AKADEMIA NAUK STOSOWANYCH
W NOWYM SĄCZU

Wydział Nauk Inżynieryjnych
Katedra Informatyki

DOKUMENTACJA PROJEKTOWA

PROGRAMOWANIE URZĄDZEŃ MOBILNYCH

Wirtualny Dziekanat

Autor:

Marcin Dudek
Mateusz Basiaga

Prowadzący:
mgr inż. Dawid Kotlarski

Nowy Sącz 2025

Spis treści

1. Ogólne określenie wymagań

1.1. Główne funkcje aplikacji

- **Logowanie i autoryzacja:**
  Użytkownicy powinni móc logować się za pomocą uczelnianego adresu e-mail oraz hasła. Po piewrszym logowaniu użytkownik dostanie opcje logowania za się pomocą odcisku palaca albo skanu twarzy. Możliwość logowania tyczy się zarówno dla studentów, jak i pracowników (np. wykładowcy, administracja).

- **Podgląd ocen i zaliczeń:**
  Studenci powinni mieć możliwość przeglądania swoich ocen z egzaminów, kolokwiów oraz innych zaliczeń. Możliwość filtrowania wyników na podstawie przedmiotu, semestru czy wykładowcy.

- **Plan zajęć:**
  Podgląd bieżącego planu zajęć z opcją aktualizacji na żywo (jeśli np. zajęcia zostaną odwołane czy przeniesione).

- **Harmonogram egzaminów i sesji:**
  Informacje o nadchodzących egzaminach, sesjach poprawkowych i innych ważnych wydarzeniach związanych z uczelnią. Możliwość zapisywania się na egzaminy, jeżeli to wymagane.

- **Powiadomienia:**
  Push notifications o nowych ocenach, nadchodzących zajęciach, zmianach w harmonogramie lub ważnych ogłoszeniach.

- **Informacje ogólne:**
  Tablica ogłoszeń z najważniejszymi informacjami od uczelni, np. nowe zarządzenia rektora, wydarzenia na kampusie itp.

- **Profile użytkowników:**
  Każdy użytkownik powinien mieć profil z podstawowymi danymi (imię, nazwisko, nr indeksu, rocznik, itd.). Możliwość aktualizacji niektórych danych kontaktowych.

- **Responsywność i UX:**
  Chcemy, żeby aplikacja była prosta i szybka w obsłudze.

- **Bezpieczeństwo danych:**
  Chcemy, żeby dane były dobrze chronione, bo będą tu przechowywane prywatne informacje studentów. Może jakieś szyfrowanie?

- **Offline mode:**
  Dobrze by było, gdyby część funkcji działała offline (np. podgląd planu zajęć lub ocen).

2. Określenie wymagań szczegółowych

2.1. Cel aplikacji

Celem aplikacji jest ułatwienie studentom oraz pracownikom uczelni dostępu do kluczowych funkcji administracyjnych i informacyjnych związanych z edukacją. Aplikacja ma zastąpić tradycyjne interakcje z dziekanatem, umożliwiając szybki dostęp do ocen, planu zajęć, harmonogramu egzaminów, czy otrzymywania powiadomień.

- **Zakres aplikacji:**
  Studenci: dostęp do ocen, planu zajęć, harmonogramu egzaminów, powiadomień.
  Administracja: zarządzanie danymi, kontakt ze studentami.
- **Platformy:**
  Systemy operacyjne:
  Android,
  iOS.
- **Dane wejściowe: (Logowanie użytkowników)**
  Dane studentów: nr indeksu, rocznik, oceny, plan zajęć, egzaminów, zgłoszenia do dziekanatu.
  Zdarzenia dziekanatu: zmiany w planie, ogłoszenia, nowe dokumenty, powiadomienia o ocenach.
- **Opis interfejsu użytkownika i elementów interaktywnych:**

Ekran logowania: Rys. 2.1 (s. 9)

Rys. 2.1. Ekran logowania

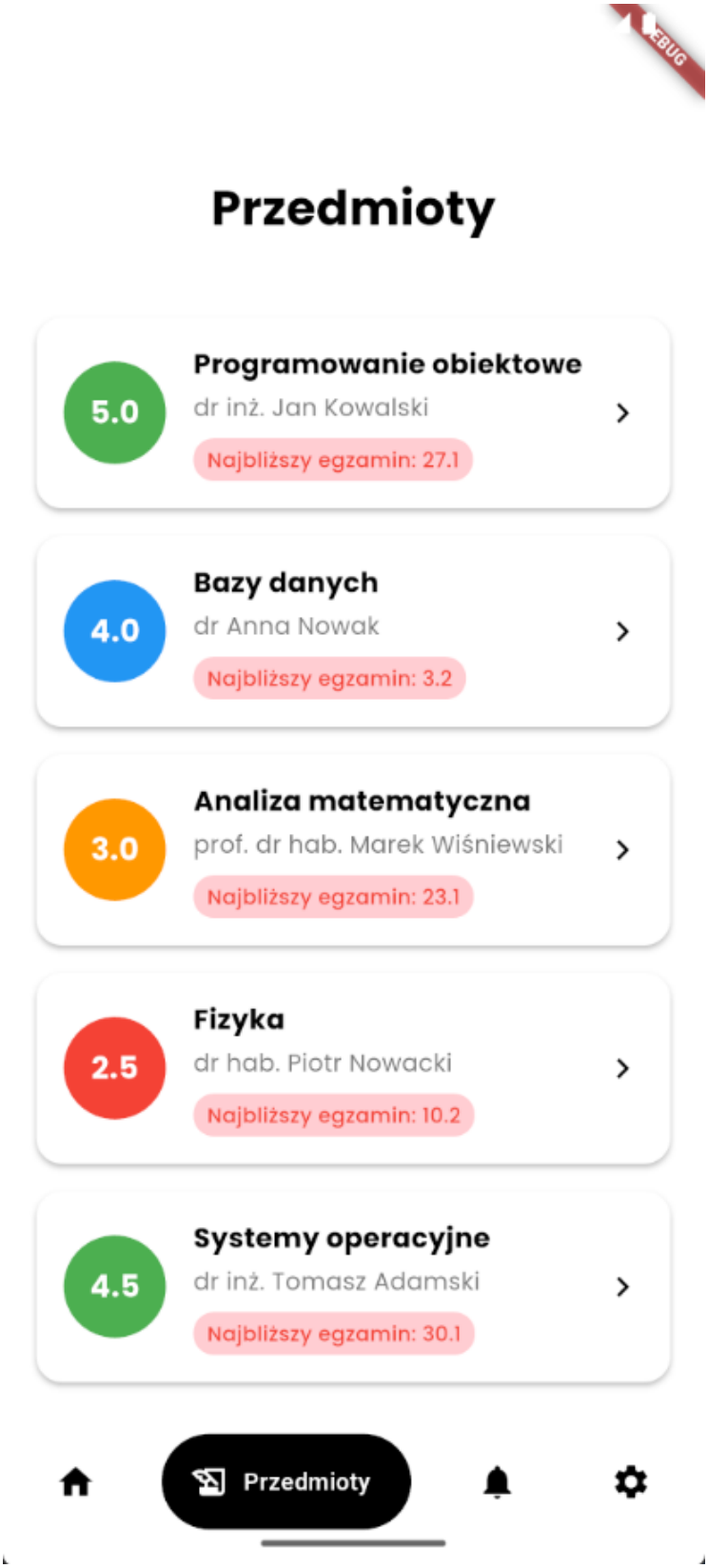
Pola tekstowe: „Email” oraz „Hasło”.
Przyciski: „Zaloguj” – po kliknięciu, autoryzacja danych w tle i przejście do ekranu głównego. W przypadku błędnych danych, komunikat „Niepoprawne dane logowania”. „Zapomniałem hasła” – przekierowanie do formularza resetowania hasła.

- **Ekran główny (Dashboard): Rys. 2.2 (s. 10)**

Rys. 2.2. Ekran główny

Wyświetlane informacje: skrót do ocen, nadchodzących zajęć, powiadomienia o ważnych wydarzeniach.
Przyciski:
„Przedmioty” – po kliknięciu, przejście do ekranu z listą ocen, harmonogramu egzaminów z możliwością filtrowania według przedmiotu.
„Plan zajęć” – po kliknięciu, podgląd planu zajęć (interaktywny kalendarz).
„Ustawienia” – po kliknięciu, przejście do ustawień naszego profilu
Automatyczne zdarzenia: wyświetlanie powiadomień push o nadchodzących zajęciach, ocenach, ważnych wydarzeniach.

- **Ekran ocen: Rys. 2.3 (s. 11)**



Rys. 2.3. Ekran ocen

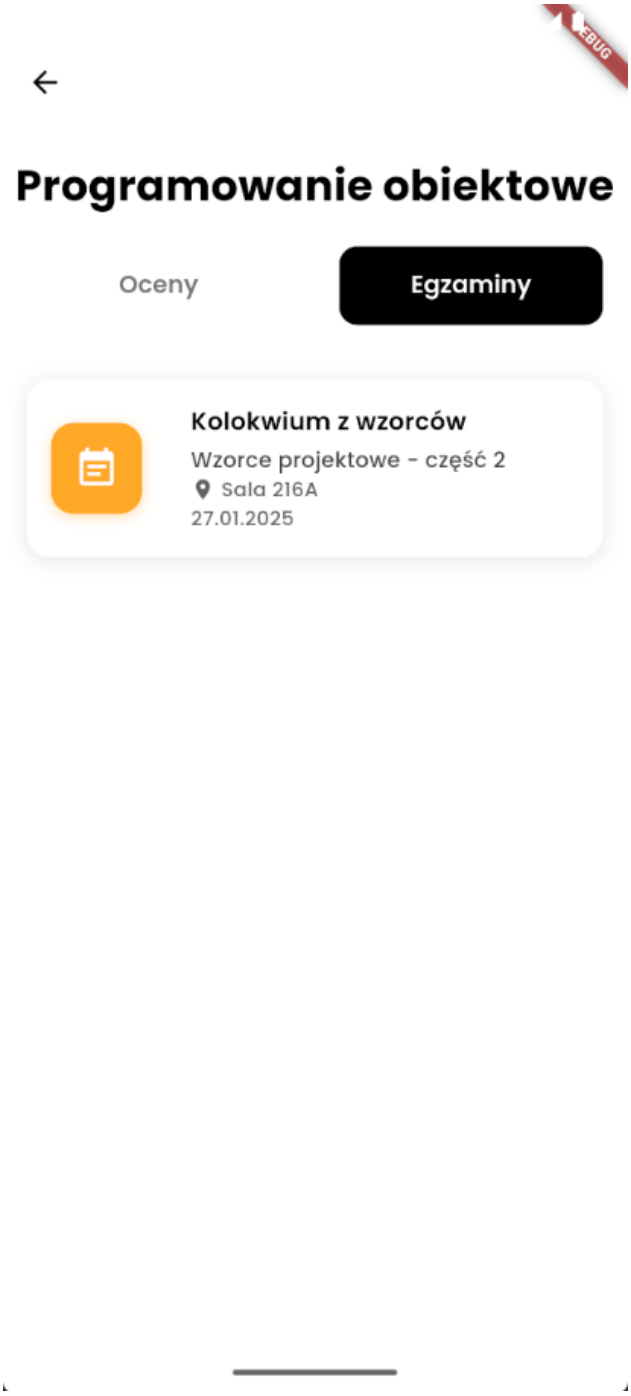
Tabela ocen: kolumny „Przedmiot”, „Ocena”, „Komentarz wykładowcy”, „Data”.
Opcje sortowania: sortowanie ocen po przedmiocie, dacie.
Zachowanie: po kliknięciu w przedmiot, otwarcie szczegółów przedmiotu (np. opis, prowadzący, historia ocen). Plan zajęć:

- **Widok kalendarza**: wyświetlanie zajęć na dany tydzień/dzień.
  Funkcje interaktywne Rys. 2.4 (s. 12):

Rys. 2.4. Ekran Planu zajęć

Zmiana tygodnia za pomocą przesuwania palcem (swipe left/right).
Kliknięcie na zajęcia otwiera szczegóły, np. nazwisko wykładowcy, sala, godziny.
Powiadomienia push: automatyczne przypomnienia o nadchodzących zajęciach z możliwością wyłączenia.

- **Harmonogram egzaminów Rys. 2.5 (s. 13)**:



Rys. 2.5. Ekran Egzaminów

Lista egzaminów: możliwość filtrowania według przedmiotu, prowadzącego, daty. **Powiadomienia push**: przypomnienia o zbliżających się egzaminach. Komunikacja z dziekanatem:

- **Zdarzenia automatyczne**:
  Aplikacja będzie automatycznie wysyłać powiadomienia push o zmianach w planie zajęć, wynikach egzaminów, nowo dodanych dokumentach i ogłoszeniach.
  Automatyczne aktualizacje planu zajęć oraz harmonogramu egzaminów po synchronizacji z serwerem (np. co 30 minut).

- **Działanie offline**:
  Gdy brak połączenia z internetem, użytkownik ma dostęp do zapisanych wcześniej danych (plan zajęć, oceny).

- **Zachowanie aplikacji w niepożądanych sytuacjach**:
  Brak połączenia z internetem:
  Aplikacja wyświetla komunikat „Brak połączenia. Sprawdź połączenie z internetem" przy próbie wykonania operacji wymagającej synchronizacji z serwerem (np. wysłanie wiadomości do dziekanatu). W trybie offline: dostęp do zapisanych danych, brak możliwości interakcji z funkcjami wymagającymi połączenia.
  Błędne dane logowania:
  Komunikat „Niepoprawny email lub hasło" oraz możliwość ponownego wpisania danych. Pole tekstowe zostaje podświetlone na czerwono.
  Serwer niedostępny:
  Wyświetlenie komunikatu „Serwer dziekanatu jest obecnie niedostępny. Spróbuj ponownie później".
  Niepoprawne działanie aplikacji:
  W przypadku błędu technicznego aplikacja wyświetli komunikat „Wystąpił błąd. Spróbuj ponownie" i zapisze logi błędów do późniejszej analizy przez programistów.

3. Projektowanie

3.1. Przygotowanie narzędzi

W ramach przygotowania środowiska do implementacji aplikacji wirtualnego dziekanatu oraz zarządzania wersjami kodu, wybrano zestaw narzędzi wspierających proces tworzenia oraz zapewniających automatyzację wielu czynności. W poniższych punktach opisano każde z wykorzystanych narzędzi wraz z ich rolą oraz załączonym linkiem do dokumentacji.

