projektu. Dzięki temu możliwe jest utrzymanie wysokiej jakości kodu w większych projektach programistycznych.

PhpStorm oferuje również szerokie wsparcie dla nowoczesnych frameworków i bibliotek wykorzystywanych w tworzeniu aplikacji internetowych. W szczególności zapewnia pełną integrację z frameworkiem Laravel, który został wykorzystany podczas implementacji aplikacji opisanej w niniejszej pracy. Środowisko umożliwia między innymi szybkie tworzenie kontrolerów, modeli oraz migracji bazy danych, a także wygodne zarządzanie routingiem aplikacji. Dzięki temu proces tworzenia aplikacji webowej jest bardziej przejrzysty i efektywny.

Rys. 2.2. Logo JetBrains

źródło: [https://www.nuget.org/profiles/jetbrains]

2.2.3 Git & GitHub

Git jest rozproszonym systemem kontroli wersji wykorzystywanym do zarządzania zmianami w kodzie źródłowym podczas tworzenia oprogramowania. System ten został zaprojektowany w celu umożliwienia programistom śledzenia historii zmian w projekcie, przywracania wcześniejszych wersji plików oraz pracy nad różnymi funkcjonalnościami w sposób uporządkowany i bezpieczny. Git jest obecnie jednym z najpopularniejszych narzędzi tego typu i znajduje zastosowanie zarówno w małych projektach indywidualnych, jak i w dużych przedsięwzięciach programistycznych realizowanych przez zespoły programistów.

Podstawową funkcją systemu Git jest zapisywanie kolejnych wersji projektu w postaci tzw. commitów. Każdy commit stanowi zapis zmian wprowadzonych w kodzie źródłowym wraz z opisem dotyczącym charakteru wprowadzonych modyfikacji. Dzięki temu możliwe jest prześledzenie całej historii projektu oraz identyfikacja momentów, w których wprowadzono konkretne funkcjonalności lub poprawki błędów. W przypadku wystąpienia problemów w działaniu aplikacji system umożliwia także powrót do wcześniejszej, stabilnej wersji kodu.

GitHub jest natomiast internetową platformą hostingową służącą do przechowywania repozytoriów Git oraz zarządzania projektami programistycznymi. Platforma ta umożliwia przechowywanie kodu źródłowego w chmurze, co pozwala na dostęp do projektu z dowolnego miejsca oraz tworzenie kopii zapasowych kodu. GitHub oferuje również wiele dodatkowych funkcjonalności wspierających pracę zespołową, takich jak system zgłaszania problemów (issues), przegląd zmian w kodzie (pull requests) czy narzędzia do zarządzania zadaniami projektowymi.

Rys. 2.3. Logo git i GitHub

źródło: [https://medium.com/cs-note/git-and-github-for-beginners-i-tutorial-263caa01f9c3]

2.2.4 PHP

PHP jest jednym z najpopularniejszych języków programowania wykorzystywanych do tworzenia dynamicznych aplikacji internetowych. Jest to język skryptowy wykonywany po stronie serwera, którego głównym zadaniem jest generowanie dynamicznej zawartości stron internetowych oraz obsługa logiki aplikacji webowych. PHP został zaprojektowany z myślą o prostocie użycia oraz łatwej integracji z technologiami sieciowymi, dzięki czemu znalazł szerokie zastosowanie w projektowaniu nowoczesnych systemów internetowych.

Jedną z kluczowych cech języka PHP jest jego ścisła integracja z protokołem HTTP oraz serwerami webowymi, takimi jak Apache czy Nginx. Kod PHP wykonywany jest na serwerze, a jego wynik – najczęściej w postaci kodu HTML – przesyłany jest do przeglądarki użytkownika. Dzięki temu możliwe jest tworzenie dynamicznych stron internetowych reagujących na działania użytkownika, takie jak wypełnianie formularzy, logowanie do systemu czy przeglądanie danych zapisanych w bazie danych.

PHP oferuje szeroki zestaw funkcji umożliwiających komunikację z bazami danych. Najczęściej wykorzystywanymi systemami bazodanowymi w aplikacjach PHP są MySQL oraz MariaDB. Dzięki temu możliwe jest przechowywanie oraz przetwarzanie danych w sposób uporządkowany, co ma szczególne znaczenie w przypadku aplikacji wymagających zarządzania dużą liczbą informacji. W kontekście niniejszej pracy język PHP został wykorzystany do obsługi logiki biznesowej aplikacji, w tym zarządzania wydarzeniami, użytkownikami oraz informacjami o samorządach studenckich.

Zastosowanie języka PHP w projekcie umożliwiło implementację funkcjonalności związanych z obsługą użytkowników, autoryzacją, zarządzaniem wydarzeniami oraz komunikacją z bazą danych. Dzięki swojej elastyczności oraz dużym możliwościom integracji z innymi technologiami webowymi PHP stanowi odpowiednie rozwiązanie do tworzenia aplikacji internetowych wspierających zarządzanie informacjami oraz procesami organizacyjnymi.

Rys. 2.4. Logo PHP

źródło: [https://en.wikipedia.org/wiki/PHP]

2.2.5 Laravel

Laravel jest nowoczesnym frameworkiem programistycznym przeznaczonym do tworzenia aplikacji internetowych w języku PHP. Framework ten został zaprojektowany z myślą o uproszczeniu procesu tworzenia aplikacji poprzez dostarczenie zestawu gotowych narzędzi oraz struktur organizujących kod źródłowy. Dzięki temu programista może skupić się na implementacji logiki biznesowej aplikacji, zamiast na tworzeniu podstawowych mechanizmów wymaganych w większości systemów webowych.

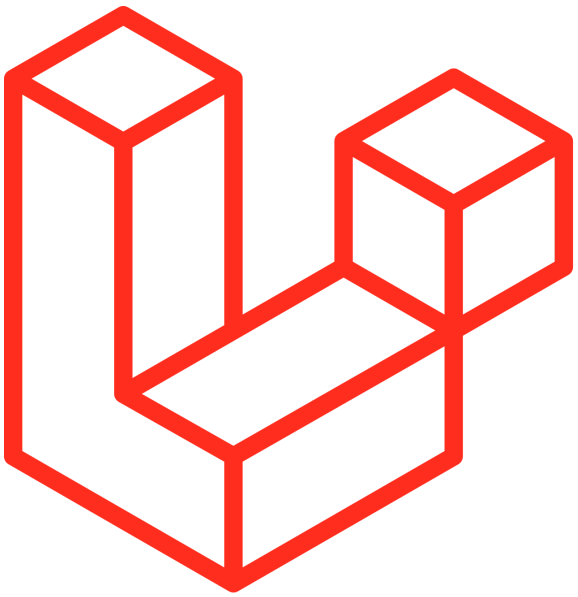
Jedną z najważniejszych cech frameworka Laravel jest wykorzystanie wzorca architektonicznego MVC. Wzorzec ten polega na podziale aplikacji na trzy główne warstwy: model odpowiadający za komunikację z bazą danych, widok odpowiedzialny za prezentację danych użytkownikowi oraz kontroler zarządzający logiką aplikacji i obsługą żądań. Takie podejście pozwala na lepszą organizację kodu, zwiększa jego czytelność oraz ułatwia rozwój i utrzymanie aplikacji w przyszłości.

Laravel oferuje również rozbudowany system routingu, który umożliwia definiowanie sposobu obsługi żądań HTTP kierowanych do aplikacji. Dzięki temu możliwe jest przypisywanie konkretnych adresów URL do odpowiednich metod kontrolerów. Mechanizm ten pozwala na tworzenie przejrzystej struktury aplikacji oraz łatwe zarządzanie jej funkcjonalnościami.

Istotnym elementem frameworka jest także system migracji baz danych, który umożliwia zarządzanie strukturą bazy danych w sposób programistyczny. Migracje pozwalają na tworzenie, modyfikowanie oraz usuwanie tabel w bazie danych przy użyciu kodu źródłowego. Dzięki temu możliwe jest zachowanie spójności struktury bazy danych pomiędzy różnymi środowiskami programistycznymi oraz łatwe wdrażanie zmian w projekcie.

Framework Laravel oferuje także wiele dodatkowych mechanizmów usprawniających tworzenie aplikacji internetowych. Należą do nich między innymi system uwierzytelniania użytkowników, obsługa sesji, mechanizmy ochrony przed atakami typu CSRF oraz narzędzia do walidacji danych wprowadzanych przez użytkownika. Dzięki tym funkcjom możliwe jest szybkie tworzenie bezpiecznych i skalowalnych aplikacji webowych.

W ramach realizacji niniejszej pracy framework Laravel został wykorzystany jako podstawowa technologia backendowa aplikacji. Pozwolił on na implementację logiki systemu, zarządzanie komunikacją z bazą danych oraz obsługę żądań użytkowników. Zastosowanie tego frameworka znacząco przyspieszyło proces tworzenia aplikacji oraz umożliwiło zachowanie przejrzystej i uporządkowanej struktury projektu.



Rys. 2.5. Logo Laravel

źródło: [https://en.wikipedia.org/wiki/Laravel]

2.2.6 Bootstrap

Bootstrap jest popularnym, otwartoźródłowym frameworkiem CSS wykorzystywanym do tworzenia nowoczesnych i responsywnych interfejsów użytkownika dla aplikacji internetowych. Framework ten został opracowany przez firmę Twitter i obecnie jest jednym z najczęściej stosowanych narzędzi wspierających projektowanie warstwy wizualnej stron internetowych. Bootstrap dostarcza zestaw gotowych komponentów, stylów oraz narzędzi, które pozwalają na szybkie tworzenie przejrzystych i estetycznych interfejsów użytkownika.

Jedną z najważniejszych cech Bootstrap jest jego system siatki (ang. grid system), który umożliwia projektowanie responsywnych układów stron internetowych. System ten oparty jest na podziale ekranu na kolumny, dzięki czemu możliwe jest łatwe dostosowanie wyglądu strony do różnych rozmiarów ekranów. Pozwala to na poprawne wyświetlanie aplikacji zarówno na komputerach stacjonarnych, jak i na urządzeniach mobilnych takich jak smartfony czy tablety. Responsywność interfejsu jest szczególnie istotna w przypadku aplikacji webowych, które mogą być wykorzystywane przez użytkowników korzystających z różnych typów urządzeń.

Bootstrap oferuje również szeroki zestaw gotowych komponentów interfejsu użytkownika. Należą do nich między innymi przyciski, formularze, menu nawigacyjne, tabele, okna dialogowe czy elementy nawigacji. Komponenty te można łatwo integrować z kodem HTML, co znacząco przyspiesza proces tworzenia interfejsu aplikacji. Dodatkowo framework zapewnia spójny styl graficzny, dzięki czemu wszystkie elementy interfejsu zachowują jednolity wygląd.

Framework Bootstrap zawiera także zestaw klas CSS umożliwiających szybkie stylizowanie elementów strony internetowej. Programista może w prosty sposób kontrolować marginesy, wyrównanie tekstu, kolory czy rozmiary elementów, bez konieczności pisania dużej ilości własnego kodu CSS. Rozwiązanie to pozwala na znaczące skrócenie czasu potrzebnego na projektowanie interfejsu użytkownika oraz ułatwia utrzymanie spójności wizualnej całej aplikacji.

W ramach realizacji niniejszej pracy Bootstrap został wykorzystany jako podstawowy framework do budowy warstwy frontendowej aplikacji. Dzięki jego zastosowaniu możliwe było stworzenie czytelnego i responsywnego interfejsu użytkownika, który poprawnie wyświetla się na różnych urządzeniach oraz w różnych przeglądarkach internetowych. Wykorzystanie gotowych komponentów oraz systemu siatki pozwoliło również na przyspieszenie procesu implementacji interfejsu aplikacji oraz zachowanie spójnego wyglądu wszystkich jej elementów.

Rys. 2.6. Logo Bootstrap

źródło: [https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)]

2.2.7 OpenStreetMap API

OpenStreetMap (OSM) to otwarty projekt dążący do stworzenia darmowej, edytowalnej mapy całego świata, a udostępniane przez niego API stanowi kluczowy interfejs programistyczny pozwalający na integrację danych geograficznych z aplikacjami zewnętrznymi. W przeciwieństwie do komercyjnych rozwiązań, takich jak Google Maps API, OpenStreetMap opiera się na modelu otwartych danych (Open Data), co pozwala programistom na swobodne wykorzystywanie, modyfikowanie i renderowanie map bez konieczności ponoszenia wysokich kosztów licencyjnych.

Kluczowym elementem OpenStreetMap API jest specyficzna struktura danych przestrzennych, oparta na trzech podstawowych typach obiektów: punktach (ang. *nodes*),

liniach (ang. *ways*) oraz relacjach (ang. *relations*). Dzięki takiemu podejściu API umożliwia nie tylko proste wyświetlanie mapy, ale również precyzyjne pobieranie informacji o konkretnych obiektach geograficznych, takich jak budynki, drogi, punkty użyteczności publicznej (POI) czy granice administracyjne. Elastyczność tego systemu sprawia, że jest on wykorzystywany w szerokim spektrum projektów – od prostych wizualizacji po zaawansowane systemy informacji geograficznej (GIS).

2.2.8 Docker

Docker jest wiodącą platformą typu Open Source, która służy do automatyzacji procesów wdrażania, skalowania i zarządzania aplikacjami za pomocą konteneryzacji. Narzędzie to pozwala na odizolowanie aplikacji od warstwy infrastruktury sprzętowej oraz systemu operacyjnego poprzez umieszczenie jej wraz ze wszystkimi niezbędnymi bibliotekami, plikami konfiguracyjnymi i zależnościami wewnątrz lekkiego, przenośnego kontenera. Dzięki temu programiści mogą mieć pewność, że aplikacja będzie działać w identyczny sposób na komputerze lokalnym, serwerze testowym oraz w środowisku produkcyjnym.

Fundamentem technologii Docker jest wykorzystanie obrazów (ang. *images*), które stanowią statyczne, warstwowe szablony zawierające instrukcje budowy środowiska. Na podstawie obrazu uruchamiany jest kontener, który w przeciwieństwie do tradycyjnych maszyn wirtualnych, nie wymaga emulacji pełnego systemu operacyjnego. Kontenery współdzielą jądro systemu operacyjnego hosta, co czyni je znacznie lżejszymi, szybszymi w uruchamianiu oraz mniej wymagającymi pod kątem zasobów sprzętowych takich jak pamięć RAM czy moc procesora.

Docker oferuje szereg narzędzi wspomagających cykl życia oprogramowania, wśród których kluczowe znaczenie mają:

- **Dockerfile:** plik tekstowy zawierający zestaw instrukcji niezbędnych do automatycznego zbudowania obrazu aplikacji.
- **Docker Hub:** publiczne repozytorium (rejestr), umożliwiające przechowywanie i współdzielenie gotowych obrazów bazowych (np. baz danych, serwerów www).
- **Docker Compose:** narzędzie pozwalające na definiowanie i uruchamianie wielokontenerowych aplikacji (np. połączenie frontendu, backendu i bazy danych w jedną spójną strukturę).

W ramach realizacji niniejszej pracy Docker został wykorzystany do konteneryzacji całej infrastruktury aplikacji. Dzięki przygotowaniu odpowiednich obrazów i konfiguracji Docker Compose, proces instalacji oraz uruchomienia systemu na dowolnym serwerze został sprowadzony do kilku prostych poleceń. Pozwoliło to na zachowanie pełnej spójności środowisk deweloperskich i produkcyjnych oraz zagwarantowało izolację poszczególnych komponentów aplikacji, co przełożyło się na większe bezpieczeństwo i stabilność całego rozwiązania.