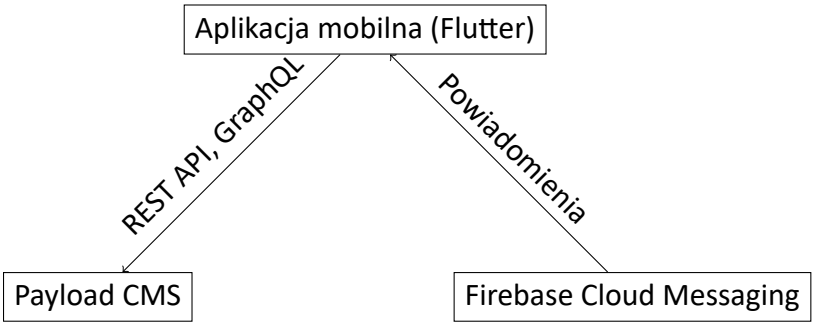
- **Git** – system kontroli wersji, umożliwiający śledzenie zmian w kodzie oraz współpracę w zespole. Dokumentacja narzędzia: https://git-scm.com/doc[1]
- **VSCode** – edytor kodu źródłowego, który zapewnia wsparcie dla wielu języków programowania i umożliwia instalację rozszerzeń wspierających programowanie. Dokumentacja narzędzia: https://code.visualstudio.com/docs[2]
- **Android Studio** – środowisko programistyczne wykorzystywane głównie do testowania aplikacji na emulatorze Android oraz zarządzania SDK. Dokumentacja: https://developer.android.com/studio/intro[3]
- **Flutter SDK** – zestaw narzędzi do tworzenia aplikacji wieloplatformowych. Dokumentacja: https://flutter.dev/docs[4]
- **Doxygen** – narzędzie do generowania dokumentacji automatycznej na podstawie komentarzy w kodzie źródłowym. Dokumentacja narzędzia: https://www.doxygen.nl/[5]
- **Doxygen Awesome** – motyw graficzny dostosowujący wygląd strony wygenerowanej przez Doxygen do współczesnych standardów. Więcej informacji: https://github.com/jothepro/doxygen-awesome-css[6]
- **Lefthook** – narzędzie do zarządzania hookami Git, które wspiera automatyczne formatowanie, walidację kodu oraz zgodność wiadomości commitów z konwencją. Dokumentacja: https://github.com/evilmartians/lefthook[7]
- **Commitlint** – narzędzie do sprawdzania zgodności wiadomości commitów z konwencją *Conventional Commits*. Dokumentacja: https://commitlint.js.org/[8]
- **GitHub Actions** – platforma do automatyzacji procesów CI/CD. Umożliwia automatyczną walidację commitów, generowanie dokumentacji oraz wersjonowanie wydań. Dokumentacja: https://docs.github.com/en/actions[9]

3.2. Architektura systemu

Aplikacja została zbudowana w oparciu o dwa główne komponenty:

- **Backend - Payload CMS** – główny system zarządzania treścią, odpowiedzialny za:
  - Autoryzację i uwierzytelnianie użytkowników
  - Przechowywanie i zarządzanie danymi aplikacji
  - API REST do komunikacji z aplikacją mobilną
  - Panel administracyjny dla pracowników uczelni
- **Firebase Cloud Messaging** – wykorzystywany wyłącznie do obsługi powiadomień push:
  - Wysyłanie powiadomień o zmianach w planie zajęć
  - Informowanie o nowych ogłoszeniach
  - Powiadomienia o nowych wiadomościach

System składa się z trzech głównych komponentów **Rys. 3.1**:



Rys. 3.1. Architektura systemu

Architektura aplikacji została zaprojektowana w modelu klient-serwer z następującymi komponentami:

- **Frontend (aplikacja mobilna)**:
  - Interfejs użytkownika w Material Design
  - Zarządzanie stanem aplikacji (Riverpod)
  - Przechowywanie danych lokalnych
  - Obsługa trybu offline
- **Backend (Payload CMS)**:
  - REST API
  - System autoryzacji
  - Baza danych
  - Panel administracyjny
  - GraphQL
- **Usługi zewnętrzne**:
  - Firebase Cloud Messaging (powiadomienia push)

3.3. Framework Flutter

Flutter to otwartoźródłowy framework opracowany przez firmę Google, służący do tworzenia wieloplatformowych aplikacji za pomocą jednej bazy kodu. Jego architektura opiera się na języku Dart, co pozwala na wydajną kompilację do kodu natywnego dla systemów Android, iOS, macOS, Windows, Linux, a także do aplikacji webowych. Główne zalety Fluttera to szybki rozwój interfejsu użytkownika, możliwość tworzenia aplikacji o wysokiej wydajności oraz wsparcie dla wzorców projektowych takich jak Material Design i Cupertino.

3.3.1. Możliwości języka Dart

Język Dart, na którym opiera się Flutter, jest wieloparadygmatowym językiem programowania opracowanym przez Google. Wśród jego cech charakterystycznych można wymienić:

- **Statyczne typowanie**: Pozwala na wykrywanie błędów już na etapie kompilacji.

- **Obsługa programowania obiektowego**: Dart wspiera klasy, dziedziczenie, interfejsy i mixiny, co pozwala na tworzenie złożonych struktur kodu.
- **Hot Reload**: Funkcja ta umożliwia szybkie wprowadzanie zmian w kodzie i ich natychmiastowe testowanie, co przyspiesza proces rozwoju aplikacji.
- **Wydajność**: Kod Dart jest kompilowany do kodu natywnego, co pozwala na uzyskanie wysokiej wydajności aplikacji.
- **Asynchroniczność**: Obsługa asynchronicznych strumieni i przyszłości (ang. Futures) umożliwia efektywną pracę z operacjami wejścia/wyjścia.

3.3.2. Material Design

Flutter zapewnia natywne wsparcie dla Material Design, języka projektowania stworzonego przez Google, który definiuje zasady tworzenia nowoczesnych, responsywnych i estetycznych interfejsów użytkownika. Flutter udostępnia szeroki zestaw gotowych komponentów, takich jak:

- `AppBar` - pasek aplikacji z tytułem i opcjonalnymi akcjami.
- `FloatingActionButton` - unoszący się przycisk akcji.
- `Drawer` - wysuwane menu nawigacyjne.
- `Card` - komponent do wyświetlania informacji w stylu kart.
- `TextField` - pole tekstowe do wprowadzania danych.

Dzięki temu deweloperzy mogą tworzyć spójne interfejsy użytkownika zgodne z najnowszymi trendami projektowymi.

3.3.3. Pakiety i narzędzie Pub

Pub to menedżer pakietów dla Fluttera i języka Dart, który umożliwia łatwe zarządzanie bibliotekami i zależnościami w projekcie. Za pomocą Pub można:

- Dodawać gotowe pakiety z `pub.dev`, takich jak `http` do pracy z protokołem HTTP czy `provider` do zarządzania stanem aplikacji.
- Tworzyć własne pakiety i udostępniać je społeczności programistów.
- Aktualizować zależności i zarządzać ich wersjami, co pozwala na utrzymanie zgodności między różnymi komponentami aplikacji.

Dzięki Pub, rozwój aplikacji w Flutterze staje się bardziej efektywny, a społeczność użytkowników ma dostęp do bogatego ekosystemu bibliotek i narzędzi.

3.4. Payload CMS

Payload CMS to nowoczesny system zarządzania treścią (CMS), który został zaprojektowany z myślą o deweloperach. Jest to w pełni konfigurowalny headless CMS oparty na Node.js, który umożliwia łatwe tworzenie i zarządzanie treściami w aplikacjach internetowych. Payload wyróżnia się prostotą integracji, elastycznością oraz wysoką wydajnością.

3.4.1. Funkcje i zalety Payload CMS

Payload CMS oferuje szereg zaawansowanych funkcji, które ułatwiają pracę deweloperom:

- **Headless CMS**: Payload działa jako API-first CMS, co oznacza, że treści można dostarczać do dowolnego frontendowego frameworka lub aplikacji.
- **Elastyczna konfiguracja**: System pozwala na definiowanie niestandardowych schematów treści (ang. schemas) przy użyciu kodu JavaScript lub TypeScript.
- **Bezpieczeństwo**: Payload oferuje wbudowane mechanizmy uwierzytelniania, zarządzania użytkownikami i ról, a także obsługę protokołu OAuth.
- **Interfejs użytkownika**: Payload udostępnia intuicyjny panel administracyjny, który jest generowany automatycznie na podstawie zdefiniowanych schematów.
- **API REST i GraphQL**: Payload obsługuje zarówno REST API, jak i GraphQL, co umożliwia łatwy dostęp do danych na różne sposoby.
- **Wtyczki i rozszerzalność**: Payload wspiera wtyczki, co pozwala na rozszerzanie jego funkcjonalności o dodatkowe moduły.

3.4.2. Proces tworzenia projektu

Tworzenie projektu w Payload CMS przebiega w kilku prostych krokach:

1. **Instalacja**: Payload można zainstalować za pomocą menedżera pakietów npm lub yarn: **Listing. 1 (s. 20)**:

```
npx create-payload-app my-project
```

Listing 1. Tworznie cms

2. **Definicja schematów**: Deweloperzy definiują kolekcje danych w formie kodu, np.: **Listing. 2 (s. 20)**:

```
const Posts = {
  slug: 'posts',
  fields: [
    {
      name: 'title',
      type: 'text',
    },
    {
      name: 'content',
      type: 'richText',
    },
  ],
};
```

Listing 2. Przykład deklaracji w Payload CMS

3. **Uruchomienie serwera**: Po skonfigurowaniu projektu, serwer Payload można uruchomić poleceniem `npm run dev`.

4. **Integracja z frontendem**: Payload CMS udostępnia API, które można wykorzystać w aplikacjach frontendowych, np. z wykorzystaniem React, Next.js czy Vue.js.

3.4.3. Przykładowe zastosowania Payload CMS

Payload CMS znajduje zastosowanie w wielu typach projektów, takich jak:

- Strony internetowe i blogi,
- Aplikacje mobilne i webowe wymagające dynamicznych treści,
- Platformy e-commerce,
- Systemy zarządzania użytkownikami.

Dzięki swojej elastyczności i nowoczesnemu podejściu Payload CMS stanowi doskonałe rozwiązanie dla deweloperów poszukujących prostego w użyciu, ale jednocześnie potężnego narzędzia do zarządzania treścią.

3.5. Projekt interfejsu użytkownika aplikacji

3.5.1. Style i motywy

- Spójny system projektowy Material Design 3
- Wsparcie dla motywu jasnego i ciemnego
- Dynamiczne dostosowanie do różnych rozmiarów ekranów
- Responsywne komponenty UI

3.5.2. Nawigacja

- Menu dolne z 4 głównymi sekcjami:
  - Panel główny
  - Powiadomienia
  - Wiadomości
  - Ustawienia
- Nawigacja stosowa między ekranami
- Gesty nawigacyjne (swipe)

3.6. Model danych

3.6.1. Główne encje

- **User**:
  - Dane podstawowe (imię, nazwisko, email)
  - Rola (student/wykładowca/admin)
  - Ustawienia (język, powiadomienia)
- **Schedule**:
  - Przedmiot
  - Data i czas
  - Sala
  - Prowadzący

- **Notification**:
  - Typ
  - Treść
  - Data
  - Odbiorcy

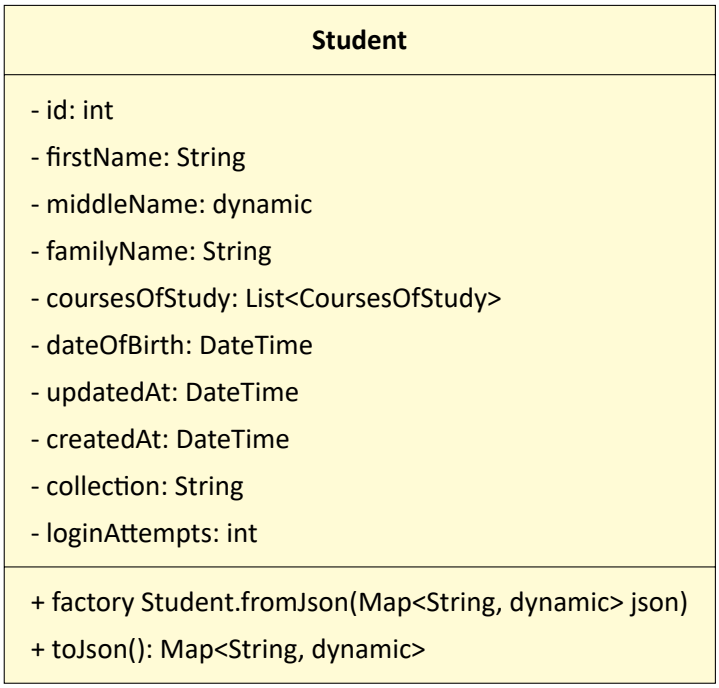
3.7. Bezpieczeństwo

- **Autoryzacja**:
  - JWT tokens
  - Refresh tokens
  - Szyfrowanie danych wrażliwych
- **Walidacja danych**:
  - Walidacja formularzy
  - Sanityzacja danych wejściowych
  - Obsługa błędów

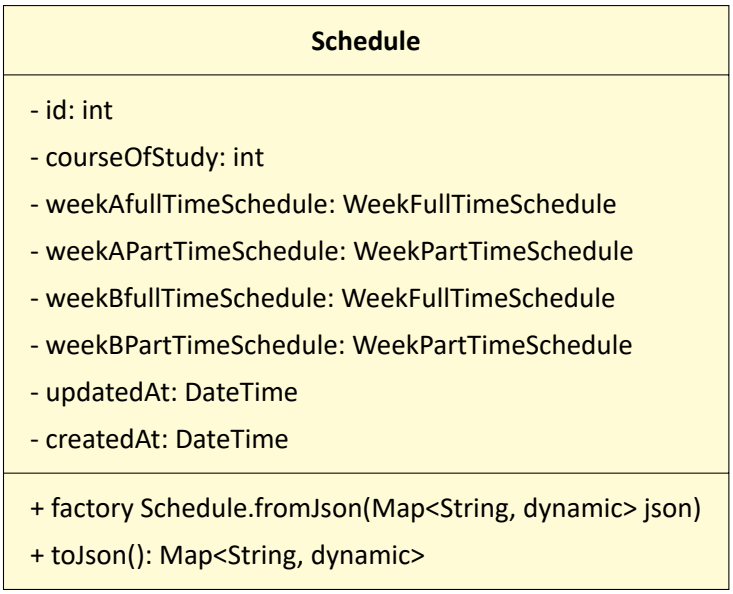
3.8. Synchronizacja danych

- **Cache lokalny**:
  - Przechowywanie planu zajęć
  - Buforowanie ogłoszeń
  - Ustawienia użytkownika
- **Strategia synchronizacji**:
  - Automatyczna synchronizacja w tle
  - Ręczne odświeżanie danych
  - Kolejkowanie operacji offline

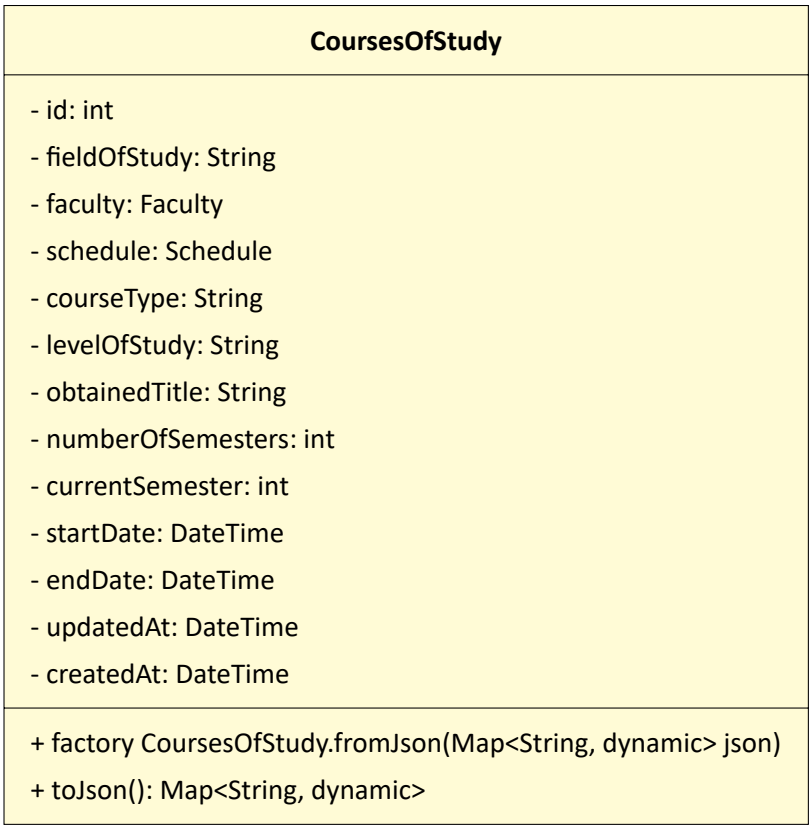
3.9. Diagram UML



Rys. 3.2. Klasa Student



Rys. 3.3. Klasa Schedule



Rys. 3.4. Klasa CoursesOfStudy

3.10. Design backendu

Modularna i skalowalna architektura backendu stanowi kluczowy element współczesnych systemów programistycznych. Projektowanie backendu oparte na najlepszych praktykach

pozwala na łatwe zarządzanie, rozwijanie oraz integrację z różnorodnymi usługami. Poniżej przedstawiono kluczowe aspekty, które powinny być uwzględnione przy projektowaniu nowoczesnego backendu.

3.10.1. Budowa modułowa

Budowa modułowa pozwala na podzielenie systemu na niezależne komponenty, które mogą być rozwijane i utrzymywane autonomicznie. Dzięki temu:

- Każdy moduł jest odpowiedzialny za określony zestaw funkcjonalności, co ułatwia ich implementację i testowanie.
- Możliwe jest równoległe rozwijanie różnych części systemu przez różne zespoły deweloperskie.
- Moduły można łatwo wymieniać lub rozszerzać bez wpływu na pozostałe części systemu.

3.10.2. Łatwa skalowalność

Aby backend był skalowalny, należy:

- Projektować usługi zgodnie z zasadami mikroserwisów lub serverless, co pozwala na niezależne skalowanie poszczególnych funkcjonalności.
- Korzystać z elastycznych baz danych, takich jak NoSQL, które łatwo adaptują się do wzrostu danych.
- Wykorzystać load balancery do równoważenia obciążenia między serwerami.
- Używać narzędzi monitorujących, takich jak Prometheus czy Grafana, aby efektywnie zarządzać wydajnością systemu.

3.10.3. Internacjonalizacja

Internacjonalizacja (i18n) jest kluczowa dla systemów działających na rynkach globalnych. Kluczowe kroki to:

- Używanie narzędzi takich jak gettext lub formatów JSON/YAML do przechowywania tłumaczeń.
- Projektowanie interfejsów API, które wspierają wielojęzyczność, np. poprzez parametry `locale`.

- Uwzględnianie lokalnych formatów dat, liczb i walut w zależności od ustawień użytkownika.
- Testowanie systemu pod kątem poprawności wyświetlania treści w różnych językach.

3.10.4. Bezpieczeństwo i zarządzanie danymi

Bezpieczeństwo danych i użytkowników to fundament każdego backendu. Zaleca się:

- Stosowanie szyfrowania danych w ruchu (TLS/SSL) oraz w spoczynku.
- Implementację autoryzacji i uwierzytelniania z wykorzystaniem standardów takich jak OAuth 2.0 czy JWT.
- Regularne testy bezpieczeństwa w celu wykrywania podatności.
- Spełnianie wymogów prawnych, takich jak RODO (GDPR) w przypadku zarządzania danymi osobowymi.

3.10.5. Monitorowanie i obsługa błędów

Monitorowanie systemu pozwala na szybkie wykrywanie i rozwiązywanie problemów. Warto:

- Korzystać z narzędzi takich jak Sentry do logowania błędów.
- Implementować mechanizmy automatycznego restartu usług w przypadku awarii.
- Tworzyć raporty wydajnościowe i systemy alertów w celu bieżącej kontroli stanu backendu.

Modularna budowa, łatwa skalowalność, internacjonalizacja, bezpieczeństwo oraz monitorowanie stanowią podstawę dobrze zaprojektowanego backendu, który może sprostać wymaganiom współczesnych aplikacji.

4. Implementacja

4.1. Struktura projektu

Projekt został podzielony na następujące główne katalogi:

- **lib/** - główny katalog z kodem źródłowym
  - **models/** - modele danych
  - **screens/** - widoki aplikacji
  - **utils/** - narzędzia pomocnicze
  - **providers/** - zarządzanie stanem
  - **services/** - warstwa komunikacji z API
  - **widgets/** - komponenty wielokrotnego użytku

4.2. Modele danych

4.2.1. Model użytkownika Listing. 3 (s. 27)

```
class AppUser {
  final String uid;
  final String email;

  AppUser({
    required this.uid,
    required this.email,
  });
}
```

Listing 3. Model użytkownika

Model ‘AppUser‘ reprezentuje użytkownika aplikacji. Zawiera unikalny identyfikator ‘uid‘ oraz adres email ‘email‘.

4.2.2. Model wydarzenia Listing. 4 (s. 28)

```dart
class Event {
  final String title;
  final TimeOfDay startTime;
  final TimeOfDay endTime;
  final String room;
  final String type;
  final String lecturer;
  final DateTime date;

  Event({
    required this.title,
    required this.startTime,
    required this.endTime,
    required this.room,
    required this.lecturer,
    required this.type,
    required this.date,
  });
}
```

Listing 4. Model wydarzenia

Model ‘Event‘ reprezentuje wydarzenie w aplikacji. Zawiera informacje takie jak tytuł ‘title‘, czas rozpoczęcia ‘startTime‘, czas zakończenia ‘endTime‘, sala ‘room‘, typ zajęć ‘type‘, wykładowca ‘lecturer‘ oraz data ‘date‘.

4.3. Zarządzanie stanem

4.3.1. Dostawca motywu Listing. 5 (s. 28)

```dart
final darkModeProvider = StateNotifierProvider<DarkModeNotifier, bool
    >((ref) {
  return DarkModeNotifier();
});

class DarkModeNotifier extends StateNotifier<bool> {
  DarkModeNotifier() : super(false);

  void toggleDarkMode(bool value) {
    state = value;
  }
}
```

Listing 5. Provider motywu

Provider ‘darkModeProvider‘ zarządza stanem trybu ciemnego w aplikacji. Klasa ‘DarkModeNotifier‘ dziedziczy po ‘StateNotifier<bool>‘ i umożliwia przełączanie trybu ciemnego za pomocą metody ‘toggleDarkMode‘.

4.4. Usługi API

4.4.1. Serwis powiadomień Listing. 6 (s. 29)

```
class FirebaseApi {
  final _firebaseMessaging = FirebaseMessaging.instance;

  Future<void> initNotifications() async {
    await _firebaseMessaging.requestPermission();
    final token = await _firebaseMessaging.getToken();
    print('Token: $token');
  }

  void handleMessage(RemoteMessage? message) {
    if (message == null) return;
  }
}
```

Listing 6. Serwis powiadomień

Klasa ‘FirebaseApi‘ odpowiada za obsługę powiadomień push. Metoda ‘initNotifications‘ inicjalizuje powiadomienia, a ‘handleMessage‘ obsługuje otrzymane wiadomości.

4.5. Komponenty wielokrotnego użytku

4.5.1. Przycisk stylizowany Listing. 7 (s. 29)

```
class StyledButton extends StatelessWidget {
  final VoidCallback onPressed;
  final Widget child;

  const StyledButton({
    Key? key,
    required this.onPressed,
    required this.child,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return ElevatedButton(
```

```
      onPressed: onPressed,
      style: ElevatedButton.styleFrom(
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(12),
        ),
      ),
      child: child,
    );
  }
}
```

Listing 7. Komponent przycisku

Komponent ‘StyledButton‘ to stylizowany przycisk, który można wielokrotnie używać w aplikacji. Przyjmuje funkcję ‘onPressed‘ oraz widget ‘child‘ jako argumenty.

4.6. Logika biznesowa

4.6.1. Filtrowanie wydarzeń Listing. 8 (s. 30)

```
List<Event> filterEvents(List<Event> events, DateTime selectedDay) {
  return events
    .where((event) => isSameDay(event.date, selectedDay))
    .toList()
    ..sort((a, b) {
      final aStart = DateTime(
        selectedDay.year,
        selectedDay.month,
        selectedDay.day,
        a.startTime.hour,
        a.startTime.minute
      );
      final bStart = DateTime(
        selectedDay.year,
        selectedDay.month,
        selectedDay.day,
        b.startTime.hour,
        b.startTime.minute
      );
      return aStart.compareTo(bStart);
    });
}
```

Listing 8. Filtrowanie wydarzeń

Funkcja ‘filterEvents‘ filtruje i sortuje wydarzenia na podstawie wybranego dnia ‘selectedDay‘.

4.7. Wygodne SDK do komunikacji z backendem

Poniższy kod implementuje klasę `WirtualnySdk`, która pełni rolę wygodnego narzędzia do komunikacji z backendem. Dzięki wykorzystaniu wzorców projektowych takich jak **Singleton** oraz **Factory**, zapewniono efektywną inicjalizację i dostęp do komponentów SDK.

4.7.1. Wzorzec Singleton

Wzorzec **Singleton** umożliwia istnienie tylko jednej instancji klasy `WirtualnySdk` w całej aplikacji. Dzięki temu wszystkie moduły aplikacji mają dostęp do tej samej konfiguracji SDK i mogą współdzielić zasoby. W kodzie poniżej mechanizm Singleton realizowany jest poprzez:

- Prywatne pole statyczne `_instance`, które przechowuje instancję klasy.
- Prywatny konstruktor `_internal`, który zapobiega tworzeniu instancji z zewnątrz.
- Publiczną metodę `initialize()`, która inicjalizuje SDK i ustawia konfigurację.
- Getter `instance`, który zwraca istniejącą instancję SDK lub zgłasza wyjątek, jeśli SDK nie zostało zainicjalizowane.

Kod weryfikuje, czy SDK zostało już zainicjalizowane. Próba wielokrotnej inicjalizacji prowadzi do zgłoszenia wyjątku **Listing. 9 (s. 31)**:

```
if (_instance != null) {
    throw Exception('WirtualnySdk has already been initialized.');
}
```

Listing 9. Zgłoszenie wyjątku

4.7.2. Wzorzec Factory

Wzorzec **Factory** umożliwia tworzenie instancji klasy poprzez wywołanie metody `factory`. W kodzie zastosowano fabrykę do implementacji uproszczonego dostępu do obiektu SDK **Listing. 10 (s. 31)**:

```
factory WirtualnySdk() {
    return instance;
```

```
}
```

Listing 10. Wzorzec Factory

Fabryka zwraca już istniejącą instancję klasy `WirtualnySdk`, co integruje się z mechanizmem Singleton. Umożliwia to bezpieczny i wygodny dostęp do SDK bez potrzeby ręcznego zarządzania instancją.

4.7.3. Modularna budowa SDK

Klasa `WirtualnySdk` zapewnia dostęp do modułów `auth` oraz `notifications`, odpowiedzialnych za uwierzytelnianie i obsługę powiadomień. Każdy z modułów jest instancjonowany jako prywatne pole i dostępny za pośrednictwem getterów **Listing. 11 (s. 32)**:

```
final WirtualnyAuth _auth = WirtualnyAuth();
final WirtualnyNotifications _wirtualnyNotifications =
    WirtualnyNotifications();

WirtualnyAuth get auth => _auth;
WirtualnyNotifications get notifications => _wirtualnyNotifications;
```

Listing 11. WirtualnyAuth

Dzięki takiemu podejściu aplikacja ma zapewnioną modularność i łatwość rozbudowy SDK o kolejne komponenty.

4.7.4. Podsumowanie

Implementacja klasy `WirtualnySdk` pokazuje skuteczne wykorzystanie wzorców Singleton i Factory, dzięki czemu SDK jest łatwe w użyciu, bezpieczne i zgodne z zasadami dobrej architektury. Klasa zapewnia wygodny dostęp do kluczowych funkcji, takich jak uwierzytelnianie i powiadomienia, co czyni ją centralnym punktem integracji z backendem.

4.8. Wysyłanie powiadomień o ogłoszeniach - implementacja backendowa

Poniższy kod w języku TypeScript przedstawia proces wysyłania powiadomień typu FCM (Firebase Cloud Messaging) do wielu urządzeń jednocześnie w formie powiadomień o ogłoszeniach. Obsługiwane są zarówno urządzenia Android, jak i iOS. Kod zawiera również mechanizm logowania oraz obsługi błędów. **Listing. 12 (s. 32)**

```
try {
  getMessaging().sendEachForMulticast({
```

```
    tokens: fcmTokens,
    notification: {
      title: doc.subject,
      body: stripHtml(doc.content_html).result,
    },
    data: {
      type: 'announcement',
      id: doc.id.toString(),
      click_action: 'FLUTTER_NOTIFICATION_CLICK',
    },
    android: {
      priority: doc.priority === 'high' ? 'high' : 'normal',
      notification: {
        clickAction: 'FLUTTER_NOTIFICATION_CLICK',
      },
    },
    apns: {
      headers: {
        'apns-priority': doc.priority === 'high' ? '10' : '5',
      },
    },
    fcmOptions: {
      analyticsLabel: 'announcement',
    },
  });

  if (enableLogs) {
    payload.logger.info(
      `Successfully sent FCM notification for '${collectionSlug}'
   document with ID: '${doc.id}''.`,
    );
  }
} catch (error: unknown) {
  const msg = error instanceof Error ? error.message : error;
  throw new APIError(
    `Failed to send FCM notification for '${collectionSlug}' document
   with ID: '${doc.id}': ${msg}'.`,
  );
}
```

Listing 12. Implementacja wysyłania powiadomień FCM

Opis działania:

- **Powiadomienia FCM:** Funkcja `sendEachForMulticast` umożliwia wysyłanie powiadomień do grupy urządzeń zdefiniowanych w tablicy `fcmTokens`.

- **Struktura powiadomienia:** Powiadomienia zawierają tytuł (`title`), treść (`body`) oraz dane dodatkowe (`data`), takie jak typ powiadomienia (`announcement`) czy identyfikator dokumentu (`id`).
- **Obsługa platform:**
  - **Android:** Ustawiane są priorytety (`high` lub `normal`) oraz akcje kliknięcia.
  - **iOS:** Nagłówki `apns-priority` definiują priorytet wiadomości.
- **Logowanie:** W przypadku sukcesu wysyłania, logowana jest informacja o wysłaniu powiadomienia.
- **Obsługa błędów:** W przypadku błędu, rzucany jest wyjątek `APIError` z odpowiednim komunikatem.

4.9. Integracja z backendem

Aplikacja komunikuje się z backendem Payload CMS poprzez REST API. Główne endpointy:

- `/api/auth/login` - logowanie użytkownika
- `/api/events` - zarządzanie planem zajęć
- `/api/subjects` - zarządzanie przedmiotami
- `/api/notifications` - zarządzanie powiadomieniami

Uwierzytelnianie odbywa się poprzez tokeny JWT przechowywane w `SharedPreferences`.