Rys. 2.7. Logo docker

źródło: [https://www.influxdata.com/partners/docker/]

2.2.9 PWA – Progressive Web App

Progressive Web App to nowoczesne podejście do tworzenia aplikacji internetowych, które wykorzystuje zaawansowane możliwości przeglądarek, aby dostarczyć użytkownikom doświadczenie zbliżone do korzystania z natywnych aplikacji mobilnych lub desktopowych. Technologia ta, promowana głównie przez firmę Google, pozwala na budowanie rozwiązań, które są dostępne bezpośrednio przez adres URL, a jednocześnie oferują funkcjonalności dotychczas zarezerwowane dla oprogramowania instalowanego ze sklepów takich jak Google Play czy App Store. Kluczową ideą PWA jest progresywność, co oznacza, że aplikacja działa poprawnie u każdego użytkownika, niezależnie od przeglądarki, ale oferuje bogatsze funkcje tam, gdzie są one wspierane.

Fundamentem technicznym Progressive Web App są mechanizmy zapewniające wysoką wydajność i niezawodność, nawet w trudnych warunkach sieciowych. Do najważniejszych komponentów PWA należą:

- Service Workers: skrypty działające w tle, niezależnie od głównego wątku przeglądarki, które umożliwiają inteligentne buforowanie zasobów (caching), obsługę powiadomień push oraz pracę w trybie offline.
- Web App Manifest: plik konfiguracyjny w formacie JSON, który definiuje wygląd aplikacji po zainstalowaniu, w tym jej ikonę, kolory motywu oraz sposób wyświetlania (np. bez widocznego paska adresu przeglądarki).
- HTTPS: wymóg bezpieczeństwa, który gwarantuje, że dane przesyłane między użytkownikiem a serwerem są szyfrowane, co jest niezbędne do poprawnego działania Service Workerów.

Dzięki zastosowaniu PWA aplikacja zyskuje szereg cech, które znacząco poprawiają retencję użytkowników. Użytkownik ma możliwość dodania skrótu do aplikacji bezpośrednio na ekran główny swojego urządzenia (ang. *Add to Home Screen*), co eliminuje barierę instalacji i pobierania dużych plików instalacyjnych. Ponadto, dzięki mechanizmom buforowania, aplikacje te ładują się niemal natychmiastowo przy kolejnych wizytach, co jest kluczowe dla zachowania płynności interfejsu i pozytywnego odbioru systemu przez użytkownika końcowego.

W ramach realizacji niniejszej pracy zdecydowano się na wdrożenie standardu PWA, aby zapewnić wysoką dostępność aplikacji na urządzeniach mobilnych bez konieczności tworzenia osobnych wersji natywnych dla systemów Android i iOS. Wykorzystanie Service Workerów pozwoliło na zoptymalizowanie czasu ładowania kluczowych komponentów interfejsu oraz umożliwiło podstawową interakcję z systemem nawet w sytuacjach ograniczonego dostępu do Internetu. Zastosowanie manifestu aplikacji pozwoliło z kolei na nadanie projektowi profesjonalnego charakteru, upodabniając jego wygląd i zachowanie do dedykowanego oprogramowania mobilnego.

Rys. 2.8. Logo PWA
źródło: [https://www.svgrepo.com/svg/354234/pwa]

2.2.10 phpMyAdmin

phpMyAdmin jest jednym z najpopularniejszych, otwartoźródłowych narzędzi napisanym w języku PHP, służącym do graficznego zarządzania bazami danych MySQL oraz MariaDB za pośrednictwem przeglądarki internetowej. Narzędzie to stanowi kluczowy element ekosystemu webdeweloperskiego, oferując intuicyjny interfejs użytkownika (GUI), który eliminuje konieczność ręcznego wpisywania złożonych zapytań SQL w terminalu podczas wykonywania typowych operacji administracyjnych.

Głównym zadaniem phpMyAdmin jest ułatwienie interakcji z bazą danych poprzez dostarczenie szerokiego wachlarza narzędzi do manipulacji strukturą i danymi. Do najważniejszych funkcjonalności tego rozwiązania należą:

- **Zarządzanie strukturą:** tworzenie, modyfikowanie i usuwanie baz danych, tabel, kolumn oraz indeksów bez bezpośredniej edycji kodu SQL.
- **Eksport i import danych:** wsparcie dla wielu formatów plików, takich jak SQL, CSV, XML czy PDF, co znacząco ułatwia proces migracji danych oraz tworzenia kopii zapasowych.
- **Wykonywanie zapytań:** wbudowany edytor SQL z podświetlaniem składni, który pozwala na ręczne uruchamianie i testowanie zapytań oraz analizę ich wydajności.

- **Zarządzanie uprawnieniami:** panel administracyjny do tworzenia kont użytkowników bazy danych i precyzyjnego definiowania ich przywilejów na poziomie globalnym lub konkretnej tabeli.

Narzędzie to charakteryzuje się dużą mobilnością i łatwością wdrożenia, ponieważ jako aplikacja webowa nie wymaga instalacji dodatkowego oprogramowania na komputerze programisty – wystarczy dostęp do serwera, na którym phpMyAdmin jest zainstalowany. Dzięki wielojęzycznemu wsparciu oraz obsłudze relacji między tabelami (widok projektanta), staje się ono kompletnym środowiskiem do pracy nad warstwą danych aplikacji.

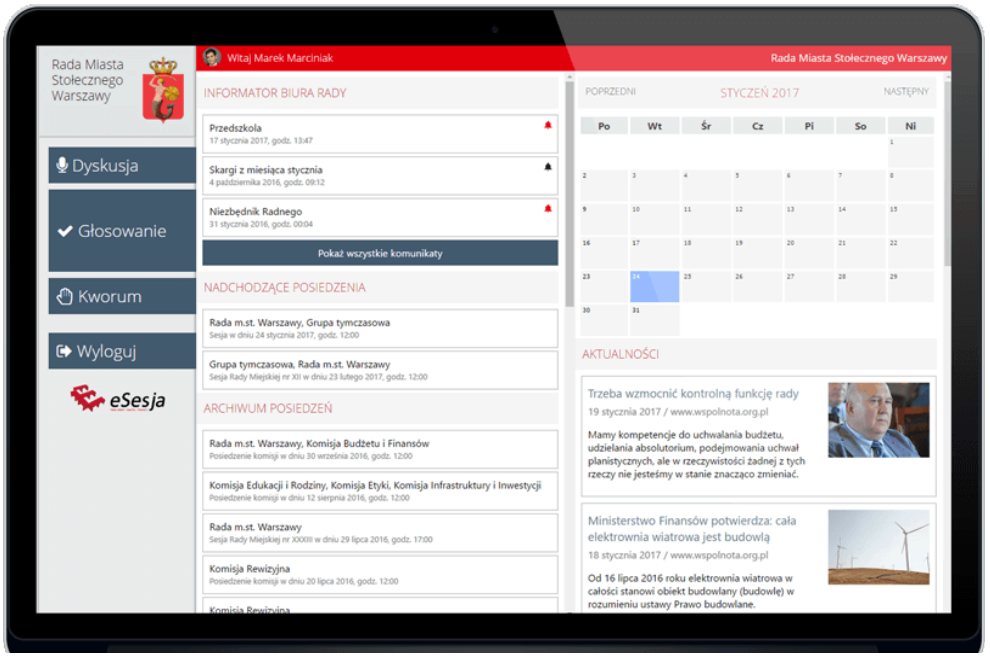
W ramach realizacji niniejszej pracy phpMyAdmin został wykorzystany jako główne narzędzie administracyjne do zarządzania bazą danych projektu. Dzięki jego zastosowaniu możliwe było szybkie definiowanie schematu bazy, wprowadzanie danych testowych oraz weryfikacja poprawności operacji wykonywanych przez backend aplikacji. Narzędzie to pozwoliło również na sprawną kontrolę spójności danych oraz ułatwiło proces debugowania zapytań SQL generowanych podczas implementacji logiki biznesowej systemu.

Rys. 2.9. Logo phpMyAdmin
źródło:
[https://www.stickpng.com/img/icons-logos-emojis/iconic-brands/phpmyadmin-logo]

3. Analiza istniejących rozwiązań

3.1. System eSesja

eSesja jest kompleksowym systemem informatycznym zaprojektowanym do cyfrowej obsługi posiedzeń organów kolegialnych, takich jak rady miast, powiatów czy zarządy instytucji. Rozwiązanie to opiera się na architekturze klient-serwer i dostarcza dedykowane moduły do zarządzania procesem legislacyjnym oraz dokumentacją spotkań. Głównym zadaniem systemu jest cyfryzacja porządku obrad, co pozwala na rezygnację z papierowego obiegu dokumentów na rzecz interaktywnego panelu radnego.



Rys. 3.1. Widok ekranu posiedzeń w systemie eSesja

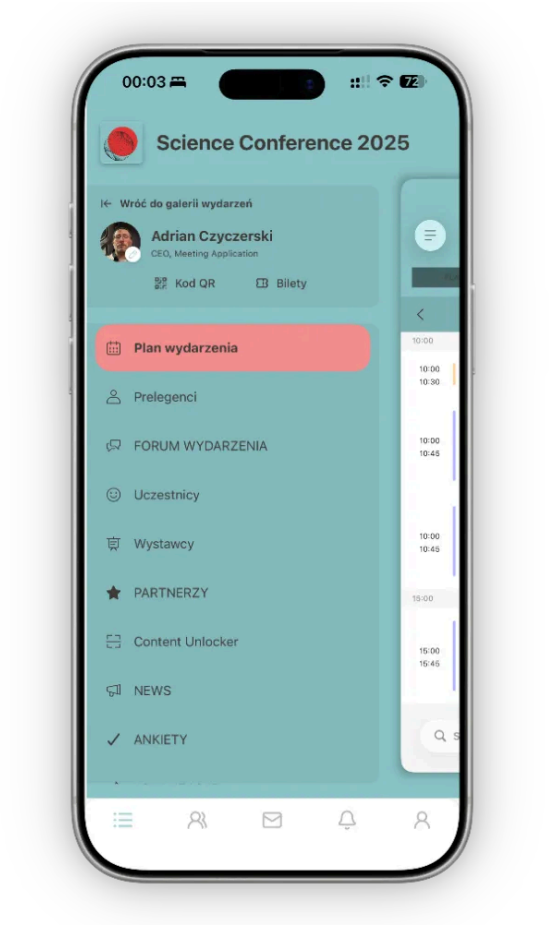
źródło: [https://esesja.pl/]

W obszarze funkcjonalnym eSesja koncentruje się na precyzyjnym definiowaniu punktów porządku dziennego oraz zarządzaniu załącznikami w formie cyfrowej. System umożliwia administratorom tworzenie struktury posiedzenia, w której każdy punkt może posiadać przypisane materiały, projekty uchwał lub odnośniki zewnętrzne. Kluczowym elementem technologicznym jest tutaj moduł autoryzacji i uwierzytelniania, który zapewnia dostęp do poufnych danych jedynie uprawnionym użytkownikom. Mimo zaawansowania w sferze administracyjnej, system ten posiada zamkniętą strukturę, która ogranicza jego wykorzystanie w kontekście otwartego katalogu organizacji czy wizualizacji danych geograficznych na mapach zewnętrznych.

Rys. 3.2. Logo systemu eSesja
Źródło: [https://tablet.esesja.pl/]

3.2. Platforma Meeting Application

Meeting Application to zaawansowane narzędzie typu SaaS (Software as a Service) przeznaczone do kompleksowej obsługi konferencji, zjazdów oraz wydarzeń masowych. Platforma ta łączy w sobie funkcje informacyjne z modułami interakcji z użytkownikiem, oferując dostęp do danych za pośrednictwem dedykowanych aplikacji mobilnych oraz interfejsów webowych. Rozwiązanie to kładzie duży nacisk na warstwę wizualną oraz intuicyjność nawigacji, co jest zbieżne z założeniami nowoczesnych frameworków front-endowych.



Rys. 3.3 Widok aplikacji Meeting Application

Źródło: [https://meetingapplication.com/

System ten oferuje rozbudowane narzędzia do zarządzania harmonogramem wydarzeń oraz bazą danych prelegentów i organizatorów. Istotnym elementem platformy jest integracja z systemami informacji geograficznej, co pozwala na prezentację punktów na mapie oraz nawigację użytkownika do konkretnych lokalizacji. Meeting Application dostarcza również mechanizmy filtrowania i wyszukiwania treści, jednak są one zorientowane głównie na agendę konkretnego wydarzenia, a nie na ogólnokrajowy katalog podmiotów o różnej specyfice. Narzędzie to nie posiada również wbudowanych mechanizmów do automatycznego generowania prezentacji multimedialnych na podstawie planu spotkania, skupiając się raczej na wyświetlaniu statycznych informacji o prelekcjach.

Rys. 3.4. Logo Meeting Application

źródło: [https://meetingapplication.com/]

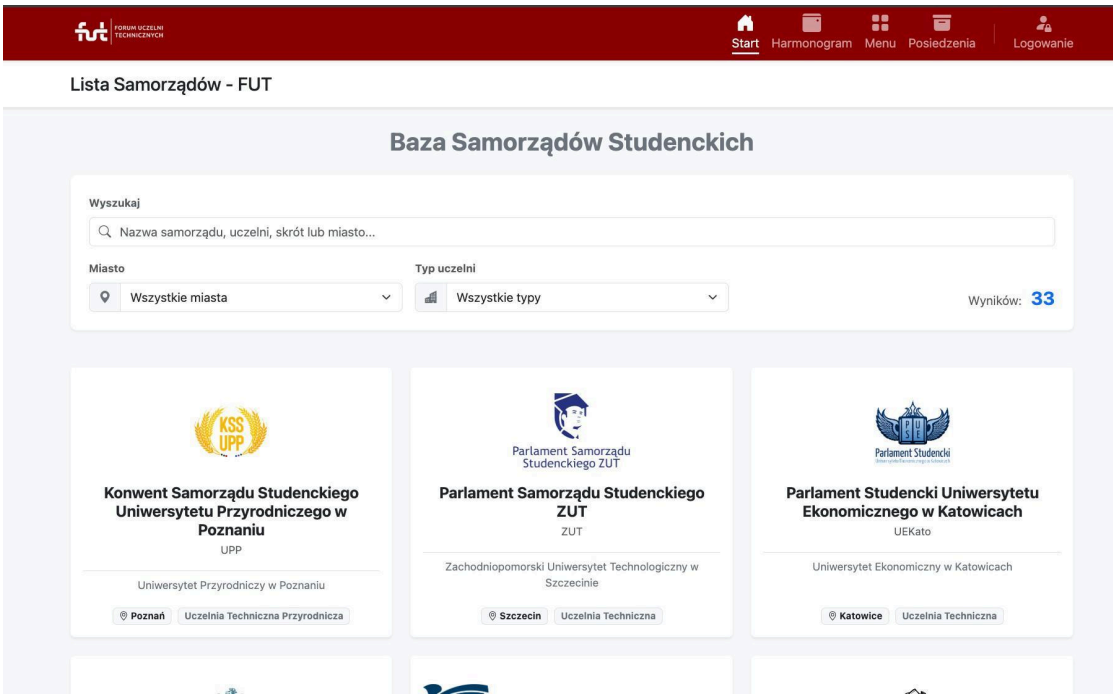
3.3 Porównanie rozwiązań w kontekście celów pracy

Analiza powyższych systemów wykazuje, że na rynku istnieją dojrzałe rozwiązania realizujące wybrane aspekty projektowanej aplikacji, jednak żadne z nich nie integruje wszystkich wymaganych funkcjonalności w jednej, spójnej platformie. System eSesja, mimo doskonałej obsługi obrad, jest rozwiązaniem zbyt sformalizowanym i pozbawionym modułów promujących organizację w formie katalogu czy interaktywnej mapy. Z kolei Meeting Application, będąc narzędziem eventowym, nie zapewnia odpowiedniego wsparcia dla procesów formalnych, takich jak zarządzanie punktami porządku dziennego posiedzeń wewnętrznych.