4.10. Deployment

Projekt PayloadCMS[10] został wdrożony na serwerze Ubuntu[11] z wykorzystaniem Nginx jako reverse proxy, certyfikatu SSL od Certbota, zarządzania DNS przez Cloudflare oraz narzędzi takich jak PM2 i pnpm. PayloadCMS w wersji 3 działa na frameworku Next.js, co zapewnia wydajność i elastyczność w budowie aplikacji.

4.10.1. Uruchamianie aplikacji za pomocą pnpm i PM2

Do zarządzania zależnościami projektu wykorzystano pnpm, co pozwala na szybsze instalacje oraz oszczędność miejsca na dysku. W katalogu projektu wszystkie niezbędne zależności zostały zainstalowane poleceniem **Listing. 13 (s. 35)**:

```
pnpm install
```

Listing 13. Instalacja pnpm

Po zakończeniu konfiguracji i upewnieniu się, że aplikacja działa lokalnie w trybie produkcyjnym (`pnpm build` oraz `pnpm start`), PayloadCMS został uruchomiony i monitorowany za pomocą PM2. PM2 jest używany do zarządzania procesami aplikacji w tle, co zapewnia niezawodność oraz możliwość automatycznego restartu w razie awarii. Aplikację uruchomiono w trybie produkcyjnym następująco **Listing. 14 (s. 35)**:

```
pm2 start npm --name "payload-cms" -- start
```

Listing 14. Uruchamianie pnpm

PM2 zapewnia również możliwość monitorowania procesów oraz logów **Listing. 15 (s. 35)**:

```
pm2 status
pm2 logs payload-cms
```

Listing 15. Sprawdzenie status oraz logów przy pomocy pnpm

4.10.2. Konfiguracja Nginx jako Reverse Proxy

Nginx został skonfigurowany jako reverse proxy, aby przekierowywać ruch HTTP/HTTPS na serwer PayloadCMS działający na porcie aplikacji. Oto przykład konfiguracji wirtualnego hosta **Listing. 16 (s. 35)**:

```
server {
    listen 80;
    server_name example.com www.example.com;

    location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

Listing 16. Konfiguracja nginx

Po dodaniu powyższej konfiguracji Nginx został przeładowany **Listing. 17 (s. 35)**:

```
sudo systemctl reload nginx
```

Listing 17. przeładowanie konfiguracji

4.10.3. SSL i Cloudflare

Domenę skonfigurowano z użyciem Cloudflare oraz certyfikatu SSL wygenerowanego przez Certbota. Certbot zaktualizował konfigurację Nginx w celu obsługi HTTPS **Listing. 18 (s. 36)**:

```
sudo certbot --nginx -d example.com -d www.example.com
```

Listing 18. Ustawienie cerbota

Cloudflare zostało skonfigurowane do zarządzania DNS (rekordy `A` i `CNAME`) oraz aktywowano tryb `Full SSL/TLS encryption mode`.

4.10.4. Podsumowanie

Dzięki wykorzystaniu pnpm do zarządzania zależnościami oraz PM2 do nadzorowania procesu aplikacji, PayloadCMS w wersji 3 (opartej na Next.js) działa stabilnie i wydajnie. Dodatkowo Nginx, Certbot oraz Cloudflare zapewniają bezpieczeństwo, wysoką dostępność i optymalizację dla użytkowników końcowych.

4.11. Przygotowanie pakietu .apk aplikacji mobilnej

Proces przygotowania pakietu `.apk` dla aplikacji mobilnej stworzonej w Flutterze obejmuje kilka kroków, od konfiguracji projektu po generowanie gotowego pliku instalacyjnego. Poniżej przedstawiono szczegóły całego procesu.

4.11.1. Konfiguracja projektu Flutter

Przed rozpoczęciem procesu budowy upewniono się, że projekt Flutter jest odpowiednio skonfigurowany do pracy w środowisku produkcyjnym:

1. Plik `pubspec.yaml` został zaktualizowany, a wszystkie zależności zostały zainstalowane **Listing. 19 (s. 36)**:

```
    flutter pub get

```

Listing 19. Pobranie pakietów w flutterze

2. Zdefiniowano odpowiednią konfigurację w pliku `AndroidManifest.xml`, w tym `applicationId`, pozwolenia (`permissions`) oraz ustawienia dotyczące wersji aplikacji (`versionCode` i `versionName`).

3. Plik `build.gradle` w module `app` zawierał właściwą konfigurację dla środowiska produkcyjnego, w tym `minSdkVersion` i `targetSdkVersion`.

4.11.2. Budowanie pakietu .apk

Pakiet `.apk` został wygenerowany za pomocą polecenia Fluttera do budowy w trybie produkcyjnym **Listing. 20 (s. 37)**:

```
flutter build apk --release
```

Listing 20. Budowa oficjalnego pakietu .apk

Wygenerowany plik `app-release.apk` został zapisany w katalogu `build/app/outputs/flutter-apk/`.

4.11.3. Podpisywanie aplikacji

Aby aplikacja mogła być zainstalowana na urządzeniach użytkowników, wymagane było jej podpisanie:

1. Wygenerowano klucz kryptograficzny (`.keystore`) za pomocą narzędzia `keytool` **Listing. 21 (s. 37)**:

```
    keytool -genkey -v -keystore my-release-key.jks \
        -keyalg RSA -keysize 2048 -validity 10000 \
        -alias my-key-alias

```

Listing 21. Generowanie kluczu krypotraficznego

2. Plik `my-release-key.jks` został umieszczony w katalogu `android/app/`.

3. W pliku `android/key.properties` zdefiniowano dane dostępowe do klucza **Listing. 22 (s. 37)**:

```
    storePassword=your-store-password
    keyPassword=your-key-password
    keyAlias=my-key-alias
    storeFile=path/to/my-release-key.jks

```

Listing 22. Dane dostępowe do klucza

4. Plik `build.gradle` został zaktualizowany, aby uwzględnić konfigurację podpisu **Listing. 23 (s. 38)**:

```
    android {
        signingConfigs {
            release {
                keyAlias 'my-key-alias'
                keyPassword 'your-key-password'
                storeFile file('path/to/my-release-key.jks')
                storePassword 'your-store-password'
            }
        }
        buildTypes {
            release {
                signingConfig signingConfigs.release
            }
        }
    }

```

Listing 23. Uwzględnienie podpisu w pliku build.gradle

4.11.4. Testowanie pakietu

Gotowy pakiet `.apk` został przetestowany na urządzeniach fizycznych i emulatorach, aby upewnić się, że działa poprawnie. Weryfikację przeprowadzono za pomocą polecenia **Listing. 24 (s. 38)**:

```
flutter install
```

Listing 24. Testowanie pakietu

4.11.5. Podsumowanie

Pakiet `.apk` aplikacji mobilnej został przygotowany i podpisany, co pozwala na jego dystrybucję do użytkowników poprzez ręczną instalację lub publikację w sklepie Google Play. Wykorzystanie Fluttera oraz właściwa konfiguracja środowiska produkcyjnego gwarantują wysoką wydajność oraz zgodność z różnymi urządzeniami Android.

5. Testowanie

5.1. Logowanie do aplikacji

W tabeli nr. 5.1 (s. 39) możemy zobaczyć że logowanie przebiega pomyślnie, jeśli dane wprowadzone przez użytkownika są poprawne, w przypadku podania niepoprawnego adresu e-mailu, hasła lub błędu składowego, logowanie zostanie odrzucone.

Funkcja	**Typ Danych**	**Rezultat**	**Działanie**
Logowanie	Poprawne dane	Działa	Poprawnie
	Poprawny email, błędne hasło	Błąd	Poprawnie
	Błędny email, poprawne hasło	Błąd	Poprawnie
	Błędne dane	Błąd	Poprawnie
	Hasło, które ma mniej niż 6 liter	Błąd	Poprawnie
	Błędny format adresu email	Błąd	Poprawnie

Tab. 5.1. Logowanie

5.2. Powiadomienia push

W tabeli nr. 5.2 (s. 39) możemy zobaczyć, że powiadomienia push są wysyłane i odbierane poprawnie.

Funkcja	**Typ Danych**	**Rezultat**	**Działanie**
Powiadomienia push	Poprawne dane	Działa	Poprawnie
	Brak połączenia z internetem	Błąd	Poprawnie
	Błędne dane	Błąd	Poprawnie

Tab. 5.2. Powiadomienia push

5.3. Biometria

W tabeli nr. 5.3 (s. 40) możemy zobaczyć, że funkcja biometrii działa poprawnie, jeśli użytkownik udzielił odpowiednich uprawnień.

Funkcja	Typ Danych	Rezultat	Działanie
Biometria	Udzielone uprawnienia	Działa	Poprawnie
	Brak uprawnień	Błąd	Poprawnie
	Błędne dane biometryczne	Błąd	Poprawnie

Tab. 5.3. Biometria

5.4. Synchronizacja danych

W tabeli nr. 5.4 (s. 40) możemy zobaczyć, że synchronizacja danych działa poprawnie, zarówno w trybie online, jak i offline.

Funkcja	Typ Danych	Rezultat	Działanie
Synchronizacja danych	Połączenie z internetem	Działa	Poprawnie
	Brak połączenia z internetem	Działa (tryb offline)	Poprawnie
	Błędne dane	Błąd	Poprawnie

Tab. 5.4. Synchronizacja danych

5.5. Autoryzacja i uprawnienia

W tabeli nr. 5.5 (s. 40) możemy zobaczyć, że autoryzacja i zarządzanie uprawnieniami działa poprawnie.

Funkcja	Typ Danych	Rezultat	Działanie
Autoryzacja	Poprawne dane	Działa	Poprawnie
	Błędne dane	Błąd	Poprawnie
	Brak uprawnień	Błąd	Poprawnie

Tab. 5.5. Autoryzacja i uprawnienia

5.6. Robienie zdjęcia i wysyłanie do serwera

W tabeli nr. 5.6 (s. 41) możemy zobaczyć, że funkcja robienia zdjęcia i wysyłania go do serwera działa poprawnie.

Funkcja	Typ Danych	Rezultat	Działanie
Robienie zdjęcia	Poprawne dane	Działa	Poprawnie
	Brak uprawnień do kamery	Błąd	Poprawnie
	Błąd kamery	Błąd	Poprawnie
Wysyłanie zdjęcia	Poprawne dane	Działa	Poprawnie
	Brak połączenia z internetem	Błąd	Poprawnie
	Błąd serwera	Błąd	Poprawnie

Tab. 5.6. Robienie zdjęcia i wysyłanie do serwera

5.7. Pobieranie danych z serwera

W tabeli nr. 5.7 (s. 41) możemy zobaczyć, że funkcja pobierania danych z serwera działa poprawnie.

Funkcja	Typ Danych	Rezultat	Działanie