Projektowana aplikacja wypełnia tę lukę, łącząc mechanizmy zarządzania strukturą obrad z nowoczesną wizualizacją danych geograficznych za pomocą OpenStreetMap API oraz automatyzacją tworzenia materiałów prezentacyjnych. Wykorzystanie technologii PWA oraz frameworka Bootstrap pozwala na uzyskanie elastyczności, której brakuje systemom zamkniętym, przy jednoczesnym zachowaniu wysokiej wydajności i responsywności interfejsu użytkownika.

4.Użytkowanie

4.1.Główny widok aplikacji



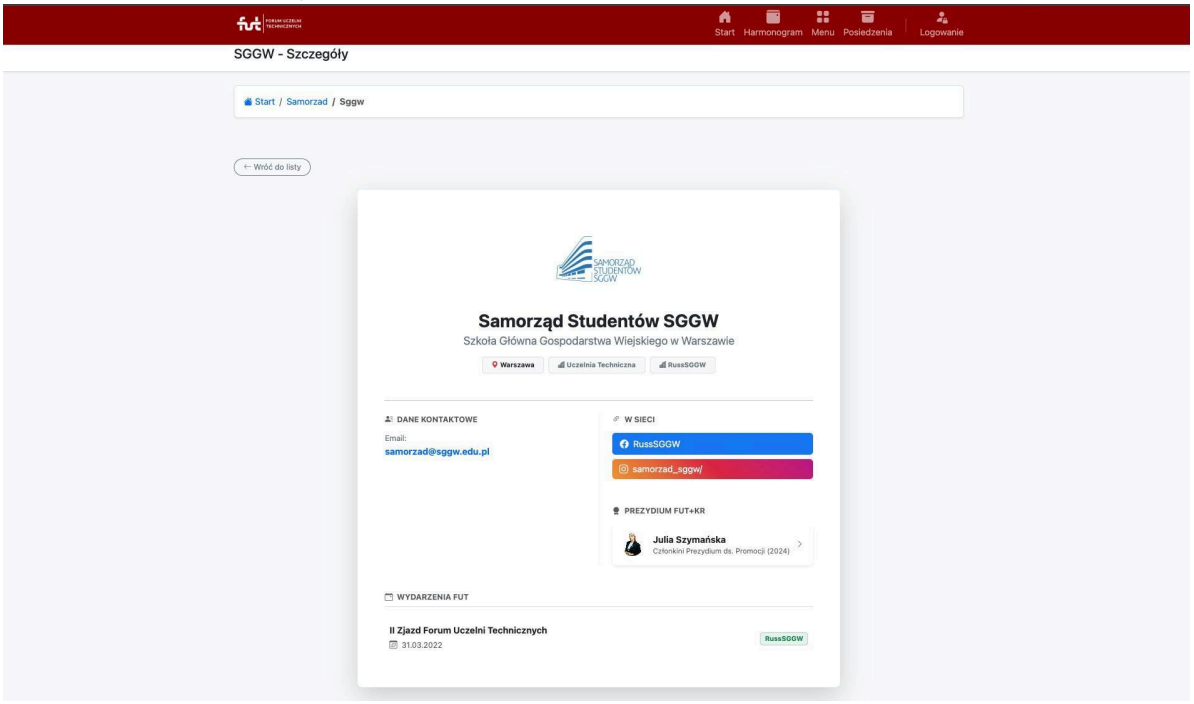
Rys. 4.1. Widok Start

Główny widok aplikacji został zaprojektowany jako centralny punkt dostępu do bazy wiedzy o samorządach studenckich, łącząc prostotę obsługi z zaawansowanymi mechanizmami przeszukiwania zbiorów danych. Centralnym elementem interfejsu jest interaktywny katalog, którego sercem jest moduł **wyszukiwania i wielokryterialnego filtrowania**. Użytkownik ma możliwość dynamicznego zawężania wyników na podstawie dwóch kluczowych parametrów: **lokalizacji (miasta)** oraz **profilu uczelni**.

Klasyfikacja według typu uczelni (np. techniczna wojskowa, techniczna przyrodnicza, techniczna) pozwala na szybką segmentację danych i odnalezienie jednostek o podobnym profilu działalności, co jest kluczowe z punktu widoku wymiany doświadczeń między samorządami. System filtrowania działa w czasie rzeczywistym, co znacząco podnosi komfort użytkowania (UX) i pozwala na błyskawiczną nawigację po ogólnopolskiej strukturze organizacji studenckich.

Górna część interfejsu została zarezerwowana dla **głównego paska nawigacyjnego**, który pełni funkcję stałego elementu kontrolnego (tzw. *Global Navigation Bar*). Zapewnia on natychmiastowy dostęp do kluczowych modułów systemu, umożliwiając intuicyjne przełączanie się między widokami bez konieczności powracania do strony startowej.

4.2.Widok samorządu



Rys. 4.2. Widok samorządu

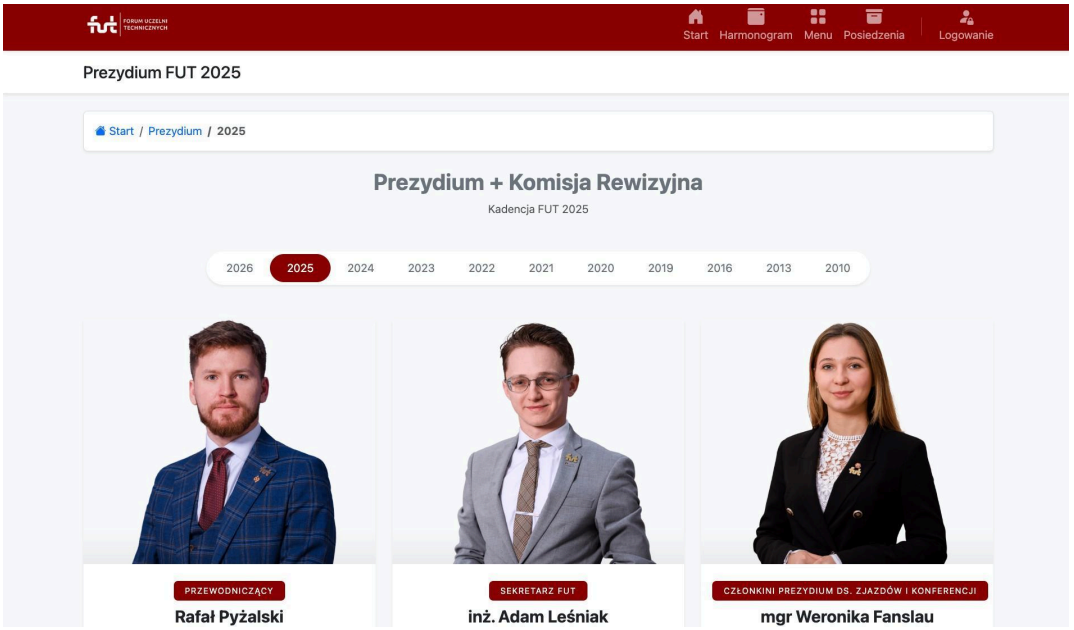
Kolejnym kluczowym elementem architektury informacji jest widok szczegółowy samorządu, który pełni rolę cyfrowej wizytówki wybranej jednostki. Ekran ten inicjowany jest poprzez akcję wyboru (kliknięcie) konkretnej pozycji list w widoku startowym. Został on zaprojektowany w sposób modułowy, co pozwala na przejrzyste zaprezentowanie zróżnicowanych danych – od bieżącej działalności operacyjnej po rys historyczny składu osobowego.

Główna sekcja tego widoku dedykowana jest zarządzaniu wydarzeniami. Wyświetlane są tutaj nadchodzące i archiwalne inicjatywy organizowane przez dany samorząd. Każde wydarzenie zintegrowane jest z modułem harmonogramu oraz współrzędnymi geograficznymi, co pozwala użytkownikowi na natychmiastowe sprawdzenie lokalizacji punktu na mapie oraz szczegółów czasowych spotkania.

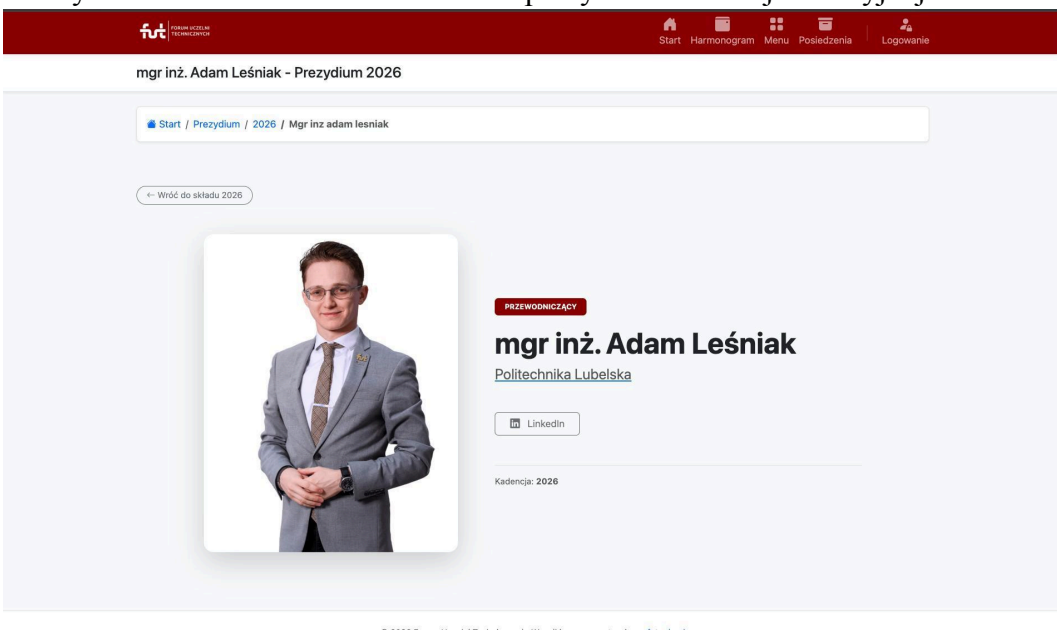
W sekcji informacyjnej szczególny nacisk położono na komunikację i obecność w sieci. Moduł ten grupuje:

- **Dane kontaktowe** – oficjalne adresy e-mail, numery telefonów oraz adresy fizyczne biur, co ułatwia nawiązywanie bezpośredniej współpracy między uczelniami.
- **Media społecznościowe** – zestaw interaktywnych odnośników (ikony social-media) prowadzących do profili samorządu (np. Facebook, Instagram,), co pozwala na bieżące śledzenie ich aktywności promocyjnej.

Unikalnym elementem widoku jest rejestr członków prezydium. W przeciwieństwie do standardowych list kontaktowych, sekcja ta prezentuje pełną strukturę osobową – obejmującą nie tylko obecne władze, ale także dane historyczne członków zarządzających jednostką. Takie rozwiązanie pozwala na budowanie bazy wiedzy o liderach studenckich i zachowanie ciągłości instytucjonalnej. Dane te są prezentowane w formie czytelnej listy lub kart pracowników, z przypisanymi rolami i okresami pełnienia funkcji.

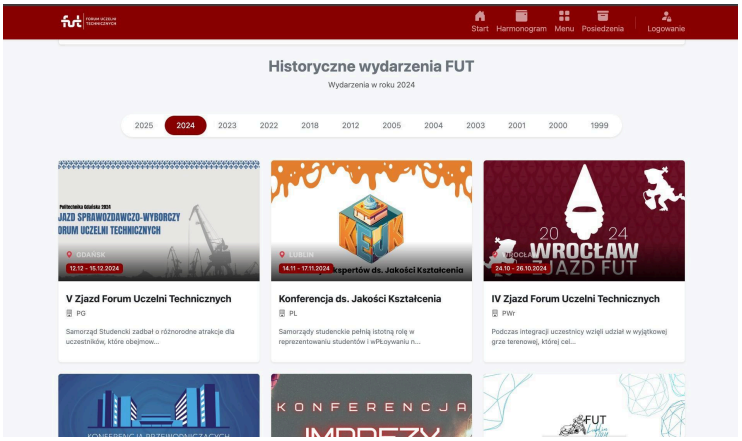


Rys. 4.3. Widok ekranu z członkami prezydium i komisji rewizyjnej FUT.



Rys. 4.4. Widok ekranu z członkiem FUT.

4.3 Widok ekranu wydarzeń

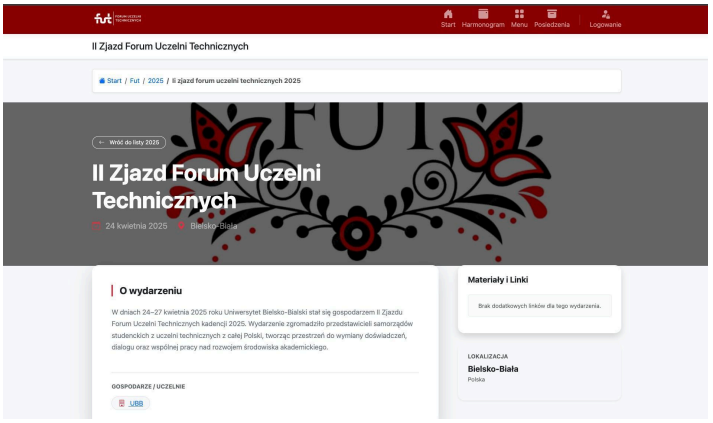


Rys. 4.5. Widok wydarzeń

Widok wydarzeń stanowi jeden z elementów systemu, łączący funkcję bieżącego kalendarza operacyjnego z historycznym repozytorium aktywności samorządowej. Centralnym punktem interfejsu jest **interaktywna oś czasu (Timeline)**, umieszczona w centralnej części ekranu. Została ona zaprojektowana jako horyzontalny suwak z czytelnym podziałem na poszczególne lata akademickie. Główne cechy funkcjonalne tego widoku to:

- **Nawigacja chronologiczna:** Użytkownik, poprzez wybór konkretnego roku na osi czasu, może błyskawicznie przełączać się między widokami. System dynamicznie filtruje rekordy w bazie danych, prezentując listę wydarzeń przypisanych do wybranego okresu.
- **Interaktywność punktów:** Każde wydarzenie na osi czasu jest reprezentowane przez aktywny węzeł. Po jego naciśnięciu system wyświetla szczegółowy panel zawierający opis celu wydarzenia oraz multimedia.

Zastosowanie osi czasu jako głównego nawigatora rozwiązuje problem "przeładowania informacyjnego". Zamiast długiej, nieczytelnej listy, użytkownik otrzymuje uporządkowaną strukturę, która wizualizuje ciągłość pracy samorządu na przestrzeni lat. Od strony technicznej, widok ten ściśle współpracuje z modułem **autoryzacji**, ograniczając możliwość edycji historycznych wpisów jedynie do użytkowników z uprawnieniami administratora, przy jednoczesnym zachowaniu powszechnego dostępu do odczytu dla wszystkich członków społeczności akademickiej.



Rys. 4.6. Widok wydarzenia

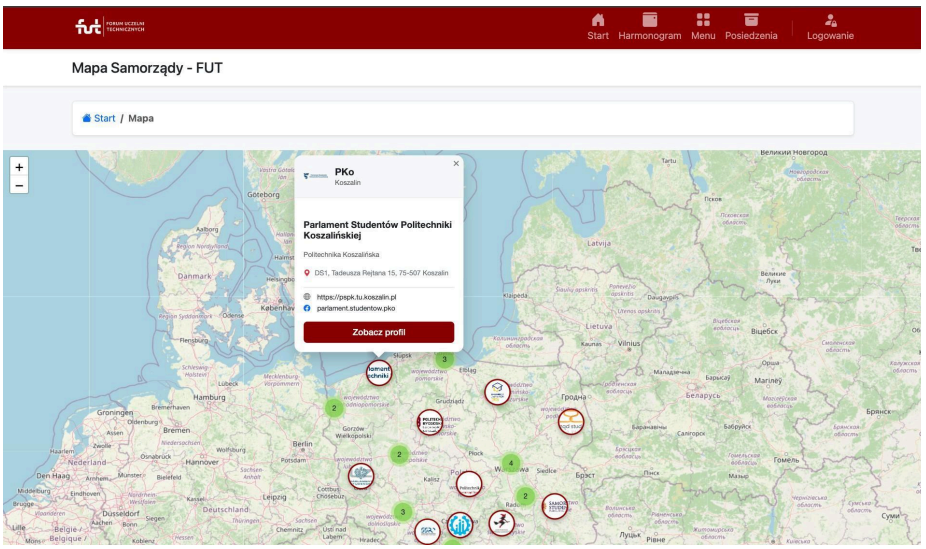
4.4. Widok interaktywnej mapy.

W ramach projektu zaimplementowano moduł interaktywnej mapy, który stanowi centralny punkt nawigacyjny dla użytkowników poszukujących informacji o jednostkach samorządowych w skali kraju. Moduł ten, dostępny bezpośrednio z poziomu menu głównego aplikacji, integruje dane przestrzenne z bazą danych, umożliwiając wizualizację lokalizacji parlamentów studenckich w formie interaktywnych znaczników (ang. *pins*). Wykorzystanie mapy jako elementu interfejsu użytkownika znacząco podnosi intuicyjność aplikacji, pozwalając na szybką identyfikację ośrodków akademickich na podstawie ich położenia geograficznego.

Każdy ze znaczników umieszczonych na mapie pełni funkcję interaktywnego punktu dostępowego do szczegółowych danych o danej jednostce. Po wybraniu konkretnej pineski, system inicjuje wyświetlenie dynamicznego okna typu pop-up, które pełni rolę skróconego profilu uczelni. Okno to zostało zaprojektowane w sposób czytelny i kompaktowy, dostarczając najważniejszych informacji bez konieczności przeładowywania strony. W ramach okna pop-up użytkownik ma wgląd w:

- **Dane adresowe:** dokładna lokalizacja siedziby samorządu.
- **Sekcję Social Media:** odnośniki do profili w mediach społecznościowych, umożliwiające szybki kontakt i podgląd bieżącej aktywności organizacji.
- **Witrynę internetową:** bezpośredni link do oficjalnej strony samorządu lub uczelni.

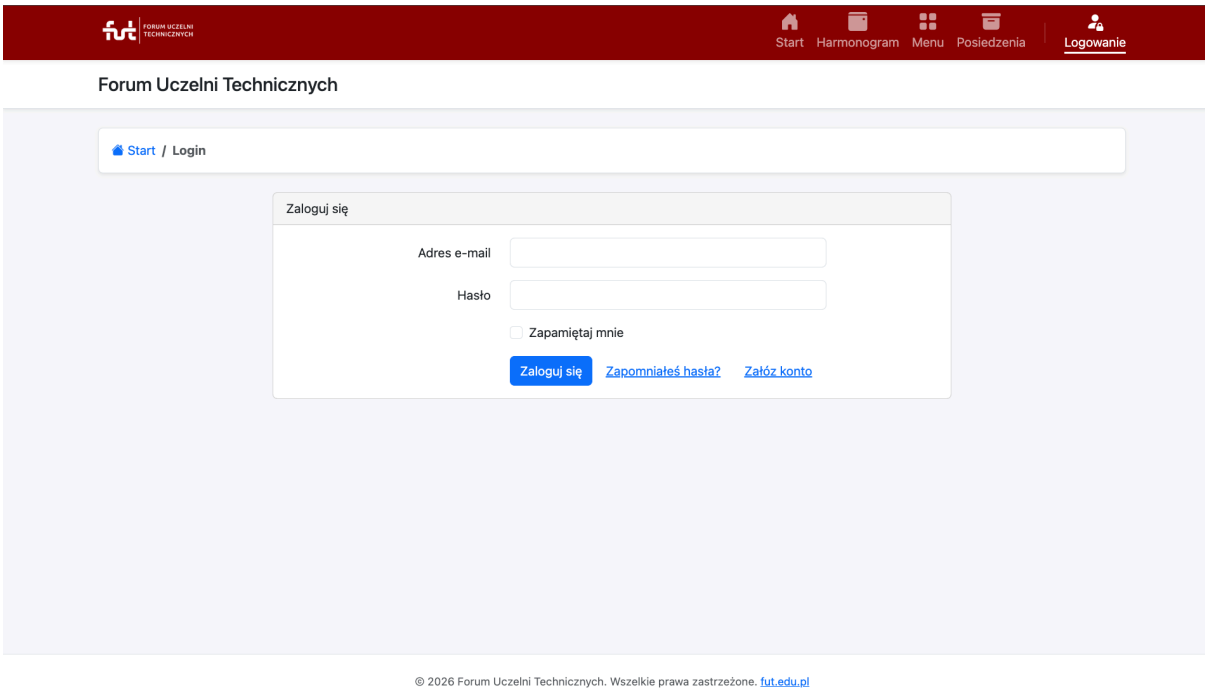
Kluczowym elementem okna informacyjnego jest przycisk funkcjonalny „Zobacz profil", który stanowi łącznik między warstwą mapy a modułem katalogowym aplikacji. Jego naciśnięcie powoduje przekierowanie użytkownika do pełnego profilu uczelni, gdzie prezentowane są rozszerzone dane, takie jak skład samorządu, historia posiedzeń czy nadchodzące wydarzenia. Dzięki takiemu rozwiązaniu mapa nie jest jedynie statycznym elementem graficznym, lecz aktywnym filtrem przestrzennym, który ułatwia eksplorację zasobów systemu.



Rys. 4.7. Widok interaktywnej mapy oraz znacznika z informacjami o uczelni.

4.5. Widok ekranu logowania.

Widok ekranu logowania stanowi kluczowy element systemu w zakresie kontroli dostępu oraz bezpieczeństwa danych. Jest to dedykowany interfejs, którego zadaniem jest przeprowadzenie procesu uwierzytelniania użytkownika przed przyznaniem mu uprawnień do edycji treści lub zarządzania zasobami aplikacji. Ekran ten został zaprojektowany z zachowaniem zasad minimalizmu i czytelności, co pozwala na szybką i intuicyjną interakcję z systemem, niezależnie od typu urządzenia, z którego korzysta użytkownik.

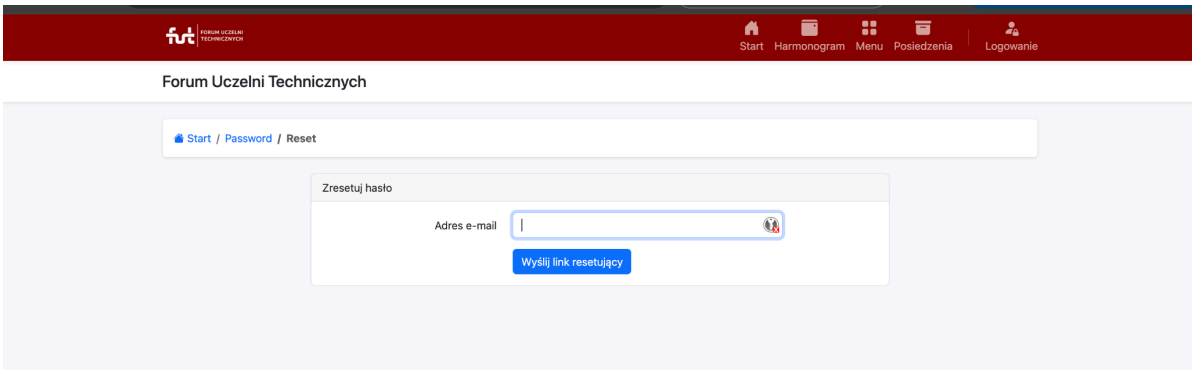


Rys. 4.8. Widok ekranu logowania.

Centralnym punktem tego modułu jest formularz logowania, składający się z pól przeznaczonych na login oraz hasło. Pola te zostały zaimplementowane z wykorzystaniem komponentów formularzy frameworka Bootstrap, co zapewnia ich pełną responsywność oraz poprawną walidację po stronie klienta. Hasło jest maskowane w czasie rzeczywistym, co stanowi podstawowy standard ochrony prywatności podczas wprowadzania danych w miejscach publicznych. Głównym elementem wykonawczym formularza jest przycisk „Zaloguj się”, który inicjuje proces weryfikacji poświadczeń w bazie danych.

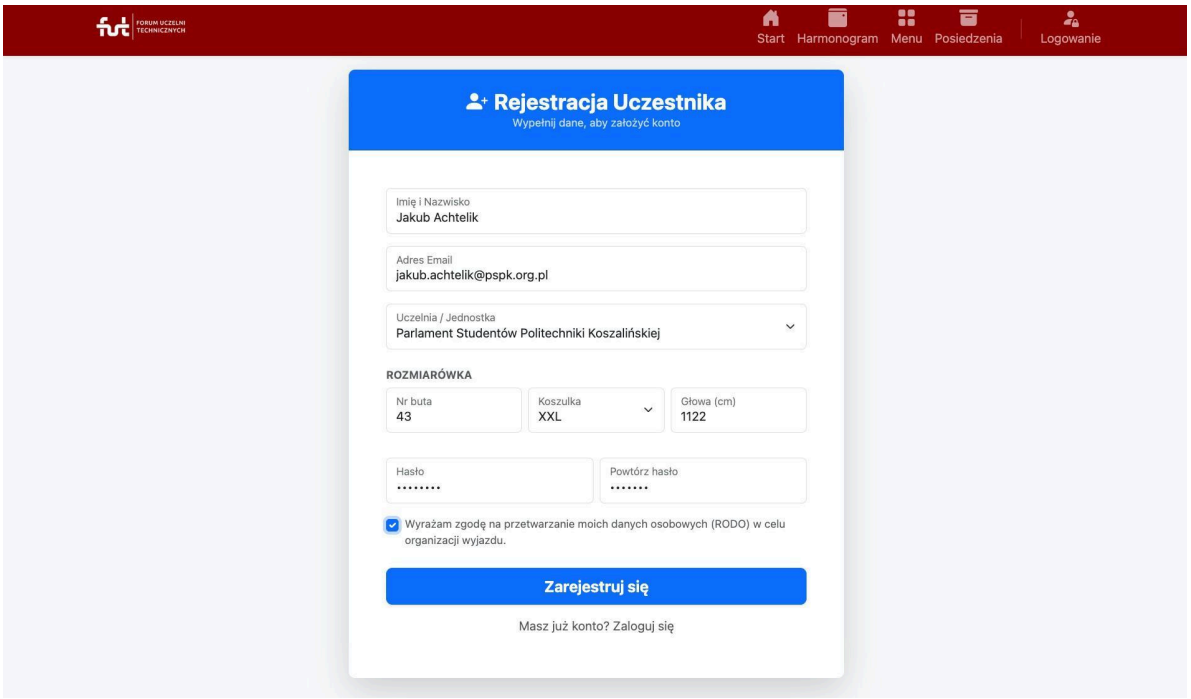
Poza główną funkcją logowania, widok ten zawiera zestaw opcji pomocniczych, które wspierają użytkownika w sytuacjach problematycznych lub podczas pierwszego kontaktu z systemem:

- **Funkcja odzyskiwania dostępu:** Przycisk „Zapomniałeś hasła?” kieruje użytkownika do procedury resetowania poświadczeń, co jest niezbędne w systemach wymagających wysokiej dostępności.



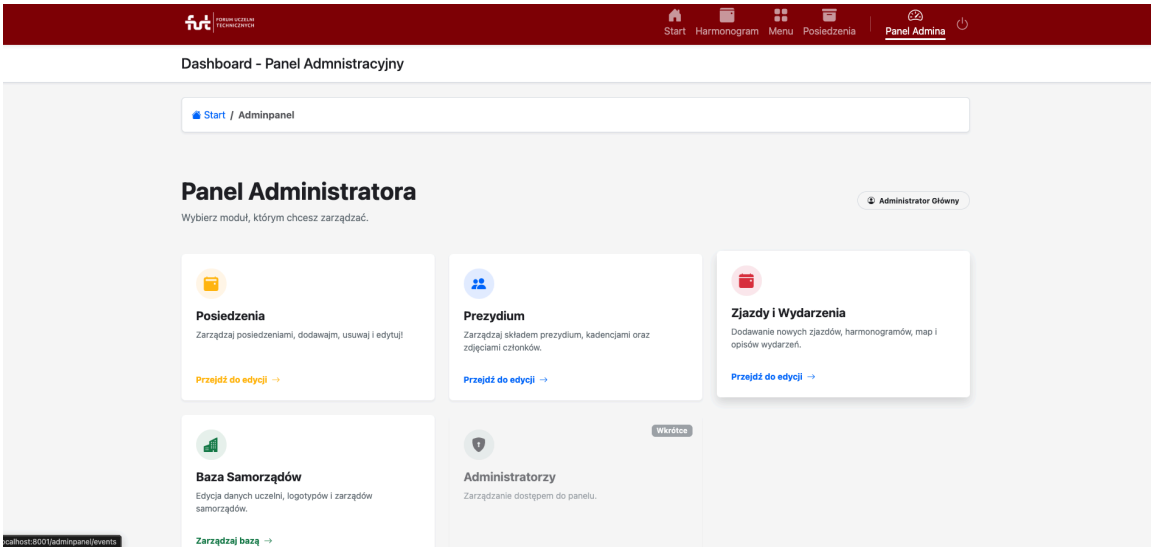
Rys. 4.9. Widok odzyskiwania hasła

- **Rejestracja nowych użytkowników:** Odnośnik „Załóż konto” pozwala na przejście do modułu tworzenia nowego profilu, co umożliwia organiczny wzrost liczby użytkowników systemu przy zachowaniu zdefiniowanych reguł walidacji danych.



Rys. 4.10. Widok ekranu rejestracji.

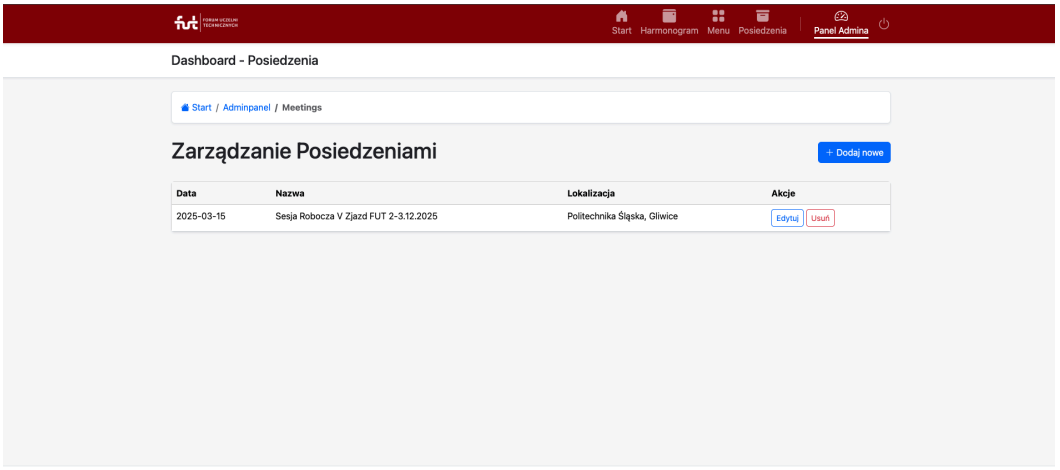
4.6. Moduł panelu administracyjnego i zarządzania systemem.