Pobieranie danych	Poprawne dane	Działa	Poprawnie
	Brak połączenia z internetem	Błąd	Poprawnie
	Błąd serwera	Błąd	Poprawnie
	Błędne dane	Błąd	Poprawnie

Tab. 5.7. Pobieranie danych z serwera

5.8. Integracja z zewnętrznymi API

W tabeli nr. 5.10 (s. 43) możemy zobaczyć, że integracja z zewnętrznymi API działa poprawnie, umożliwiając pobieranie i wysyłanie danych.

Funkcja	**Typ Danych**	**Rezultat**	**Działanie**
Pobieranie danych	Poprawne dane	Działa	Poprawnie
	Brak danych	Błąd	Poprawnie
	Błędne dane	Błąd	Poprawnie
Wysyłanie danych	Poprawne dane	Działa	Poprawnie
	Błędne dane	Błąd	Poprawnie

Tab. 5.8. Integracja z zewnętrznymi API

5.9. Pobieranie danych do kalendarza (planu zajęć)

W tabeli nr. 5.9 (s. 43) możemy zobaczyć, że funkcja pobierania danych do kalendarza działa poprawnie.

Funkcja	Typ Danych	Rezultat	Działanie
Pobieranie danych do kalendarza	Poprawne dane	Działa	Poprawnie
	Brak połączenia z internetem	Błąd	Poprawnie
	Błąd serwera	Błąd	Poprawnie
	Błędne dane	Błąd	Poprawnie

Tab. 5.9. Pobieranie danych do kalendarza (planu zajęć)

5.10. Pobieranie przedmiotów

W tabeli nr. 5.10 (s. 43) możemy zobaczyć, że funkcja pobierania przedmiotów działa poprawnie.

Funkcja	Typ Danych	Rezultat	Działanie
Pobieranie przedmiotów	Poprawne dane	Działa	Poprawnie
	Brak połączenia z internetem	Błąd	Poprawnie
	Błąd serwera	Błąd	Poprawnie
	Błędne dane	Błąd	Poprawnie

Tab. 5.10. Pobieranie przedmiotów

5.11. Pobieranie ocen

W tabeli nr. 5.11 (s. 44) możemy zobaczyć, że funkcja pobierania ocen działa poprawnie.

Funkcja	Typ Danych	Rezultat	Działanie
Pobieranie ocen	Poprawne dane	Działa	Poprawnie
	Brak połączenia z internetem	Błąd	Poprawnie
	Błąd serwera	Błąd	Poprawnie
	Błędne dane	Błąd	Poprawnie

Tab. 5.11. Pobieranie ocen

5.12. Pobieranie dat egzaminów

W tabeli nr. 5.12 (s. 44) możemy zobaczyć, że funkcja pobierania dat egzaminów działa poprawnie.

Funkcja	Typ Danych	Rezultat	Działanie
Pobieranie dat egzaminów	Poprawne dane	Działa	Poprawnie
	Brak połączenia z internetem	Błąd	Poprawnie
	Błąd serwera	Błąd	Poprawnie
	Błędne dane	Błąd	Poprawnie

Tab. 5.12. Pobieranie dat egzaminów

6. Podręcznik użytkownika

6.1. Logowanie

Po uruchomieniu aplikacji po raz pierwszy użytkownik zostanie przekierowany na ekran logowania. Należy wprowadzić adres e-mail oraz hasło, a następnie kliknąć przycisk **'Zaloguj' Rys. 6.1 (s. 45)**.

Rys. 6.1. Ekran logowania

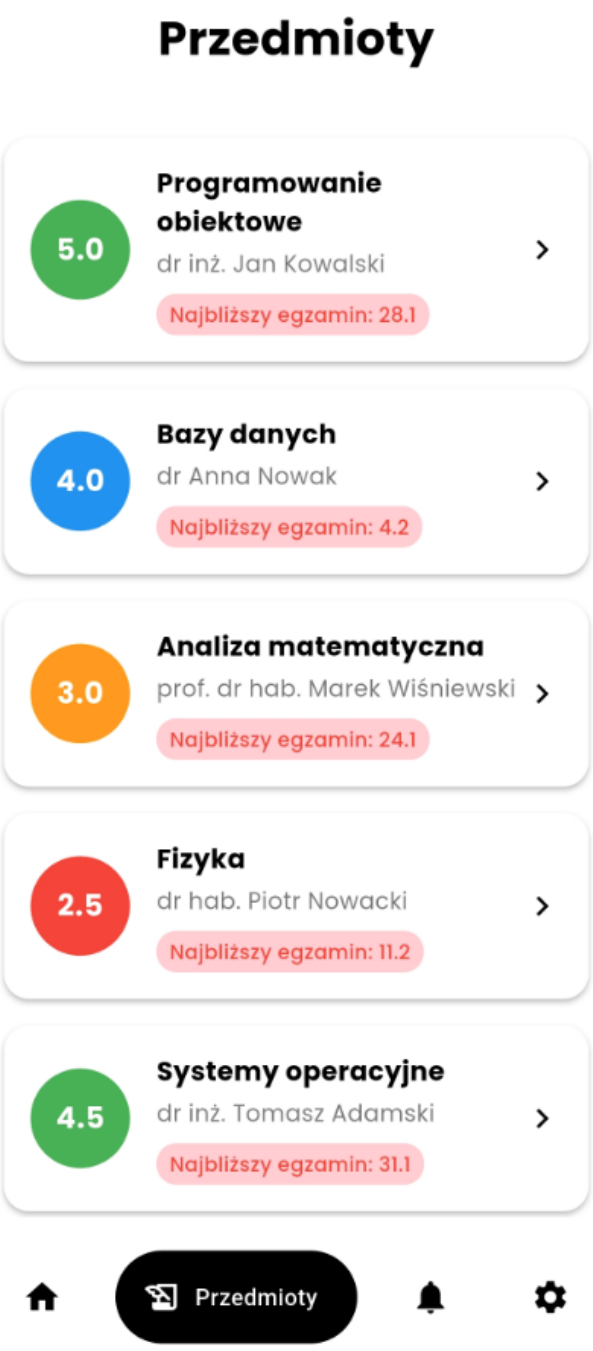
6.2. Ekran główny

Po zalogowaniu użytkownik zostanie przekierowany na ekran główny. Na ekranie głównym znajduje się plan zajęć, który pokazuje nadchodzące zajęcia. Kliknięcie na konkretny dzień wyświetli szczegóły, takie jak sala, wykładowca i godziny. **Rys. 6.2 (s. 46)**

Rys. 6.2. Ekran główny

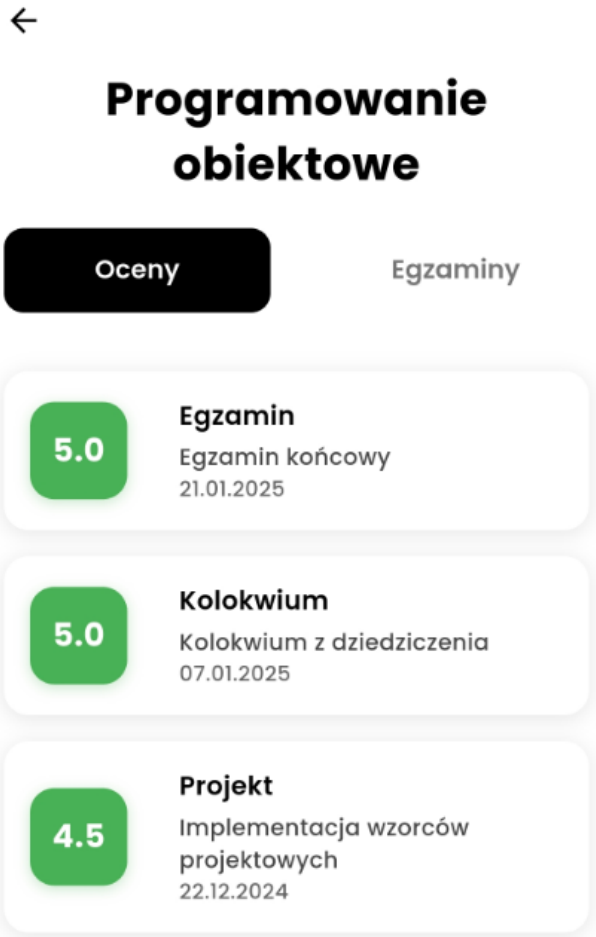
6.3. Przedmioty

Aby zobaczyć jakie przedmioty mamy na danym semetrze należy kliknąć na ikonę Przedmioty. Na ekranie przedmioty znajduję się lista przedmitów jakie posiadamy w aktualnym semestrze, średnia ocen , oraz data zbliżających się egzaminów. **Rys. 6.3 (s. 47)**



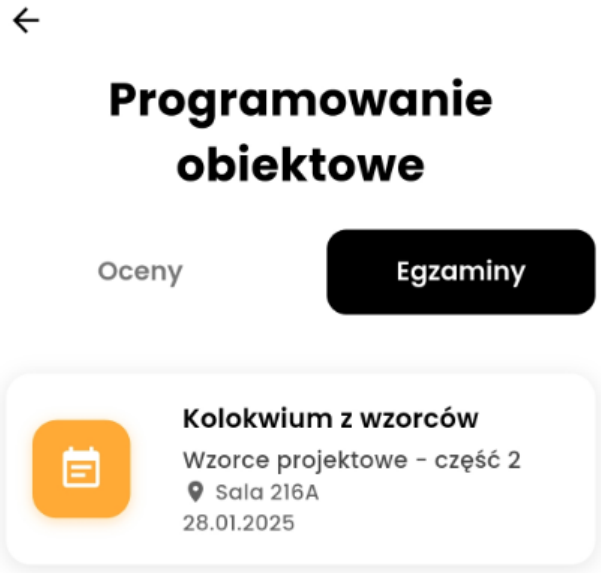