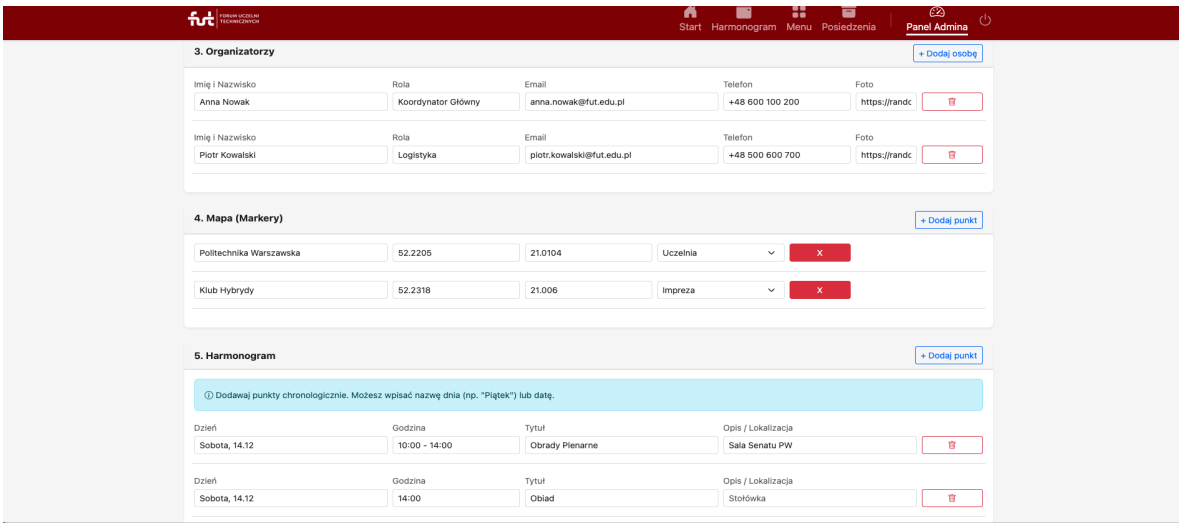
Rys. 4.11. Widok panelu administratora

Panel administracyjny stanowi nadrzędną warstwę funkcjonalną aplikacją, zaprojektowaną jako scentralizowany interfejs do zarządzania kluczowymi zasobami systemu. Widok ten jest dostępny wyłącznie dla użytkowników o podwyższonych uprawnieniach i pełni rolę panelu sterowania (ang. dashboard), integrującego mechanizmy CRUD dla wszystkich modułów biznesowych projektu. Interfejs panelu został zoptymalizowany pod kątem wydajności pracy, oferując szybki dostęp do edycji danych strukturalnych i personalnych. Kluczowym komponentem panelu administracyjnego jest rozbudowany system zarządzania wydarzeniami formalnymi, podzielony na dwa główne obszary:

Edycja zjazdów i posiedzeń: Moduł ten pozwala na pełną kontrolę nad cyklem życia spotkań – od ich planowania, przez edycję punktów porządku dziennego, aż po dołączanie materiałów w formie załączników URL. Administrator może w czasie rzeczywistym korygować harmonogram, co jest kluczowe dla zachowania spójności danych prezentowanych użytkownikom końcowym.

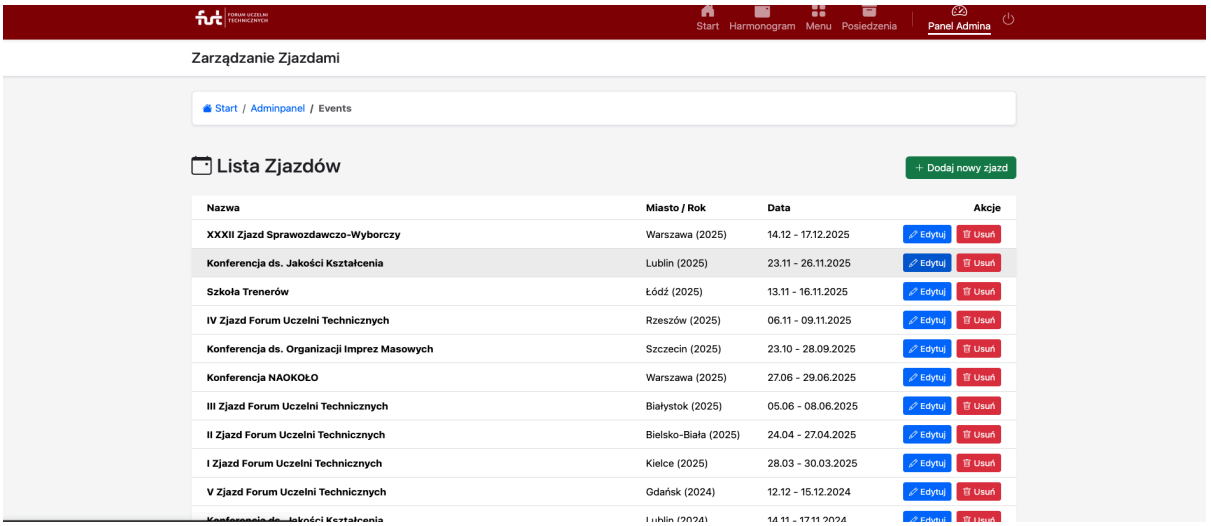


Rys. 4.12. Widok zarządzanie posiedzeniami



Rys. 4.13. Edycja informacji o wydarzeniu: Organizatorzy, Edycja, Harmonogram

Zarządzanie prezydium: Funkcjonalność ta umożliwia dynamiczną aktualizację składu organów decyzyjnych. System pozwala na modyfikację danych osobowych, pełnionych funkcji oraz okresu kadencyjności członków prezydium, co zapewnia aktualność informacji kontaktowych prezentowanych w katalogu.



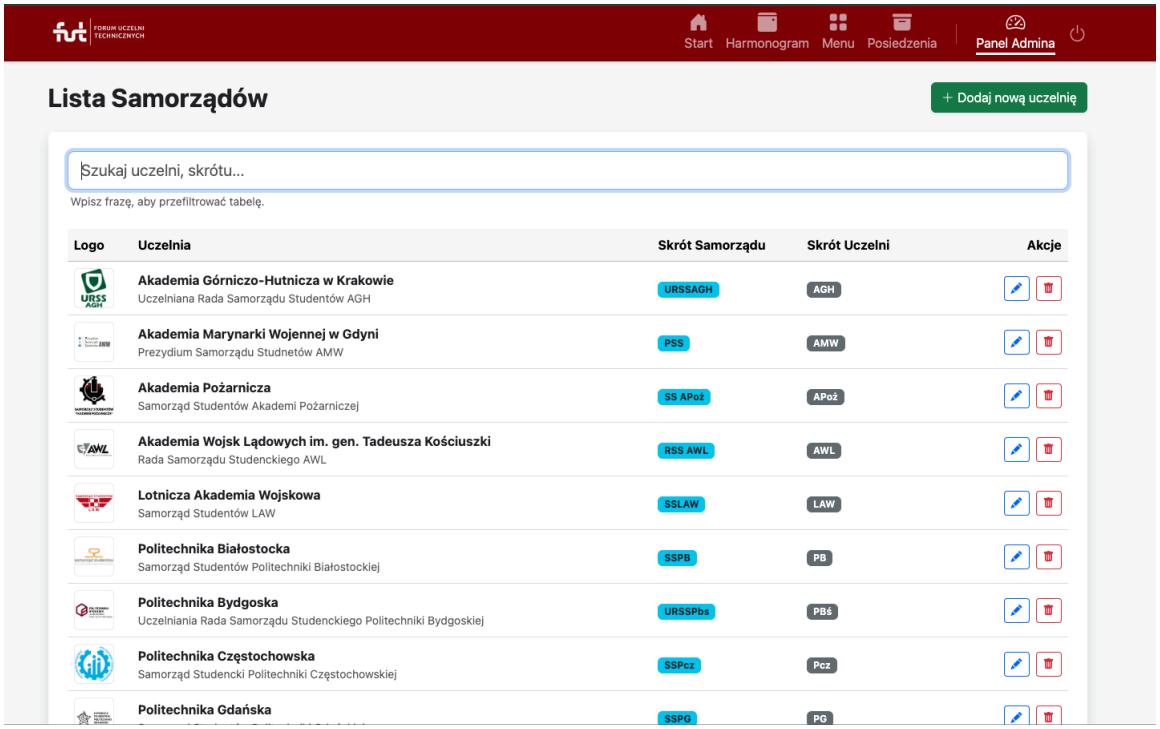
Rys. 4.14. Lista zjazdów

Istotnym elementem panelu jest również moduł zarządzania bazą samorządów. Interfejs ten pozwala na administrację rekordami dotyczącymi uczelni wyższych, ich lokalizacji geograficznej oraz przypisanych do nich kont użytkowników. Dzięki integracji z systemem siatki i tabelami frameworka Bootstrap, administrator może w łatwy sposób filtrować listę organizacji, masowo edytować ich parametry oraz weryfikować poprawność współrzędnych geograficznych, które zasilają moduł interaktywnej mapy.

4.7. Moduł Katalogu Samorządów Studenckich

Katalog samorządów studenckich stanowi kluczowy komponent administracyjny systemu, zaprojektowany w formie przejrzystego i wysoce funkcjonalnego panelu zarządzania bazą danych jednostek.

Centralnym punktem interfejsu jest moduł wyszukiwania, umieszczony w osiowej części ekranu nad tabelą danych. Pozwala on na dynamiczne filtrowanie rekordów na podstawie nazwy samorządu w trybie *real-time search*. Dzięki temu administrator może błyskawicznie odnaleźć konkretną jednostkę bez konieczności ręcznego przeszukiwania całego zbioru danych.

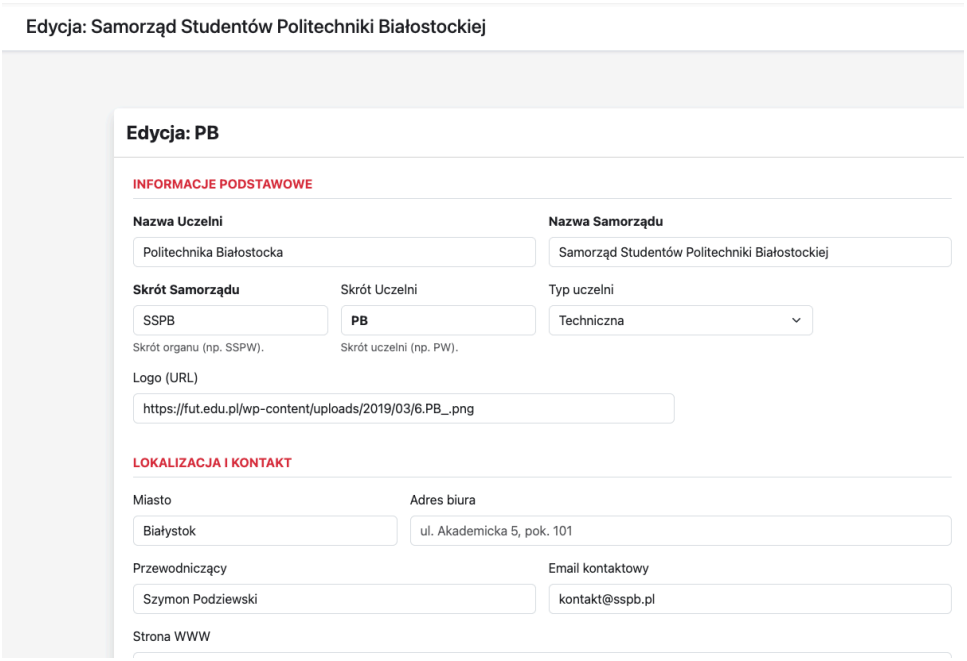


Rys. 4.15. Widok katalogu samorządów studenckich w panelu administratora

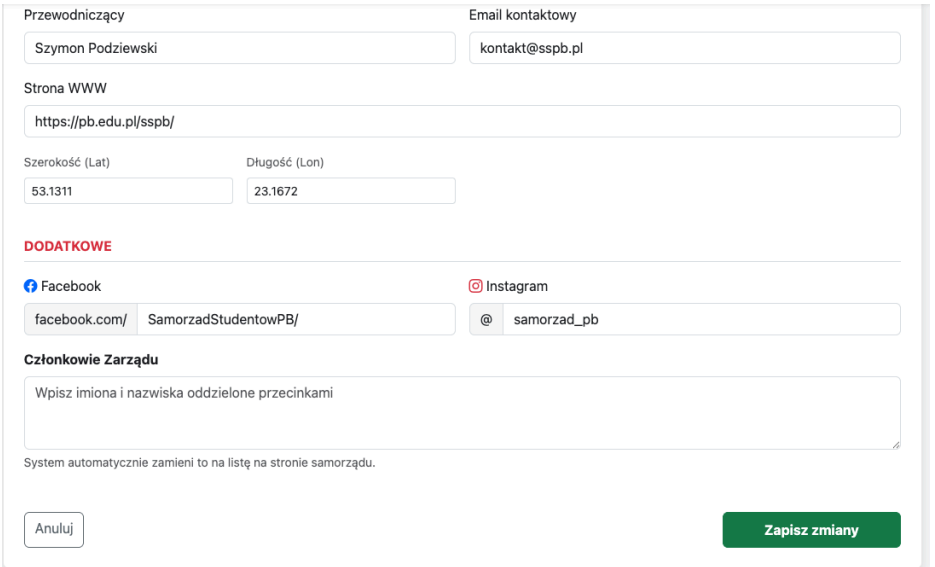
Dane w katalogu prezentowane są w formie pięciokolumnowej tabeli, która w sposób syntetyczny dostarcza najważniejszych informacji o każdej organizacji:

1. Logo – kolumna wizualna zawierająca miniatury oficjalnych logotypów samorządów. Pozwala to na szybką identyfikację graficzną marki danej uczelni i podnosi estetykę interfejsu.
2. Uczelnia – pełna nazwa jednostki szkolnictwa wyższego, przy której funkcjonuje dany samorząd. Dane te są zintegrowane z globalnym filtrem typu uczelni (np. techniczna, wojskowa).
3. Skrót samorządu – unikalna nazwa skrócona lub akronim organizacji (np. „RSS", „SSPW"), powszechnie używana w komunikacji wewnętrznej i dokumentacji obrad.
4. Skrót uczelni – oficjalny akronim uczelni (np. „PW", „UAM", „WAT"), ułatwiający szybkie sortowanie i grupowanie jednostek z tego samego regionu lub profilu.
5. Akcje (Edycja i Usuwanie) – kolumna operacyjna zawierająca interaktywne kontrolki

zarządzania rekordem. Przycisk edycji inicjuje przejście do formularza modyfikacji danych kontaktowych i prezydium, natomiast przycisk usuwania pozwala na bezpieczne wyłączenie jednostki z katalogu.



Rys. 4.16. Widok edycji informacji o samorządzie - Informacje podstawowe i Kontakt



Rys. 4.17. Widok edycji informacji o samorządzie - Dodatkowe

Dzięki takiemu układowi, katalog pełni rolę centrum dowodzenia dla administratora systemu. Spójność danych między skrótami a pełnymi nazwami, w połączeniu z kontrolą wizualną (logo), minimalizuje ryzyko błędów przy zarządzaniu dużą bazą samorządów. System uprawnień dba o to, aby kolumna „Akcje” była widoczna i dostępna wyłącznie dla osób posiadających odpowiednie role w module autoryzacji, co zapewnia bezpieczeństwo integralności danych całego katalogu.

5. Projekt i implementacja

5.1. Architektura aplikacji MVC

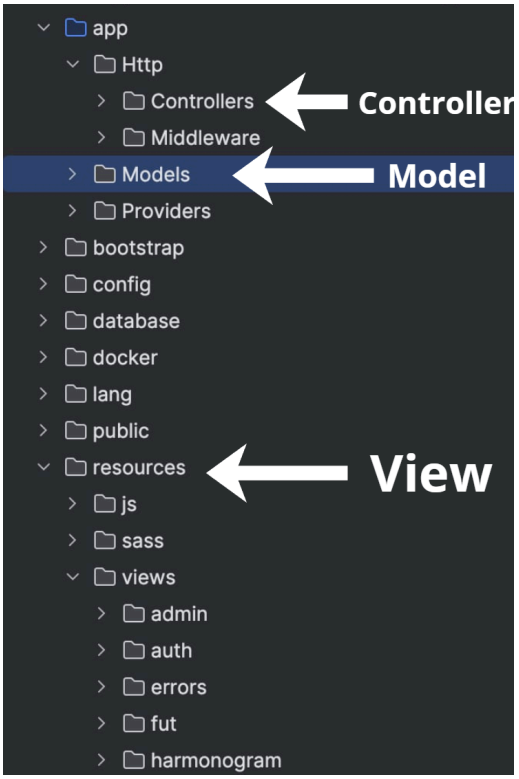
Projekt został zrealizowany w oparciu o wzorzec architektoniczny Model-View-Controller , który stanowi standard w tworzeniu skalowalnych i łatwych w utrzymaniu aplikacji internetowych. Głównym założeniem tej architektury jest podział aplikacji na trzy niezależne warstwy, co pozwala na odseparowanie logiki biznesowej od interfejsu użytkownika oraz mechanizmów dostępu do danych. Dzięki takiemu podejściu, modyfikacje w warstwie wizualnej (np. zmiana stylów Bootstrap) nie wpływają na stabilność logiki przetwarzania danych.

Struktura systemu w ramach wzorca MVC prezentuje się następująco:

- **Model (Model):** Odpowiada za bezpośrednie zarządzanie danymi oraz logiką biznesową systemu. To tutaj zdefiniowane są relacje między obiektami, takimi jak samorządy, posiedzenia czy punkty agendy. Model komunikuje się z bazą danych (np. MariaDB), wykonując operacje zapisu i odczytu, a następnie przekazuje przetworzone informacje do kontrolera.
- **Widok (View):** Stanowi warstwę prezentacji, odpowiedzialną za generowanie interfejsu użytkownika. W ramach tego komponentu wykorzystywane są pliki HTML, klasy frameworka Bootstrap oraz skrypty JavaScript obsługujące m.in. OpenStreetMap API. Widok nie posiada wiedzy o strukturze bazy danych – jego zadaniem jest jedynie czytelne wyświetlenie danych przekazanych przez kontroler.
- **Kontroler (Controller):** Pełni rolę pośrednika (ang. *mediator*), który zarządza przepływem informacji w aplikacji. Kontroler odbiera żądania użytkownika (np. kliknięcie przycisku „Pokaż profil”), komunikuje się z odpowiednim modelem w celu pobrania danych, a następnie wybiera właściwy widok, który ma zostać wyrenderowany.

Zastosowanie architektury MVC w niniejszej pracy przyniosło szereg korzyści technicznych. Przede wszystkim umożliwiło równoległe prowadzenie prac nad warstwą frontendową (PWA, Bootstrap) i backendową. Ponadto, modularność tego rozwiązania ułatwiła implementację zaawansowanych funkcji, takich jak automatyzacja prezentacji – logika generowania plików multimedialnych została zamknięta w dedykowanych komponentach modelu, podczas gdy kontroler zarządza jedynie momentem ich wywołania na podstawie interakcji w panelu administracyjnym.

Pod kątem utrzymania systemu, wzorzec MVC zapewnia wysoką przejrzystość kodu źródłowego. Każdy element aplikacji ma precyzyjnie zdefiniowane miejsce i rolę, co znacząco redukuje ryzyko powstawania błędów typu „spaghetti code”. Jest to szczególnie istotne w kontekście konteneryzacji Dockerem, gdzie spójna i uporządkowana struktura plików aplikacji przekłada się na szybsze budowanie obrazów oraz łatwiejsze wdrażanie systemu w różnych środowiskach serwerowych.



Rys. 5.1. Struktura katalogów projektu

5.2. Routing aplikacji

Routing jest jednym z podstawowych mechanizmów wykorzystywanych w frameworku Laravel, odpowiedzialnym za obsługę żądań HTTP kierowanych do aplikacji. Mechanizm ten polega na mapowaniu adresów URL na odpowiednie metody kontrolerów, które następnie przetwarzają żądanie i zwracają odpowiedź do użytkownika. Dzięki zastosowaniu routingu możliwe jest uporządkowanie struktury aplikacji oraz logiczne przypisanie poszczególnych funkcjonalności do konkretnych adresów w systemie.

W frameworku Laravel definicje tras (ang. routes) zapisywane są najczęściej w pliku **routes/web.php**, który odpowiada za obsługę żądań HTTP kierowanych do aplikacji webowej. W pliku tym programista może określić, jaki kod powinien zostać wykonany po wywołaniu konkretnego adresu URL. Trasy mogą wskazywać bezpośrednio na funkcję anonimową lub na metodę kontrolera odpowiedzialnego za obsługę danej funkcjonalności.

Laravel obsługuje różne metody protokołu HTTP, takie jak GET, POST, PUT, PATCH czy DELETE. Dzięki temu możliwe jest tworzenie pełnych interfejsów API oraz implementacja operacji CRUD dla różnych zasobów aplikacji. Przykładowo metoda GET wykorzystywana jest do pobierania danych, natomiast POST do ich zapisywania w systemie.

Istotnym elementem routingu w Laravel jest także możliwość definiowania tras z parametrami. Parametry pozwalają na dynamiczne przekazywanie danych w adresie URL, takich jak identyfikator konkretnego obiektu w bazie danych. Dzięki temu możliwe jest wyświetlanie szczegółowych informacji o danym rekordzie, na przykład wydarzeniu lub użytkowniku, na podstawie przekazanego identyfikatora.

5.3. Database Seeders – mechanizm wypełniania bazy danych

Database Seeders to wbudowany w framework Laravel mechanizm służący do automatycznego wypełniania bazy danych danymi testowymi lub inicjalnymi (tzw. danymi stałymi). W procesie tworzenia złożonych systemów, takich jak aplikacja do zarządzania samorządami, ręczne wprowadzanie rekordów do bazy danych jest nieefektywne i podatne na błędy. Seedery pozwalają na zdefiniowanie zestawów danych w formie kodu PHP, co umożliwia ich błyskawiczne i powtarzalne osadzenie w strukturze tabel przy użyciu prostego polecenia w terminalu.

5.4. Technologia PWA

Progressive Web App to standard tworzenia aplikacji internetowych, który zaciera granicę między klasyczną stroną WWW a natywną aplikacją mobilną. Wykorzystanie tej technologii w projekcie miało na celu zapewnienie użytkownikom wysokiego komfortu pracy na urządzeniach przenośnych przy jednoczesnym zachowaniu jednej bazy kodu (ang. *single codebase*). PWA nie wymaga instalacji poprzez dedykowane sklepy z aplikacjami, co znacząco obniża barierę wejścia i pozwala na natychmiastowe uruchomienie systemu z poziomu przeglądarki internetowej.

Fundament techniczny PWA opiera się na trzech kluczowych komponentach, które zostały zaimplementowane w aplikacji:

- **Service Workers:** Są to skrypty JavaScript działające w tle, niezależnie od głównego wątku przeglądarki. Pełnią one rolę serwera proxy pomiędzy aplikacją a siecią. Service Worker odpowiada za przechwytywanie zapytań sieciowych i zarządzanie mechanizmem buforowania (ang. *caching*). Dzięki temu kluczowe zasoby interfejsu, takie jak style Bootstrap czy skrypty obsługujące mapę, są dostępne natychmiastowo, nawet przy niestabilnym połączeniu internetowym.
- **Web App Manifest:** Plik konfiguracyjny w formacie JSON, który dostarcza przeglądarce informacji o tym, jak aplikacja powinna zachowywać się po zainstalowaniu na urządzeniu. Definiuje on m.in. nazwę aplikacji, zestaw ikon o różnych rozdzielczościach, kolory motywu paska systemowego oraz tryb wyświetlania (ang. *display mode*). Ustawienie parametru na standalone pozwala na ukrycie interfejsu przeglądarki, dzięki czemu aplikacja wypełnia cały ekran, imitując natywne oprogramowanie.

Zastosowanie PWA w niniejszej pracy pozwoliło na wdrożenie funkcjonalności **Add to Home Screen**, dzięki której użytkownicy mogą dodać skrót do systemu bezpośrednio na pulpit smartfona lub tabletu. Z perspektywy programistycznej, wybór tej technologii wyeliminował konieczność tworzenia i utrzymywania osobnych aplikacji dla systemów Android oraz iOS. Jednocześnie zapewniono wsparcie dla powiadomień systemowych (ang. *Push Notifications*), co w przyszłości może posłużyć do informowania członków prezydium o nadchodzących posiedzeniach lub zmianach w porządku obrad.

Dzięki integracji PWA z architekturą **MVC** oraz frameworkiem **Bootstrap**, aplikacja charakteryzuje się bardzo krótkim czasem ładowania (ang. *First Contentful Paint*)

oraz płynnością działania, która jest kluczowa w mobilnym dostępie do danych geograficznych i dokumentacji posiedzeń.

Rys. 5.2. Ikona aplikacji w systemie macOS

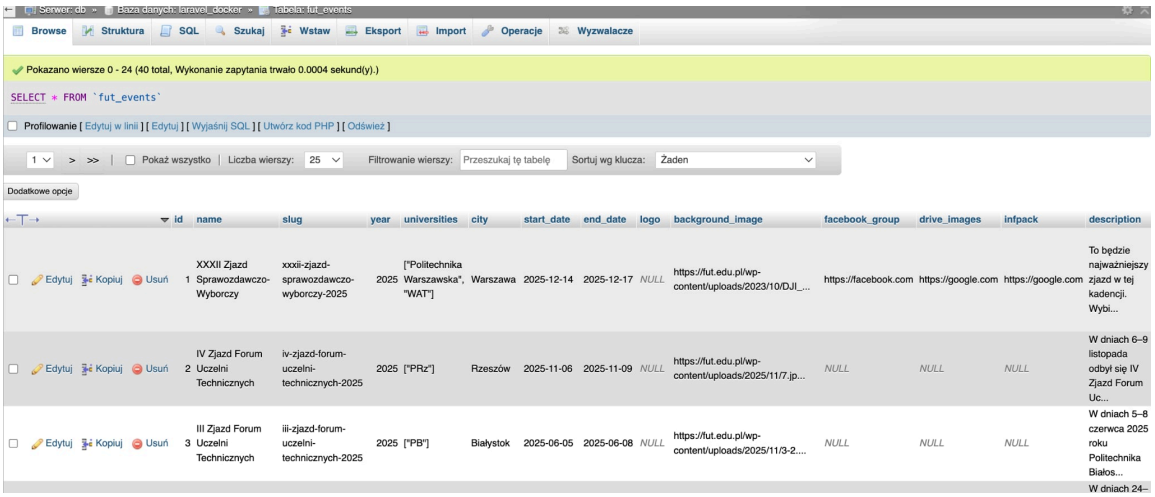
```
{
    "name":  "FUTApp  2.0", " short_name":
    "FUTApp", "  start_url":        "/", "
    background_color": "# ffffff", "
    theme_color":  "#8B0000", " display":   "
    standalone", " orientation": " portrait",
    "icons": [
        {
            "src": "/ images/icons/icon -192x192.png", "type": "image/png",
            "sizes":  "192x192"
        },
        {
            "src": "/ images/icons/icon -512x512.png", "type": "image/png",
            "sizes":  "512x512"
        }
    ]
}
```

Rys. 5.3. Konfiguracja pliku manifest.json dla aplikacji PWA

To jest plik konfiguracyjny Web App Manifest, który przekształca Twoją stronę internetową w aplikację typu PWA. Dzięki niemu telefon lub komputer traktuje stronę jak instalowalną aplikację.
Oto co oznaczają poszczególne parametry (idealne do opisu w pracy):
name / short_name: Pełna i skrócona nazwa aplikacji (widoczna pod ikoną na pulpicie).

- start_url: Określa, od której strony ma się uruchomić aplikacja po kliknięciu w ikonę (tutaj: strona główna /).
- theme_color: Kolor paska systemowego (np. kolor góry ekranu w Androidzie). Wybrałeś #8B0000 (ciemna czerwień), co pasuje do identyfikacji wizualnej Samorządu/Uczelni.
- display:standalone: To najważniejszy parametr – sprawia, że aplikacja otwiera się w własnym oknie bez paska adresu przeglądarki, dzięki czemu wygląda jak natywna aplikacja ze sklepu (App Store/Google Play).
- orientation: portrait: Wymusza wyświetlanie aplikacji tylko w pionie.
- icons: Wskazuje ścieżki do grafik, które będą służyć jako ikona na pulpicie i ekran startowy (Splash Screen).