Rys. 6.3. Ekran Przedmioty

Jeżeli chcemy zobaczyć szczegóły danego przedmiotu (Np. wszystkie oceny). Należy kliknąć na szczałkę obok tego przedmiotu co poskutkuje przeniesieniem nas do jego szczegółów. **Rys. 6.4 (s. 48)**



Rys. 6.4. Ekran szczegółowy dla danego Przedmiotu - Oceny

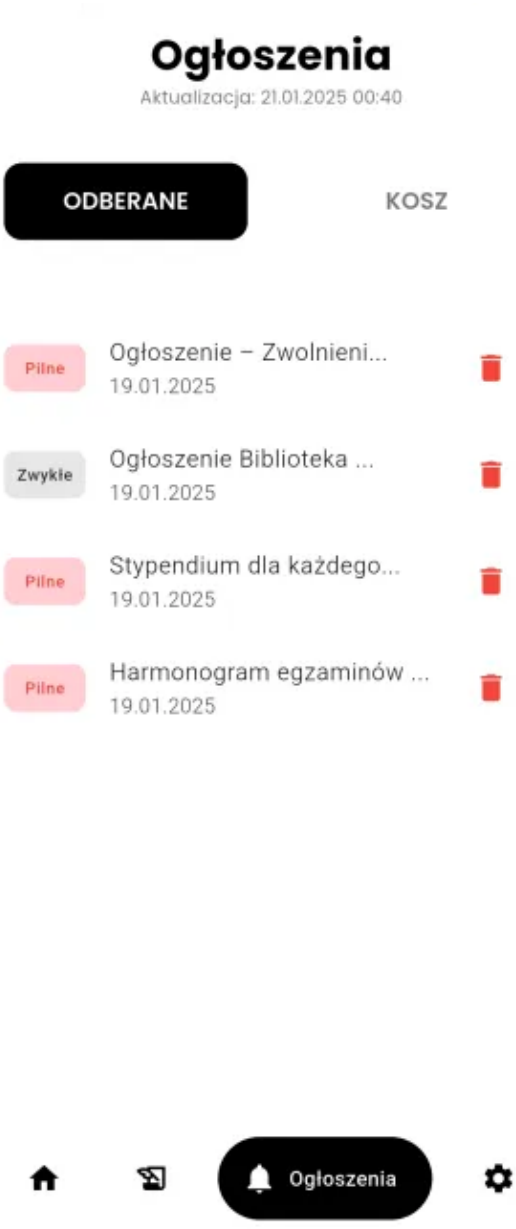
Jeżeli chcielibyśmy przejść do szczegółów dotyczących Egzaminu dla danego przedmiotu należy kliknąć na okienko Egzaminy. **Rys. 6.5 (s. 49)**



Rys. 6.5. Ekran szczegółowy dla danego Przedmiotu - Egzaminy

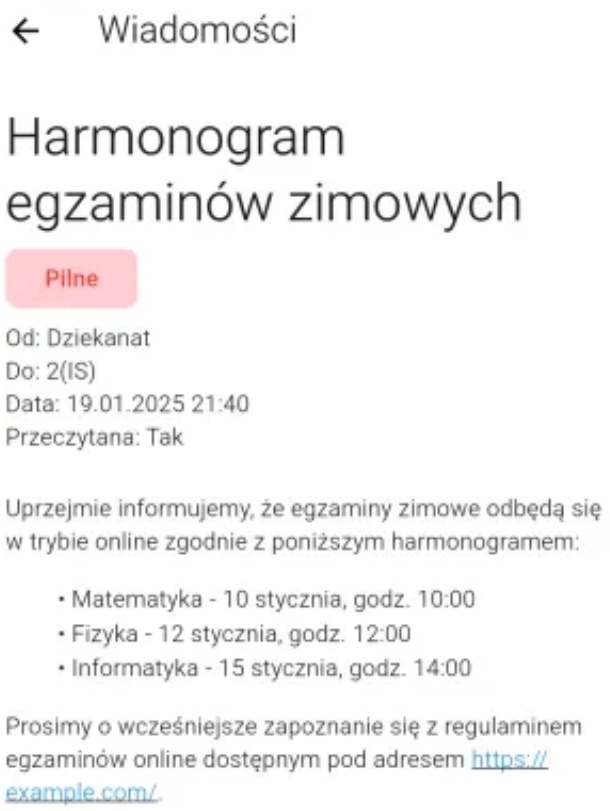
6.4. Ogłoszenia

Aby zobaczyć ogłoszenia, należy kliknąć na ikonę Ogłoszenia w dolnym pasku nawigacyjnym. Po zrobieniu tego wyświetli się lista ogłoszeń. W zależności od ustawionego piorytetu danego ogłoszenia kolor boxa obok niego będzię inny (Np. Pilne - Czerwony, Zwykłe - Szare). **Rys. 6.6 (s. 50)**



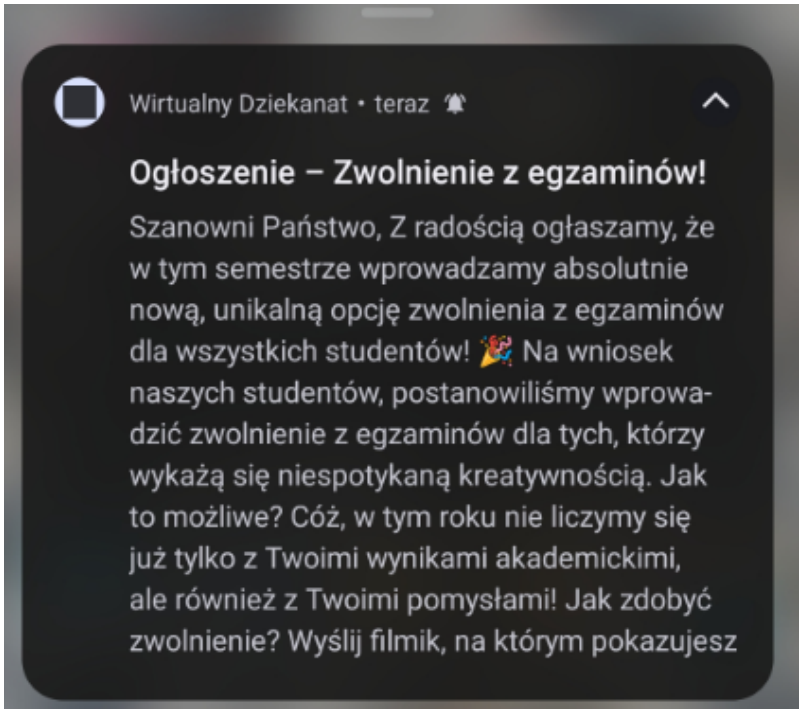
Rys. 6.6. Ekran ogłoszeń

Aby zobaczyć szczegóły danego ogłoszenie musimy je kliknąć. **Rys. 6.7 (s. 51)**

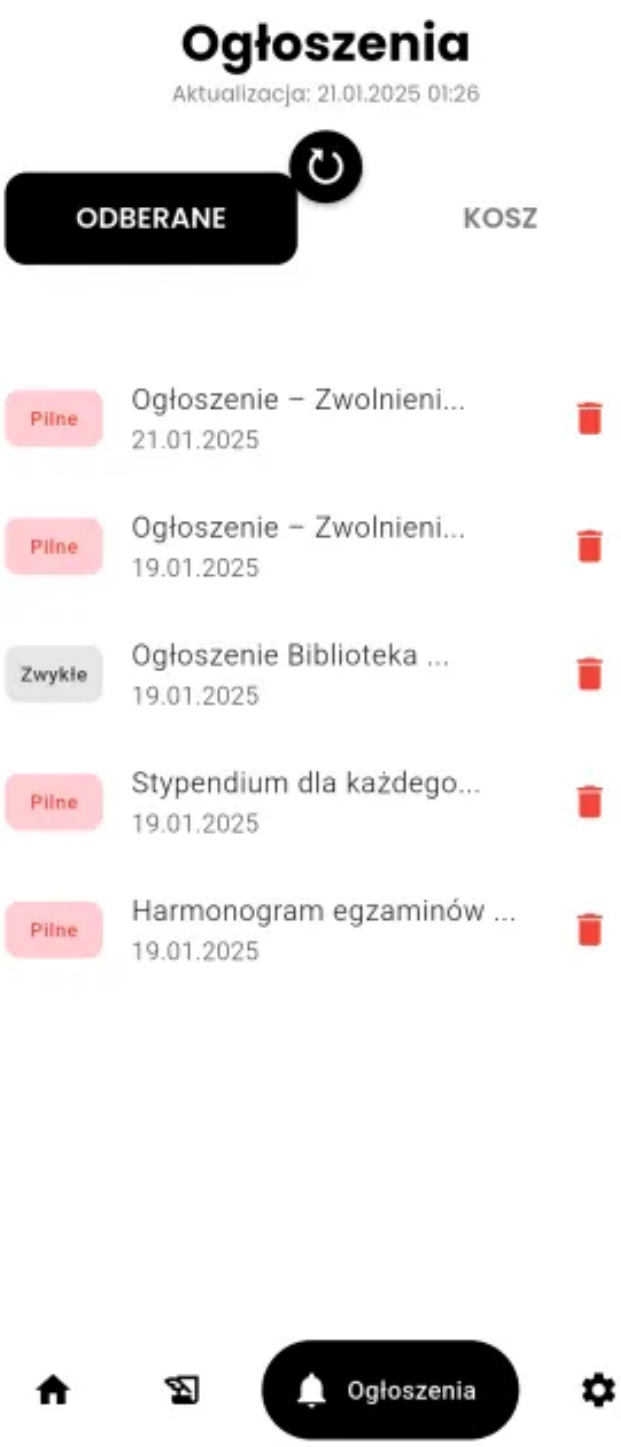


Rys. 6.7. Szczegóły danego ogłoszenia

Jeżeli na telefonie przyjdzie do nas powiadomie **Rys. 6.8 (s. 52)**. To po kliknięciu na nie zostatniemy odrazu przekierowani do ekranu z ogłoszeniami **Rys. 6.11 (s. 55)**.



Rys. 6.8. Push notification



Rys. 6.9. Push notification

6.5. Tryb offline - dostęp do zapisanych danych bez dostępu do Internetu

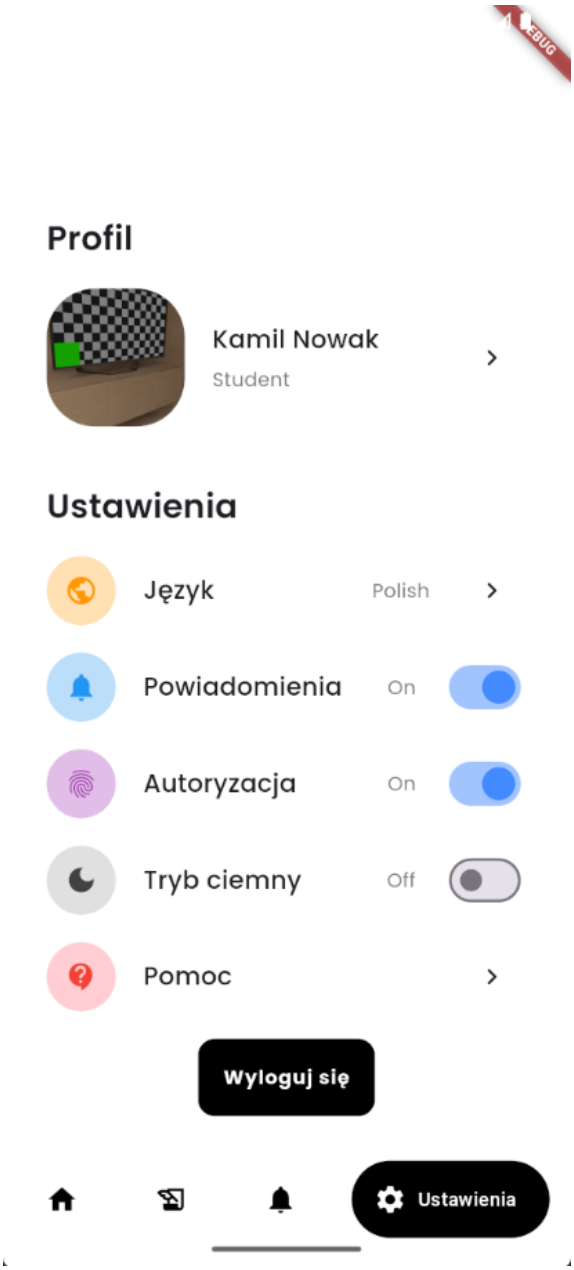
Rys. 6.10. Utracono połączenie z Internetem

Rys. 6.11. Odzyskano połączenie z Internetem

6.6. Ustawienia

Aby przejść do ustawień, należy kliknąć na ikonę ustawień w dolnym pasku nawigacyjnym. W ustawieniach użytkownik może: **Rys. 6.12 (s. 56)**

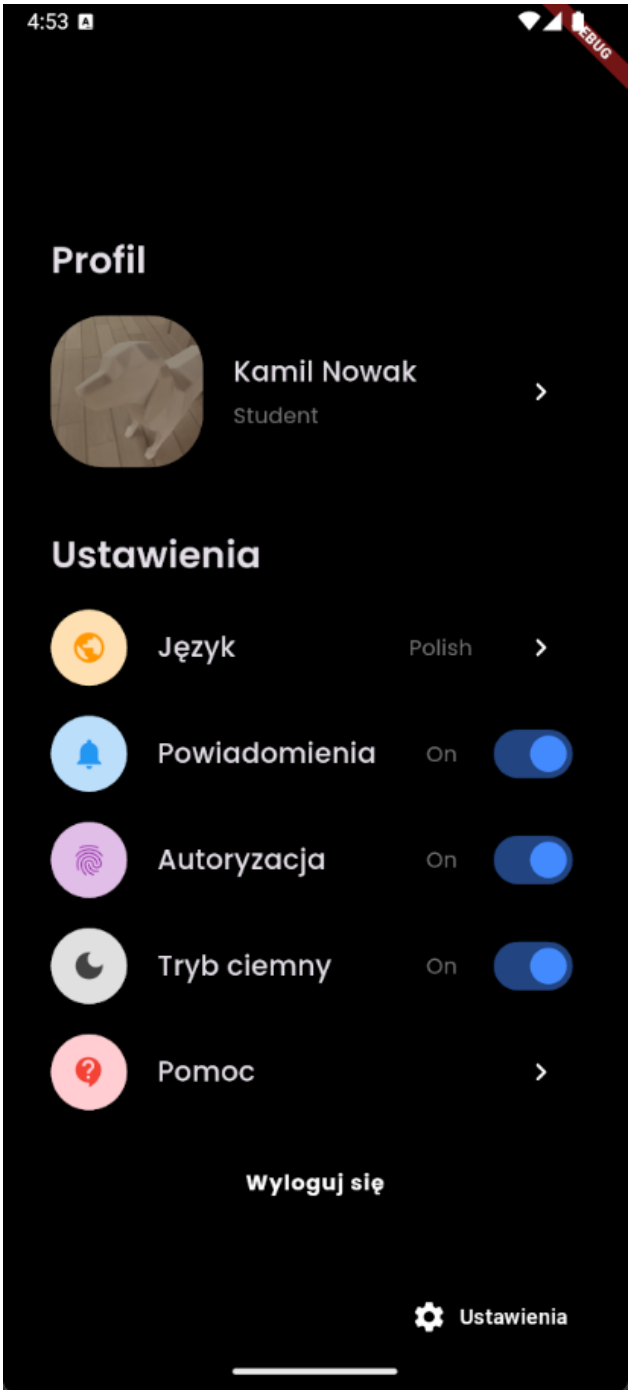
- Włączyć lub wyłączyć tryb ciemny
- Zarządzać powiadomieniami
- Włączyć lub wyłączyć biometrię



Rys. 6.12. Ekran ustawień

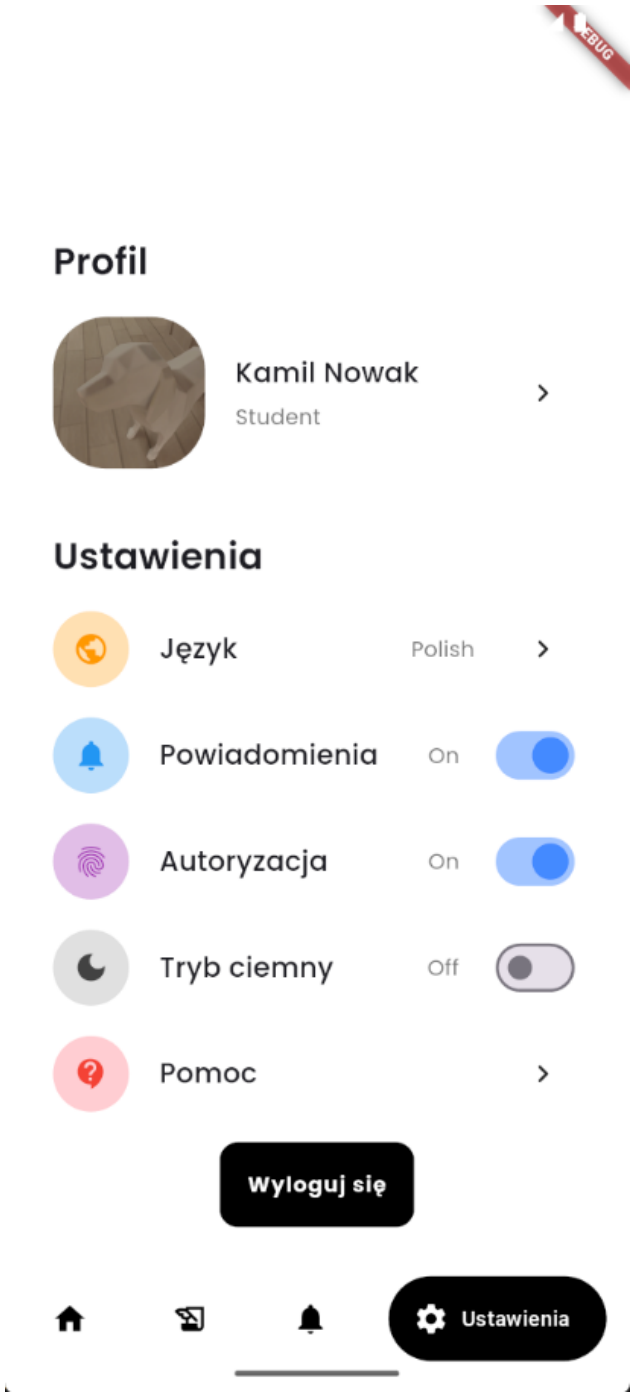
6.6.1. Tryb ciemny

Aby włączyć tryb ciemny, należy przełączyć odpowiedni przełącznik w ustawieniach. Aplikacja automatycznie zmieni motyw na ciemny. **Rys. 6.13 ,6.14(s. 57 , 58)**
DarkMode ON:



Rys. 6.13. Włączenie darkmoda

DarkMode OFF:

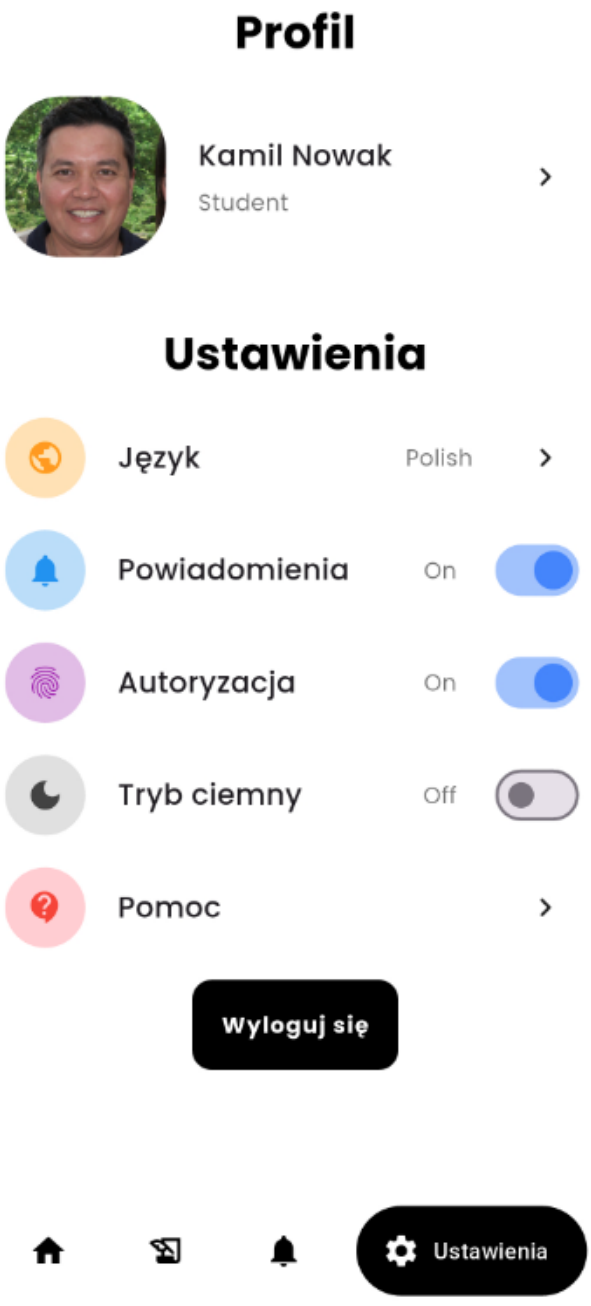


Rys. 6.14. Wyłączenie darkmoda

6.6.2. Zarządzanie powiadomieniami

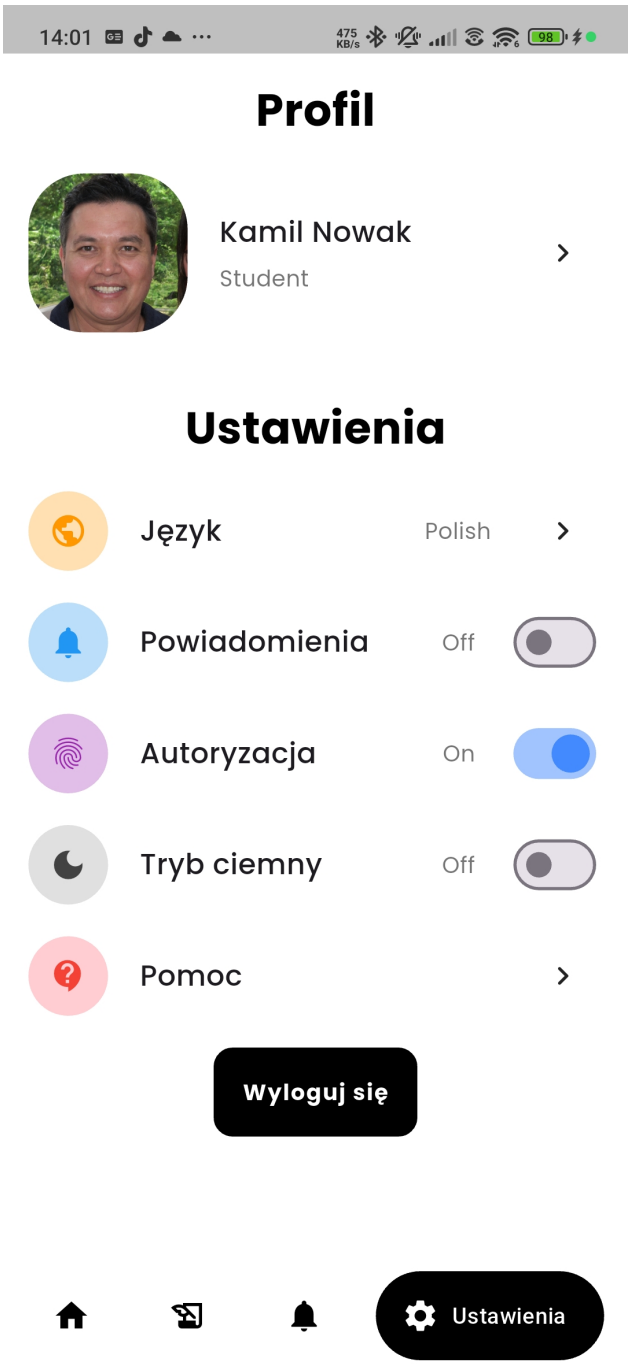
W ustawieniach można włączyć lub wyłączyć powiadomienia dla różnych typów zdarzeń, takich jak nadchodzące zajęcia czy nowe oceny.**Rys. 6.15, 6.16 (s. 59, 60)**

Powiadomienia ON:



Rys. 6.15. Zarządzanie powiadomieniami ON

Powiadomienia OFF:

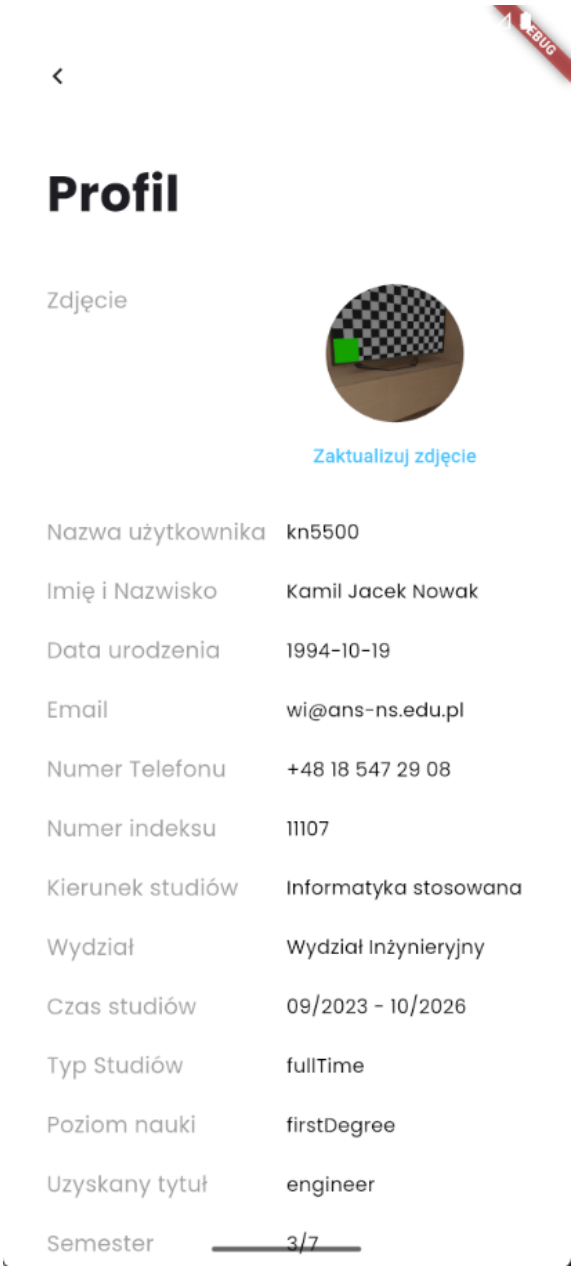


Rys. 6.16. Zarządzanie powiadomieniami OFF

6.6.3. Profil użytkownika

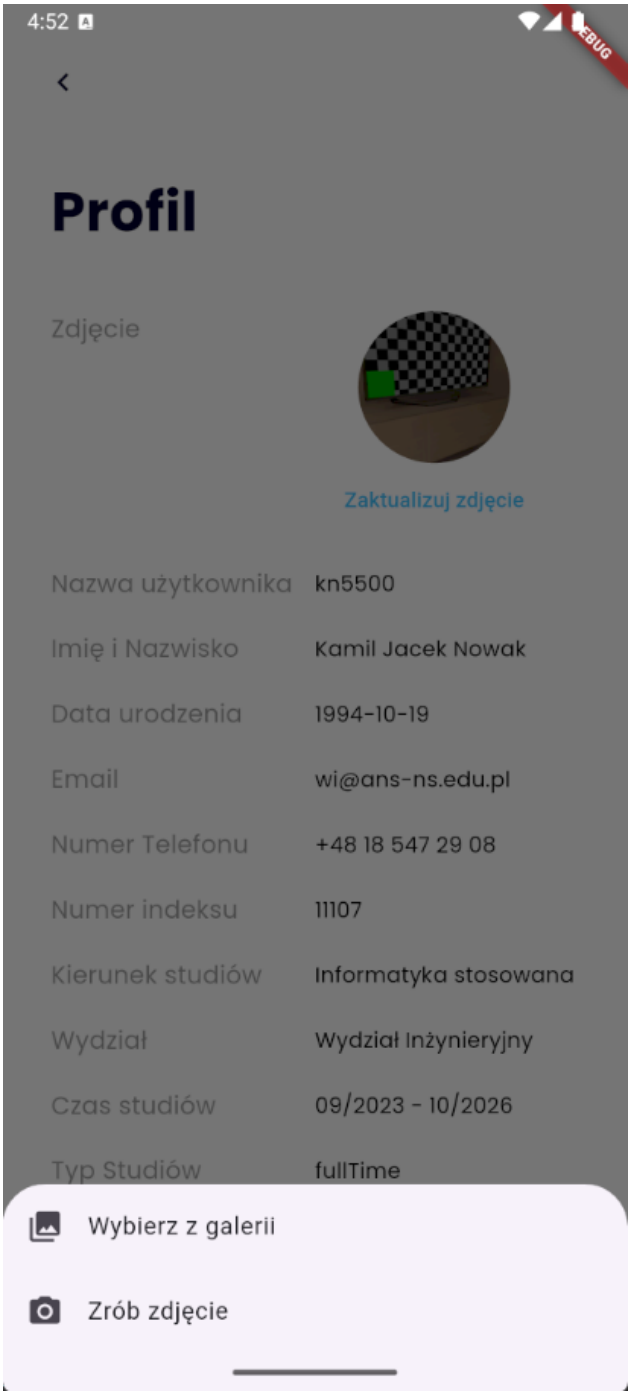
Kliknięcie na avatar w ustawieniach przenosi użytkownika do ekranu profilu **Rys. 6.17 (s. 61)**:, gdzie można

- Zmienić zdjęcie profilowe
- Zobaczyć informacje o sobie, takie jak imię, nazwisko, adres e-mail



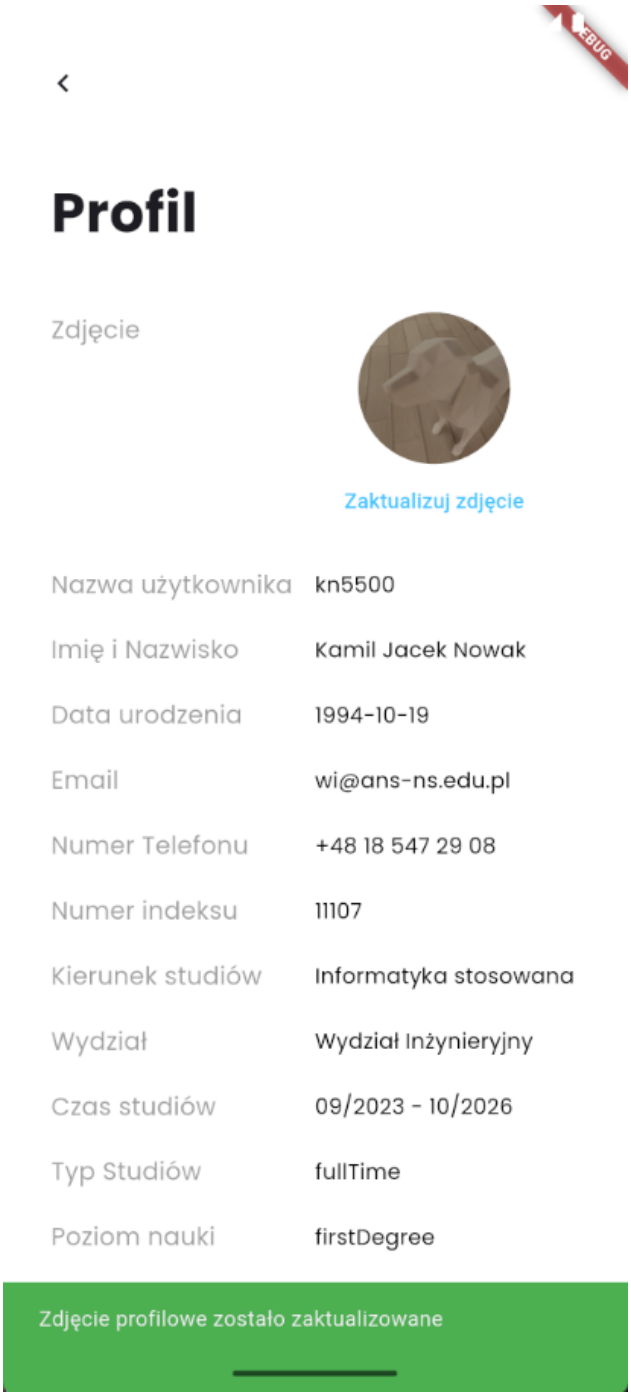
Rys. 6.17. Ekran ustawień