5.5 Baza danych



Rys. 5.4. Widok tabeli fut_events w phpMyAdmin

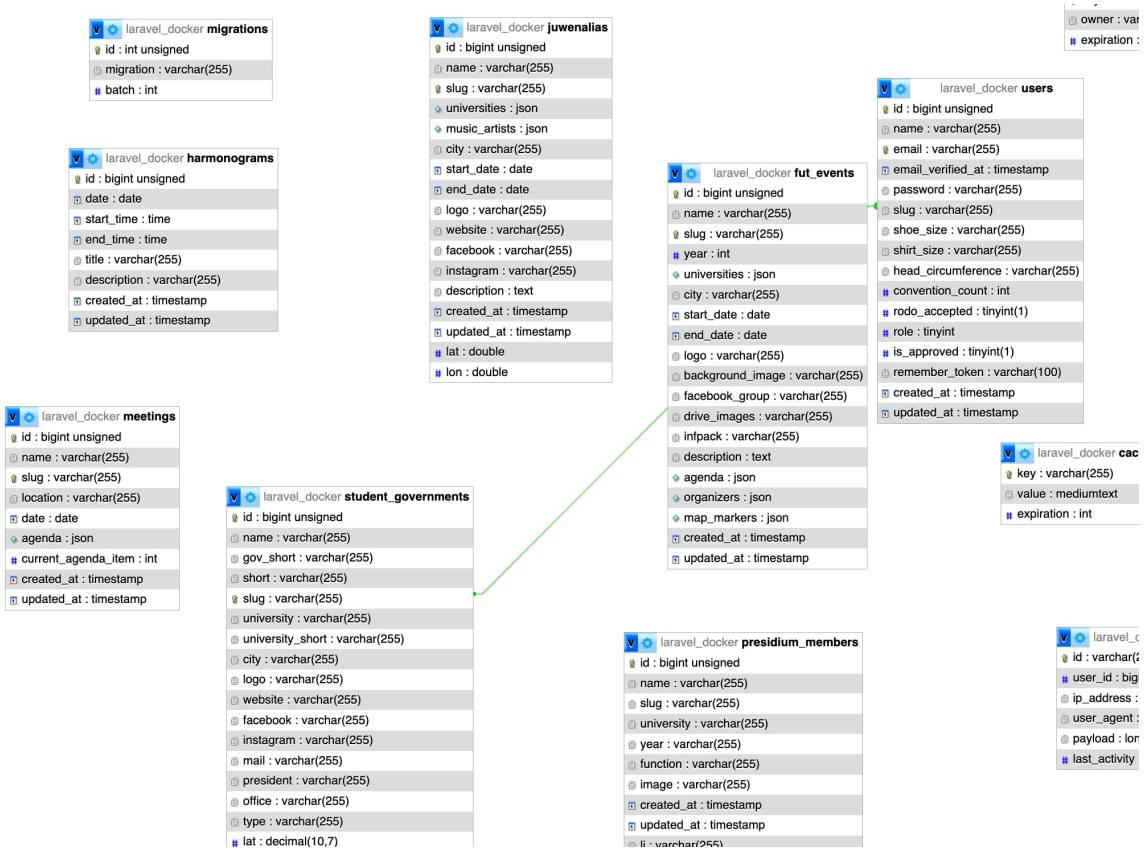
W projekcie zastosowano bazę danych zarządzaną przy użyciu systemu MySQL, obsługiwaną administracyjnie poprzez narzędzie phpMyAdmin. Proces projektowania oraz implementacji struktury danych został zrealizowany zgodnie z paradygmatem Code-First. Oznacza to, że schemat bazy nie był definiowany bezpośrednio w systemie zarządzania bazą danych (np. za pomocą języka SQL), lecz został w całości wygenerowany z poziomu kodu źródłowego. Baza danych została zintegrowana z frameworkiem Laravel, którego wbudowany mechanizm migracji automatycznie tłumaczy kod PHP na strukturę tabel. Podejście to zapewnia pełną wersjonowalność schematu bazy oraz ułatwia bezkolizyjne odtwarzanie środowiska na różnych etapach tworzenia oprogramowania.

W bazie danych znajduje się kilka tabel odpowiadających za różne funkcjonalności systemu:

users – tabela przechowująca dane użytkowników systemu, takie jak identyfikator, dane logowania oraz inne informacje niezbędne do autoryzacji i zarządzania kontami.
sessions – tabela odpowiedzialna za przechowywanie informacji o aktywnych sesjach użytkowników, co umożliwia zarządzanie logowaniem i bezpieczeństwem systemu.
student_governments – tabela zawierająca informacje o samorządach studenckich działających w systemie.
presidium_members – przechowuje dane członków prezydium samorządu studenckiego, umożliwiając zarządzanie strukturą organizacyjną.
meetings – tabela zawierająca informacje o spotkaniach organizowanych w ramach samorządu, takich jak daty, szczegóły spotkań czy uczestnicy.
harmonograms – przechowuje harmonogramy wydarzeń lub spotkań, umożliwiając planowanie działań w systemie.
fut_events – tabela przeznaczona do zapisywania informacji o wydarzeniach organizowanych w systemie.
juwenalias – tabela przechowująca dane dotyczące wydarzeń związanych z juwenaliami.

Dodatkowo w bazie znajdują się tabele systemowe generowane automatycznie przez framework Laravel:

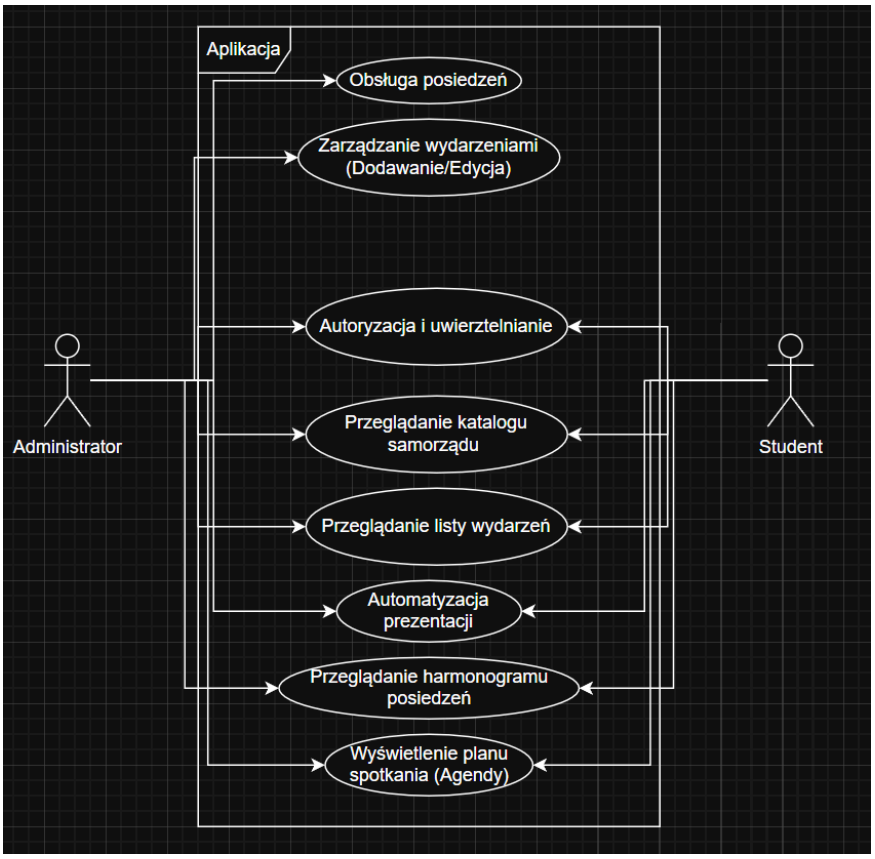
migrations – przechowuje informacje o wykonanych migracjach bazy danych, umożliwiając kontrolę wersji struktury bazy.
cache oraz **cache_locks** – odpowiadają za przechowywanie danych cache, co przyspiesza działanie aplikacji.
jobs, job_batches, failed_jobs – wykorzystywane do obsługi zadań wykonywanych w tle przez system.



Rys. 5.5. Tabele w bazie danych

Istotną decyzją architektoniczną podjętą na etapie projektowania bazy danych była rezygnacja z definicji twardych więzów integralności (kluczy obcych) na poziomie samego silnika bazy danych. Zamiast tego, zarządzanie relacjami pomiędzy poszczególnymi encjami zostało w pełni przeniesione do warstwy logiki aplikacji. Integracja oraz dbałość o spójność danych są obsługiwane przez zaawansowany system mapowania obiektowo-relacyjnego (ORM) Eloquent. Rozwiązanie to pozwoliło na zwiększenie elastyczności struktury, ułatwiło proces migracji danych oraz umożliwiło szybsze wprowadzanie zmian w modelach bez konieczności każdorazowej przebudowy schematu bazy na poziomie serwera MySQL.

5.6. Przypadki użycia



Rys. 5.6. Diagram przypadków użycia

Przypadek użycia: PU1 – Obsługa posiedzeń

Aktor: Administrator

Opis: Proces zarządzania posiedzeniami w systemie, obejmujący tworzenie, edycję oraz przeglądanie informacji o zaplanowanych posiedzeniach.

Warunki wstępne: Administrator jest zalogowany do systemu.

Przebieg:
Administrator loguje się do systemu.
Administrator przechodzi do sekcji zarządzania posiedzeniami.
System wyświetla listę istniejących posiedzeń.
Administrator może dodać nowe posiedzenie lub edytować istniejące.
Administrator wprowadza szczegóły posiedzenia, takie jak data, opis oraz agenda.
System zapisuje dane w bazie danych i aktualizuje listę posiedzeń.

Przypadek użycia: PU2 – Automatyzacja prezentacji

Aktor: Administrator

Opis: Proces automatycznego generowania prezentacji na podstawie danych zapisanych w systemie dotyczących posiedzeń i ich agendy.

Warunki wstępne: Administrator jest zalogowany do systemu oraz w systemie istnieją dane dotyczące posiedzenia.

Przebieg:
Administrator wybiera posiedzenie z listy dostępnych spotkań.
Administrator uruchamia funkcję generowania prezentacji.
System pobiera dane dotyczące punktów spotkania oraz wydarzeń powiązanych z posiedzeniem.
System generuje prezentację na podstawie zapisanych informacji.
Administrator może pobrać lub wyświetlić wygenerowaną prezentację.

Przypadek użycia: PU3 – Zarządzanie wydarzeniami

Aktor: Administrator

Opis: Proces dodawania, edycji oraz usuwania wydarzeń w systemie.

Warunki wstępne: Administrator jest zalogowany do systemu.

Przebieg:
Administrator przechodzi do sekcji zarządzania wydarzeniami.
System wyświetla listę wydarzeń zapisanych w bazie danych.
Administrator może dodać nowe wydarzenie, edytować istniejące lub je usunąć.
Administrator wprowadza szczegóły wydarzenia, takie jak nazwa, data oraz opis.
System zapisuje zmiany w bazie danych.

Przypadek użycia: PU4 – Autoryzacja i uwierzytelnianie administratora

Aktor: Administrator

Opis: Proces logowania administratora do systemu w celu uzyskania dostępu do funkcji zarządzania.

Warunki wstępne: Administrator posiada aktywne konto w systemie.

Przebieg:
Administrator otwiera stronę logowania systemu.
Administrator wprowadza login oraz hasło.
System weryfikuje dane w bazie danych.
W przypadku poprawnych danych system przyznaje dostęp do panelu administratora.
Administrator może korzystać z funkcji zarządzania systemem.

Przypadek użycia: PU5 – Autoryzacja i uwierzytelnianie studenta

Aktor: Student

Opis: Proces logowania studenta do systemu w celu uzyskania dostępu do dostępnych funkcji użytkownika.

Warunki wstępne: Student posiada konto w systemie.

Przebieg:
Student otwiera stronę logowania.
Student wprowadza dane logowania.
System weryfikuje dane w bazie danych.
Po poprawnej weryfikacji student uzyskuje dostęp do systemu.

Przypadek użycia: PU6 – Przeglądanie listy wydarzeń

Aktor: Student

Opis: Proces przeglądania wydarzeń dostępnych w systemie.

Warunki wstępne: Student jest zalogowany do systemu.

Przebieg:
Student przechodzi do sekcji wydarzeń.
System pobiera dane o wydarzeniach z bazy danych.
System wyświetla listę wydarzeń wraz z podstawowymi informacjami.
Student może przeglądać szczegóły wybranego wydarzenia.

Przypadek użycia: PU7 – Przeglądanie harmonogramu posiedzeń

Aktor: Student

Opis: Proces przeglądania harmonogramu zaplanowanych posiedzeń.

Warunki wstępne: Student jest zalogowany do systemu.

Przebieg:
Student przechodzi do sekcji harmonogramu posiedzeń.
System pobiera dane o zaplanowanych posiedzeniach z bazy danych.
System wyświetla harmonogram wraz z datami i podstawowymi informacjami o spotkaniach.

Przypadek użycia: PU8 – Wyświetlanie planu spotkania

Aktor: Student

Opis: Proces wyświetlania szczegółowego planu danego spotkania.

Warunki wstępne: Student jest zalogowany do systemu oraz w systemie istnieje zapisane spotkanie.

Przebieg:
Student wybiera spotkanie z listy dostępnych posiedzeń.
System pobiera szczegóły spotkania z bazy danych.
System wyświetla plan spotkania wraz z listą punktów porządku obrad.
Student może zapoznać się z pełnym przebiegiem spotkania.

5.6.1.Implementacja modułu uwierzytelniania

```php
use App\Http\Controllers\Controller;
use App\Http\Requests\Auth\LoginRequest;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\View\View;

class AuthenticatedSessionController extends Controller
{
    public function create(): View
    {
        return view('auth.login');
    }

    public function store(Request $request): RedirectResponse
    {
        $credentials = $request->validate([
            'email' => ['required', 'email'],
            'password' => ['required'],
        ]);

        if (Auth::attempt($credentials, $request->boolean('remember'))) {

            $request->session()->regenerate();

            return redirect()->intended('/');
        }

        return back()->withErrors([
            'email' => 'Podane dane logowania są nieprawidłowe.',
        ])->onlyInput('email');
    }

    public function destroy(Request $request): RedirectResponse
    {
        Auth::logout();

        $request->session()->invalidate();

        $request->session()->regenerateToken();

        return redirect('/');
    }
}
```

Rys. 5.7. Kod odpowiedzialny za autoryzacje

Kluczowym elementem bezpieczeństwa aplikacji E-PSPK jest system uwierzytelniania, który został zaimplementowany w oparciu o kontroler AuthenticatedSessionController. Wykorzystuje on wbudowane mechanizmy frameworka Laravel, zapewniając bezpieczną obsługę sesji oraz ochronę przed popularnymi atakami, takimi jak Session Fixation.
Opis techniczny metod kontrolera:
Metoda create(): Odpowiada za wyświetlenie widoku formularza logowania (auth.login). Jest to punkt wyjściowy dla przypadków użycia PU4 oraz PU5.

- Metoda store(): Stanowi serce procesu logowania. Realizuje ona następujące kroki:

1. Walidacja danych: System sprawdza, czy wprowadzone dane są poprawne pod kątem formatu (wymagany e-mail oraz hasło).
2. Próba uwierzytelnienia (Auth::attempt): Wykorzystywana jest metoda bezpiecznego porównania zahaszowanego hasła w bazie danych z danymi wprowadzonymi przez użytkownika.
3. Zarządzanie sesją: Po poprawnym zalogowaniu następuje regeneracja identyfikatora sesji (session()->regenerate()), co jest kluczowe dla ochrony przed przejęciem sesji.
4. Obsługa błędów: W przypadku błędnych danych, system zwraca użytkownika do formularza z odpowiednim komunikatem, dbając o bezpieczeństwo poprzez nie podawanie szczegółów (np. czy to e-mail, czy hasło było błędne).

- Metoda destroy(): Realizuje proces bezpiecznego wylogowania użytkownika poprzez unieważnienie sesji (invalidate) oraz wygenerowanie nowego tokena CSRF, co zapobiega atakom typu Cross-Site Request Forgery.

5.6.2. Modelowanie danych – Struktura posiedzeń (Meeting)

Model Meeting stanowi fundament obsługi procesów decyzyjnych w aplikacji E-PSPK. Wykorzystuje on mechanizmy mapowania obiektowo-relacyjnego (ORM Eloquent), które pozwalają na bezpieczną i wydajną komunikację z bazą danych bez konieczności pisania złożonych zapytań SQL.

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Meeting extends Model
{
    protected $guarded = ['id'];

    protected $fillable = [
    'name', 'location', 'date', 'agenda', 'slug', 'current_agenda_item'
];

    protected $casts = [
        'agenda' => 'array',
        'date' => 'date',
    ];

    public function getRouteKeyName()
    {
        return 'slug';
    }

    public function harmonograms()
    {
        return $this->hasMany(Harmonogram::class)->orderBy('start_time', 'asc');
    }
}
```

Rys. 5.8. Kod odpowiedzialny za zarządzanie i przeglądanie (CRUD)

Analiza techniczna implementacji modelu:

- Masowe przypisywanie danych ($fillable / $guarded): Zastosowanie tablicy $fillable jest kluczowe dla bezpieczeństwa (Mass Assignment Protection). Definiuje ona dokładnie, które pola (nazwa, lokalizacja, data, agenda, slug) mogą być zapisane w bazie, co zapobiega nieautoryzowanym próbom modyfikacji wrażliwych danych przez użytkownika.
- Rzutowanie typów ($casts):
  - Pole agenda jest rzutowane na typ array. Pozwala to na przechowywanie złożonej struktury punktów obrad w formacie JSON w bazie danych, podczas gdy w kodzie PHP jest ona traktowana jako łatwa w obsłudze tablica. Ma to kluczowe znaczenie dla PU2 (Automatyzacja prezentacji), gdzie agenda musi być iterowana punkt po punkcie.
  - Pole date jest rzutowane na obiekt daty, co umożliwia łatwe sortowanie i formatowanie harmonogramu (PU7).
- Przyjazne adresy URL (getRouteKeyName): Przeciążenie tej metody i zwrócenie pola slug zamiast technicznego identyfikatora id pozwala na generowanie czytelnych dla człowieka adresów URL (np. /posiedzenie/walne-zgromadzenie-2026). Wpływa to pozytywnie na dostępność systemu (SEO i UX).
- Relacje bazodanowe (harmonograms): Model definiuje relację jeden-do-wielu (hasMany) z modelem Harmonogram. Dzięki temu system automatycznie pobiera wszystkie punkty czasowe przypisane do danego posiedzenia, sortując je rosnąco według czasu rozpoczęcia (orderBy).

5.6.3. Automatyzacja prezentacji

Moduł automatyzacji prezentacji jest kluczowym elementem systemu E-PSPK, mającym na celu usprawnienie przebiegu posiedzeń organów samorządowych (PU1, PU2). Implementacja opiera się na dynamicznym przetwarzaniu struktur danych typu JSON oraz asynchronicznej komunikacji między serwerem a interfejsem prezentacyjnym.

```php
<?php

namespace App\Http\Controllers;

use App\Models\Meeting;
use Illuminate\Http\Request;

class MeetingController extends Controller
{

    public function index()
    {
        $meetings = Meeting::orderBy('date', 'asc')->get();
        return view('meetings.index', compact('meetings'));
    }

    public function show(Meeting $meeting)
    {

        return view('meetings.show', compact('meeting'));
    }


    public function updateAgendaItem(Request $request, Meeting $meeting, $itemIndex)
    {

        $data = $request->validate([
            'description' => 'nullable|string',
            'attachment_name' => 'nullable|string|required_with:attachment_url',
            'attachment_url' => 'nullable|url|required_with:attachment_name',
        ]);

        $agenda = $meeting->agenda;

        if (!isset($agenda[$itemIndex])) {
            return back()->with('error', 'Wybrany punkt agendy nie istnieje.');
        }

        if ($request->has('description')) {
            $agenda[$itemIndex]['description'] = $data['description'];
        }

        if (!empty($data['attachment_url']) && !empty($data['attachment_name'])) {

            if (!isset($agenda[$itemIndex]['attachments'])) {
                $agenda[$itemIndex]['attachments'] = [];
            }

            $agenda[$itemIndex]['attachments'][] = [
                'name' => $data['attachment_name'],
                'url' => $data['attachment_url']
            ];
        }

        $meeting->agenda = $agenda;
        $meeting->save();

        return back()->with('success', 'Zaktualizowano punkt agendy.');
    }

    public function presentation(Meeting $meeting)
    {
        return view('meetings.presentation', compact('meeting'));
    }

    public function getStatus(Meeting $meeting)
    {
        return response()->json([
            'id' => $meeting->id,
            'current_agenda_item' => $meeting->current_agenda_item,

        ]);
    }
}
```

Rys. 5.9. Kod odpowiedzialny za automatyzację prezentacji

Analiza techniczna mechanizmów kontrolera:

- Zarządzanie treścią agendy (updateAgendaItem): Metoda ta umożliwia administratorowi dynamiczne wzbogacanie punktów obrad o dodatkowe opisy oraz załączniki (URL).
  - Walidacja warunkowa: Wykorzystano regułę required_with, która wymusza podanie nazwy załącznika tylko wtedy, gdy podano adres URL.
  - Przetwarzanie struktur zagnieżdżonych: Ponieważ pole agenda jest rzutowane w modelu na tablicę (Array), kontroler pobiera całą strukturę, modyfikuje konkretny indeks ($itemIndex) i zapisuje ją z powrotem. Pozwala to na elastyczne zarządzanie multimediami przypisanymi do konkretnych punktów prezentacji.
- Mechanizm wyświetlania prezentacji (presentation): Metoda zwraca dedykowany widok meetings.presentation. W przeciwieństwie do standardowych widoków, jest on zoptymalizowany pod ekrany wielkoformatowe (tryb pełnoekranowy) i służy jako interfejs wyjściowy podczas rzutowania obrazu na salę obrad.
- Synchronizacja stanu w czasie rzeczywistym (getStatus): Jest to kluczowa metoda z punktu widoku automatyzacji. Zwraca ona aktualny stan posiedzenia w formacie JSON:
  - Umożliwia to warstwie frontendowej (JavaScript) cykliczne odpytywanie serwera o wartość current_agenda_item.
  - Dzięki temu, gdy administrator zmieni punkt obrad w panelu sterowania, prezentacja wyświetlana na rzutniku automatycznie przełączy slajd na odpowiedni punkt agendy bez konieczności odświeżania strony przez operatora.

6. Podsumowanie

Celem niniejszej pracy było zaprojektowanie i zaimplementowanie nowoczesnej aplikacji webowej wspierającej działalność samorządów studenckich poprzez integrację narzędzi do zarządzania wydarzeniami, obsługi posiedzeń oraz wizualizacji danych geograficznych. Realizacja projektu pozwoliła na stworzenie spójnego środowiska informatycznego, które wypełnia lukę między sformalizowanymi systemami obiegu dokumentów a otwartymi platformami eventowymi.

W toku prac wykorzystano zaawansowany stos technologiczny, który zagwarantował wysoką wydajność i skalowalność systemu. Zastosowanie architektury **MVC** pozwoliło na czytelne odseparowanie logiki biznesowej od warstwy prezentacji, co w połączeniu z frameworkiem **Bootstrap** zaowocowało stworzeniem responsywnego i intuicyjnego interfejsu użytkownika. Kluczowym osiągnięciem było wdrożenie standardu **Progressive Web App** , dzięki czemu aplikacja łączy dostępność stron internetowych z funkcjonalnością rozwiązań natywnych, oferując płynne działanie na urządzeniach mobilnych oraz możliwość instalacji na ekranie głównym urządzenia.

Istotnym elementem systemu jest moduł interaktywnej mapy oparty na **OpenStreetMap API**, który w czasie rzeczywistym wizualizuje lokalizację parlamentów studenckich, ułatwiając użytkownikom dostęp do danych kontaktowych i profili uczelni. Zaimplementowany panel administracyjny dostarczył natomiast narzędzi do precyzyjnego zarządzania strukturą posiedzeń i składem prezydium, co w połączeniu z mechanizmem **Database Seeders** umożliwiło sprawne zarządzanie bazą danych oraz szybkie wdrażanie systemu w środowisku produkcyjnym przy użyciu technologii **Docker**.

Podsumowując, wytworzone oprogramowanie stanowi kompletne narzędzie wspomagające cyfryzację procesów wewnątrz organizacji studenckich. Automatyzacja generowania prezentacji na podstawie agendy spotkań oraz scentralizowany katalog samorządów znacząco redukują czas potrzebny na prace administracyjne, pozwalając członkom organizacji skupić się na działaniach merytorycznych. Projekt może stanowić fundament pod dalszy rozwój, obejmujący m.in. systemy elektronicznego głosowania czy zaawansowane moduły analityczne oparte na zgromadzonych danych przestrzennych.

7. Dalsze kierunki rozwoju

Zaprojektowana aplikacja stanowi fundament, który dzięki modułowej architekturze oraz wykorzystaniu nowoczesnych frameworków, pozwala na szeroki zakres przyszłej rozbudowy. Analiza potrzeb użytkowników oraz dynamicznie zmieniające się standardy webowe pozwalają wyznaczyć trzy główne ścieżki dalszego rozwoju systemu.

7.1. System elektronicznego głosowania i kworum

Najistotniejszym kierunkiem rozwoju pod kątem funkcjonalnym jest wdrożenie modułu do przeprowadzania głosowań w czasie rzeczywistym podczas posiedzeń. Wykorzystanie technologii **WebSockets** pozwoliłoby na natychmiastową komunikację między administratorem a uczestnikami, umożliwiając sprawdzanie listy obecności (kworum) oraz oddawanie głosów w trybie jawnym lub tajnym. Wyniki takich głosowań mogłyby być automatycznie agregowane i dołączane do generowanego protokołu z obrad, co jeszcze bardziej usprawniłoby procesy decyzyjne w samorządzie.

7.2. Rozszerzenie automatyzacji i integracja z zewnętrznymi API

Kolejnym etapem rozwoju jest pogłębienie funkcjonalności **automatyzacji prezentacji** oraz dokumentów. System mógłby zostać wzbogacony o moduł automatycznego generowania plików PDF z gotowymi uchwałami i pismami urzędowymi na podstawie danych wprowadzonych w agendzie posiedzenia. Ponadto, planowana jest integracja z zewnętrznymi systemami uczelnianymi oraz kalendarzami (np. Google Calendar lub Outlook), co pozwoliłoby na automatyczną synchronizację terminów zjazdów i wydarzeń z prywatnymi harmonogramami członków prezydium.

7.3. Zaawansowana analiza danych i powiadomienia Push

W sferze technologicznej rozwój aplikacji powinien zmierzać w stronę pełniejszego wykorzystania możliwości **PWA**. Wprowadzenie natywnych powiadomień Push pozwoliłoby na informowanie użytkowników o zmianach w porządku obrad lub publikacji nowych wyników zjazdów bezpośrednio na ekranach ich urządzeń mobilnych. Dodatkowo, gromadzone dane statystyczne dotyczące aktywności samorządów mogłyby posłużyć do stworzenia modułu analitycznego, który za pomocą zaawansowanych wykresów wizualizowałby efektywność pracy poszczególnych jednostek na przestrzeni kadencji.

7.4. Internacjonalizacja i moduł wielojęzyczności

Ze względu na rosnącą liczbę studentów zagranicznych biorących udział w pracach organów studenckich, istotnym kierunkiem jest pełna internacjonalizacja interfejsu (i18n). Wdrożenie wielojęzyczności pozwoliłoby na korzystanie z katalogu oraz śledzenie obrad osobom anglojęzycznym, co znacząco podniosłoby inkluzywność systemu i umożliwiło jego promocję w ramach międzynarodowych struktur akademickich.

Bibliografia

[1] Bootstrap Documentation, Introduction to Bootstrap, [Online] Dostępne na: https://getbootstrap.com/docs/ [Dostęp: 14 marca 2026].
[2] Docker Documentation, Docker Overview and Architecture, [Online] Dostępne na: https://docs.docker.com/get-started/overview/ [Dostęp: 14 marca 2026].
[3] Google Developers, Progressive Web Apps (PWA) Overview, [Online] Dostępne na: https://web.dev/progressive-web-apps/ [Dostęp: 14 marca 2026].
[4] Laravel Documentation, The MVC Architecture in Laravel, [Online] Dostępne na: https://laravel.com/docs/ [Dostęp: 14 marca 2026].
[5] Laravel Documentation, Database: Seeding, [Online] Dostępne na: https://laravel.com/docs/migrations#database-seeding [Dostęp: 14 marca 2026].
[6] MDN Web Docs, Service Worker API, [Online] Dostępne na: https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API [Dostęp: 14 marca 2026].
[7] OpenStreetMap Wiki, API v0.6 – Main Documentation, [Online] Dostępne na: https://wiki.openstreetmap.org/wiki/API_v0.6 [Dostęp: 14 marca 2026].
[8] phpMyAdmin Documentation, Official phpMyAdmin Documentation, [Online] Dostępne na: https://docs.phpmyadmin.net/ [Dostęp: 14 marca 2026].
[9] Stauffer M., Laravel: Up & Running: A Framework for Building Modern PHP Applications, O'Reilly Media, 2019.
[10] W3C Working Group, Web Application Manifest, [Online] Dostępne na: https://www.w3.org/TR/appmanifest/ [Dostęp:14 marca 2026].
[11] Kamiński P.: Laravel. Wstęp do programowania aplikacji internetowych, Wydawnictwo Helion, Wydawnictwo Helion, 2019.

Spis skrótów

API – Application Programming Interface
CRUD – Create, Read, Update, Delete
CSS – Cascading Style Sheets
CSRF – Cross-Site Request Forgery
GUI – Graphical User Interface
HTML – HyperText Markup Language
HTTPS – Hypertext Transfer Protocol Secure
JSON – JavaScript Object Notation
MVC – Model-View-Controller
ORM – Object-Relational Mapping
OSM – OpenStreetMap
PHP – Hypertext Preprocessor
POI – Point of Interest
PWA – Progressive Web App
SaaS – Software as a Service
SQL – Structured Query Language
URL – Uniform Resource Locator

Spis rysunków

Rys. 1.1. Logo Parlamentu Studentów Rzeczypospolitej Polskiej 6
Rys. 1.2. Logo Forum Uczelni Technicznych 7
Rys. 1.3. Logo Parlamentu Studentów Politechniki Koszalińskiej 8
Rys. 2.1. Logo Visual Studio Code 11
Rys. 2.2. Logo JetBrains 12
Rys. 2.3. Logo git i GitHub 13
Rys. 2.4. Logo PHP 14
Rys. 2.5. Logo Laravel 16
Rys. 2.6. Logo Bootstrap 17
Rys. 2.7. Logo Docker 19
Rys. 2.8. Logo PWA 20
Rys. 2.9. Logo phpMyAdmin 21
Rys. 3.1. Widok ekranu posiedzeń w systemie eSesja 22
Rys. 3.2. Logo systemu eSesja 23
Rys. 3.3 Widok aplikacji Meeting Application 23
Rys. 3.4. Logo Meeting Application 24
Rys. 4.1. Widok Start 25
Rys. 4.2. Widok samorządu 26
Rys. 4.3. Widok ekranu z członkami prezydium i komisji rewizyjnej FUT. 27
Rys. 4.4. Widok ekranu z członkiem FUT. 27
Rys. 4.5. Widok wydarzeń 28
Rys. 4.6. Widok wydarzenia 28
Rys. 4.7. Widok interaktywnej mapy oraz znacznika z informacjami o uczelni. 29
Rys. 4.8. Widok ekranu logowania. 30
Rys. 4.9. Widok odzyskiwania hasła 31
Rys. 4.10. Widok ekranu rejestracji. 31
Rys. 4.11. Widok panelu administratora 32
Rys. 4.12. Widok zarządzanie posiedzeniami 32
Rys. 4.13. Edycja informacji o wydarzeniu: Organizatorzy, Edycja, Harmonogram 33
Rys. 4.14. Lista zjazdów 33
Rys. 4.15. Widok katalogu samorządów studenckich w panelu administratora 34
Rys. 4.16. Widok edycji informacji o samorządzie - Informacje podstawowe i Kontakt 35
Rys. 4.17. Widok edycji informacji o samorządzie - Dodatkowe 35
Rys. 5.1. Struktura katalogów projektu 37
Rys. 5.2. Ikona aplikacji w systemie macOS 39
Rys. 5.3. Konfiguracja pliku manifest.json dla aplikacji PWA 39
Rys. 5.4. Widok tabeli fut_events w phpMyAdmin 40
Rys. 5.5. Tabele w bazie danych 41
Rys. 5.6. Diagram przypadków użycia 42
Rys. 5.7. Kod odpowiedzialny za autoryzacje 45
Rys. 5.8. Kod odpowiedzialny za zarządzanie i przeglądanie (CRUD) 46
Rys. 5.9. Kod odpowiedzialny za automatyzację prezentacji 48