Aby zaktualizować zdjęcie należy kliknąć na przycisk **‘Zaktualizuj Zdjęcie‘**.Po kliknięciu mamy do wyboru 2 opcje (Wybierz z galerii, Zrób zdjęcie) **Rys. 6.18 (s. 62)**



Rys. 6.18. Wybór aktualizacji zdjęcia

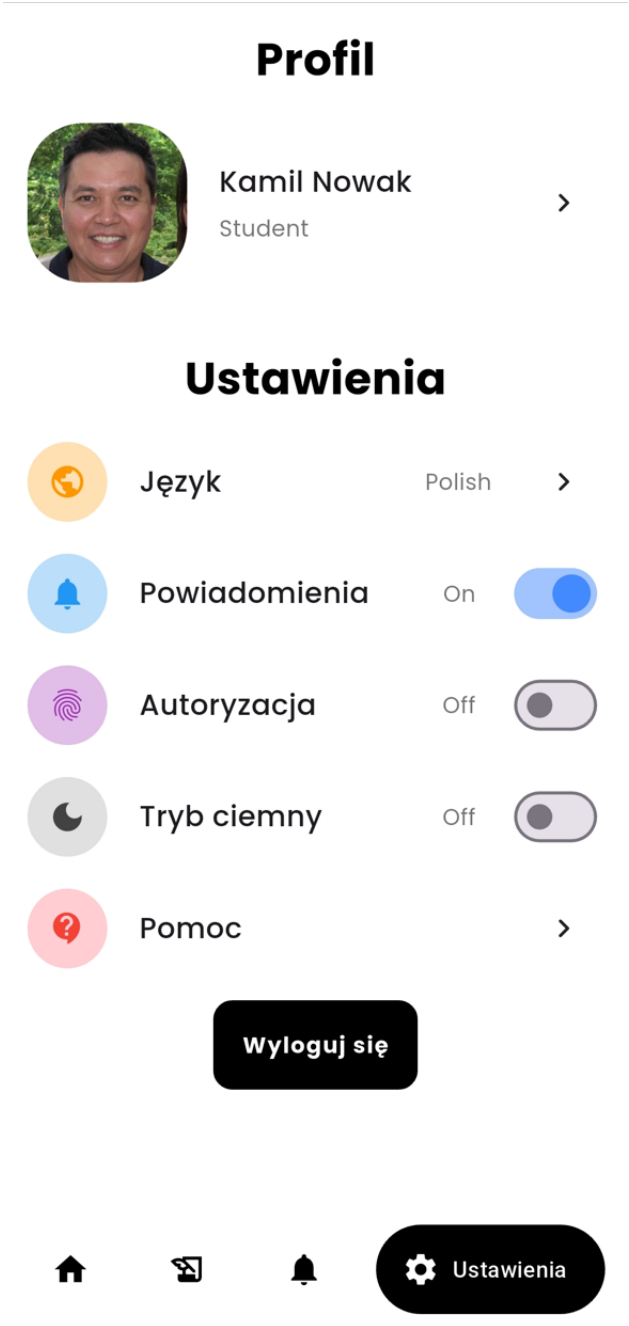
Po zrobieniu zdjęcia Gdy wyszstko przebiegło pomyślnie powienien wyświetlić się zielony komunikat ”Zdjęcie zostało zaktualizowane”co oznacza że zostało pomyślnie zaktualizowane w naszej bazy danych. **Rys. 6.19 (s. 63)**



Rys. 6.19. Zdjęcie zostało poprawnie zrobione i do naszej bazy dancyh

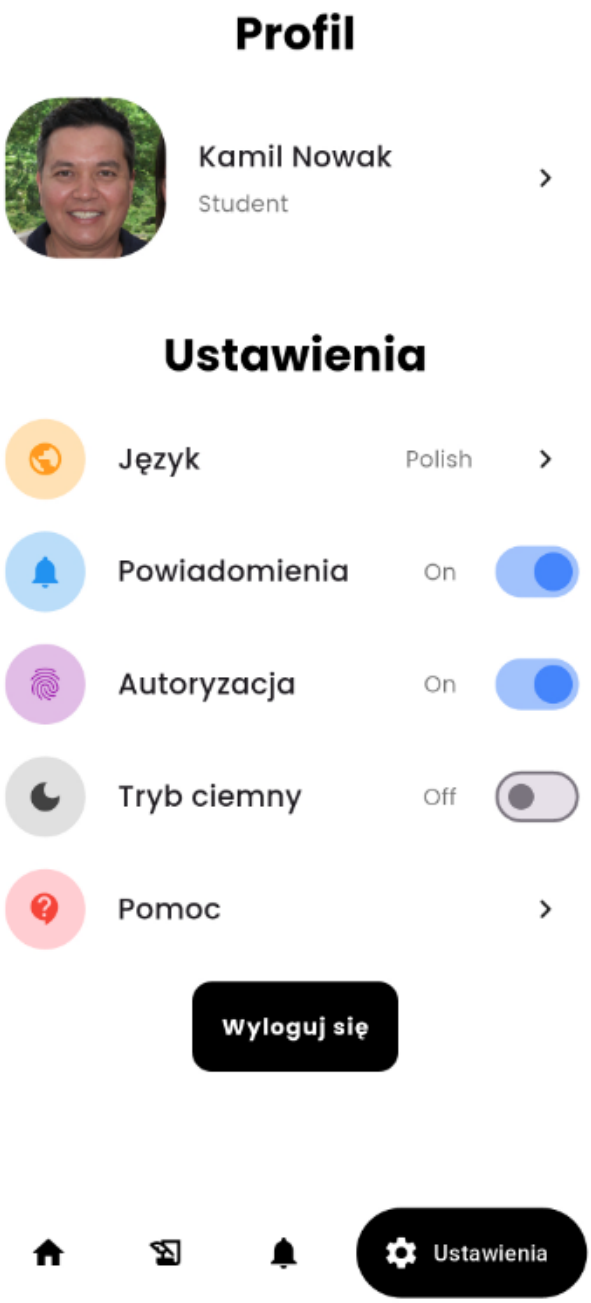
6.6.4. Biometria

Po restracie apliakcji biometria pozwala na automatyczne logowanie się wcześniejszymi danymi poprzez odcisk palca przez co nie musimy na nowo wpisywać naszych danych. Biomteria OFF **Rys. 6.20 (s. 64)**:



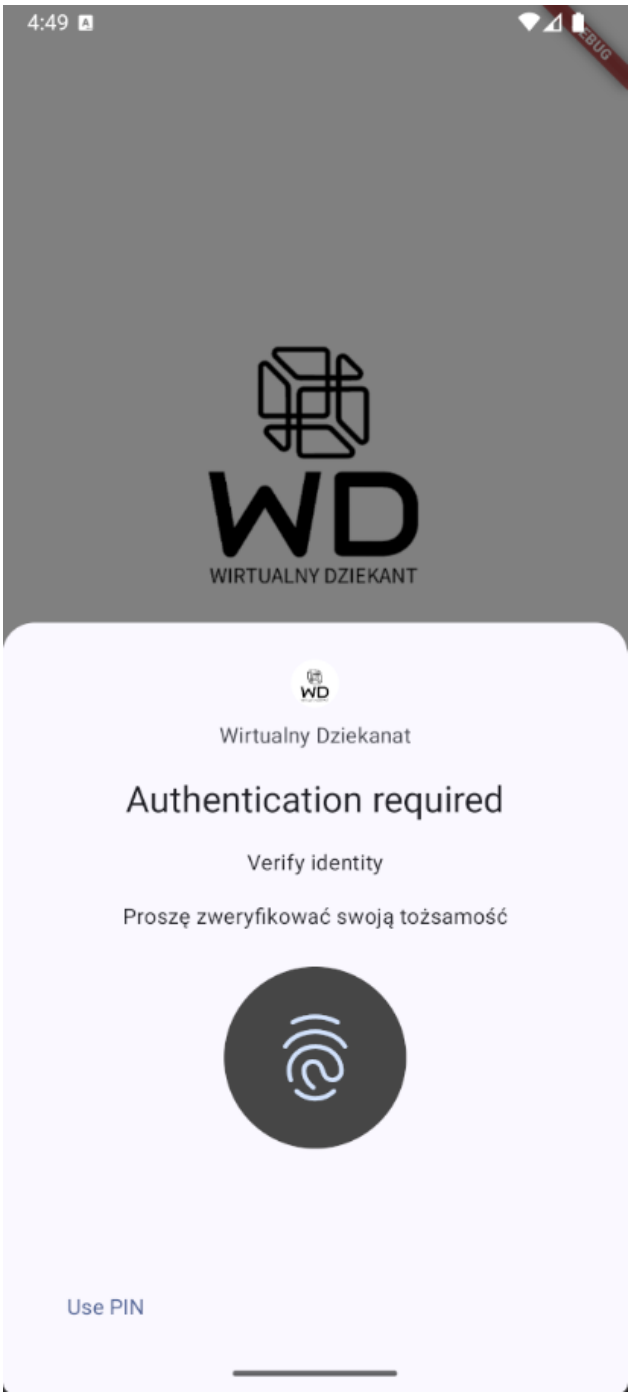
Rys. 6.20. Biomteria OFF

Biomteria ON **Rys. 6.21 (s. 65)**:



Rys. 6.21. Biometria ON

Gdy mamy włączoną biomterię to po wyłączeniu i właczeniu na nowo apikacji ta poprosi nas o podanie odcisku palca. Należy przyłożyć palec. Gdy weryfikacje się powiedzie to przeniesie nas do wnętrza aplikacji **Rys. 6.22 (s. 66)**:



Rys. 6.22. Poproszenie użytkownika o podanie odcisku palca

6.6.5. Wylogowanie

Aby się wylogować, należy kliknąć na przycisk ”Wyloguj się”na ekranie profilu. Użytkownik zostanie wylogowany i przekierowany na ekran logowania. **Rys. 6.23 (s. 67)**

Rys. 6.23. Widok po wylogowaniu

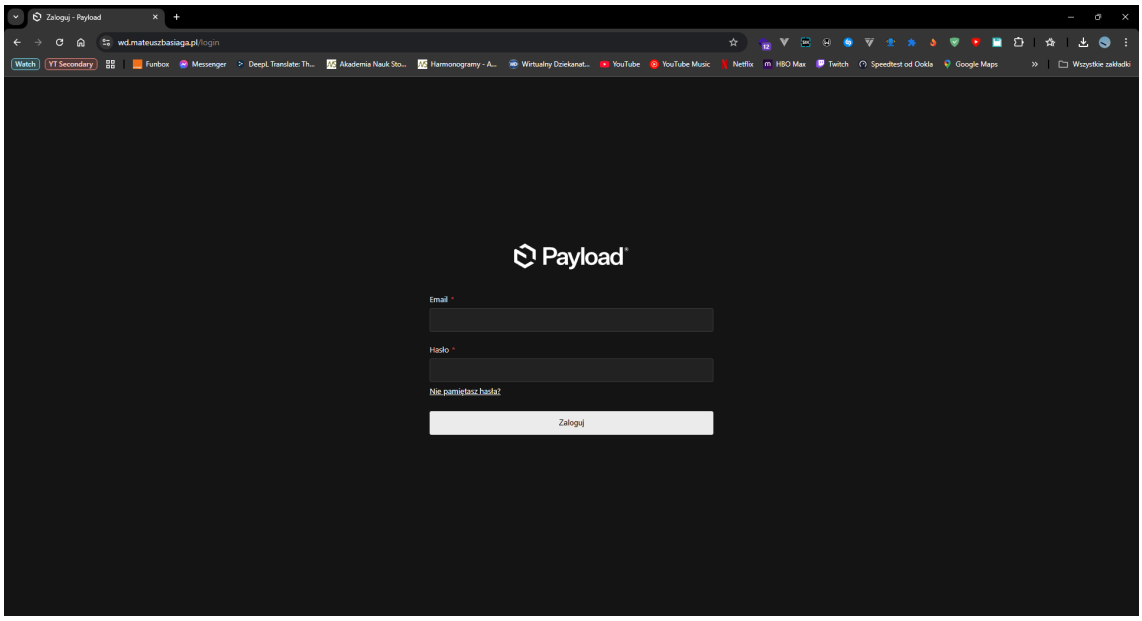
6.7. Pull to refresh

Aby użyć funkcji **pull to refresh** np. w panelu głównym musimy zrobić ruch palcem z góry aplikacji do środka (Powinno się utworzyć takie małe koło). Gdy już mamy palec na środku należy go puścić przez co naszę dane się automatycznie odświeżą. **Rys. 6.24 (s. 68)**

Rys. 6.24. Użycie pull to refresh

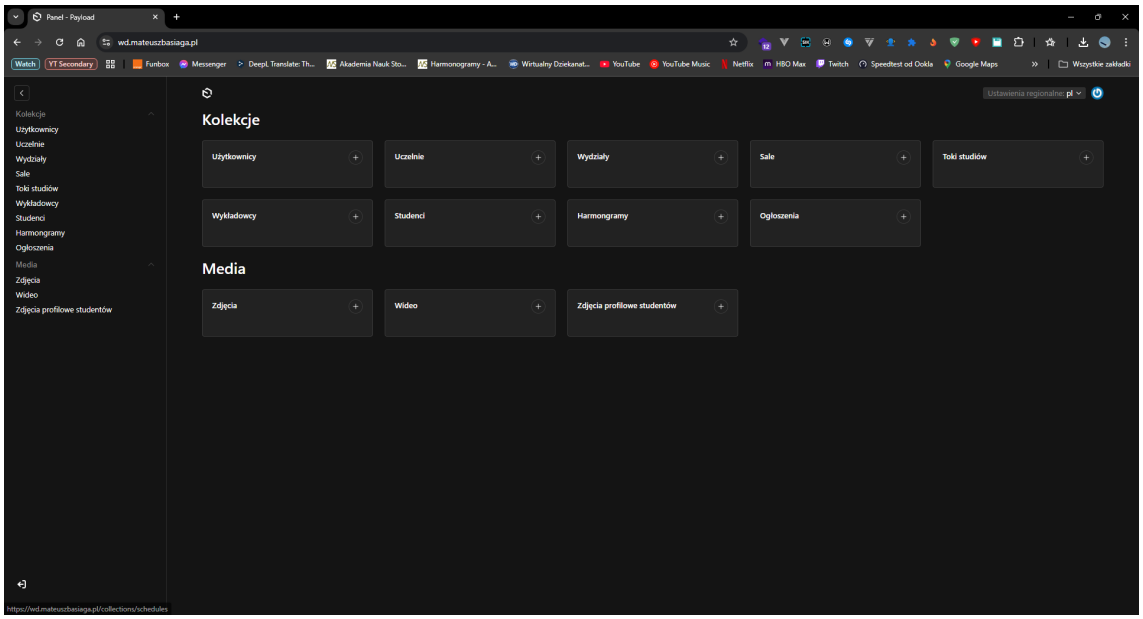
6.8. Obsługa backendu

Aby zalogować się do panelu administracyjnego backendu, należy przejść na stronę logowania i wprowadzić swoje dane uwierzytelniające. Po zalogowaniu użytkownik zostanie przekierowany do głównego interfejsu użytkownika, gdzie można zarządzać danymi aplikacji. **Rys. 6.25 (s. 69)**



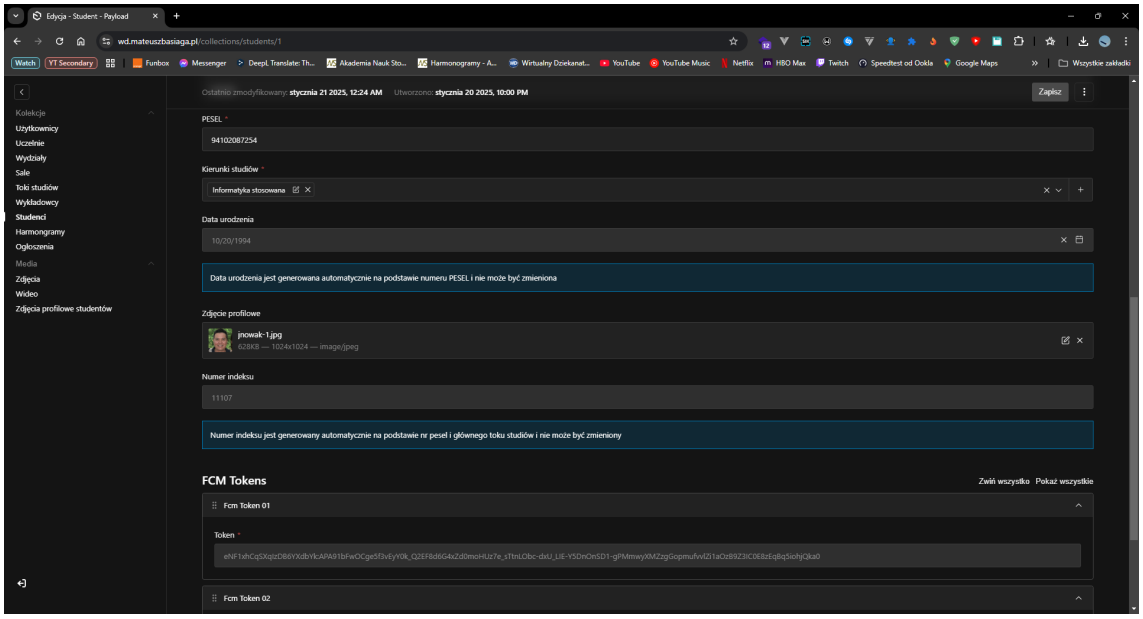
Rys. 6.25. Ekran logowania

Główny interfejs użytkownika pozwala na przeglądanie i edytowanie danych, takich jak profile studentów, plan zajęć, ogłoszenia i inne. Użytkownik może również dodawać nowe wpisy oraz usuwać istniejące. **Rys. 6.26 (s. 70)**



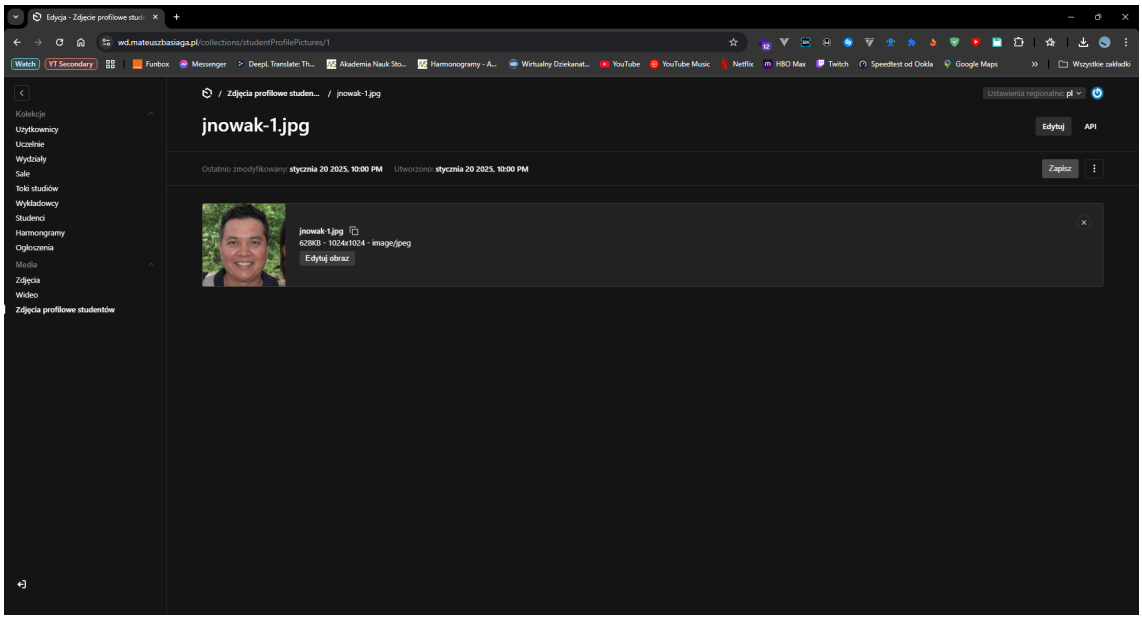
Rys. 6.26. Widok głównego interfejsu użytkownika

Profil studenta zawiera szczegółowe informacje o danym studencie, takie jak imię, nazwisko, adres e-mail, numer indeksu, rocznik, oceny i inne. Administrator może edytować te dane oraz dodawać nowe wpisy. **Rys. 6.27 (s. 70)**



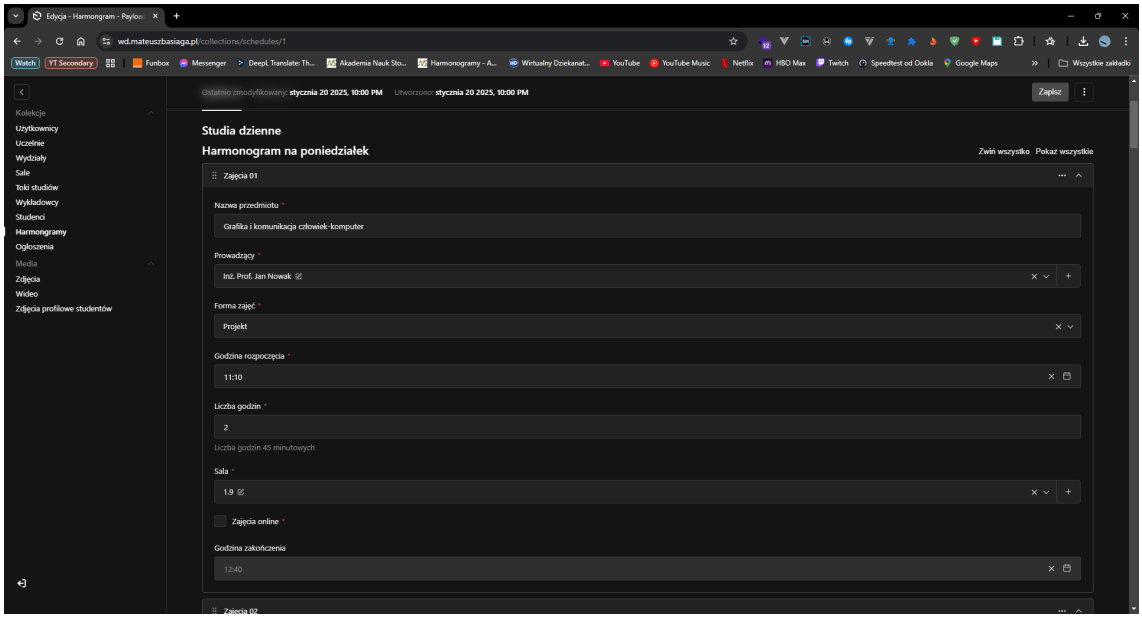
Rys. 6.27. Widok profilu studenta

Backend umożliwia również zarządzanie zdjęciami profilowymi studentów oraz innymi danymi multimedialnymi. Administrator może dodawać, edytować i usuwać zdjęcia profilowe. **Rys. 6.28 (s. 71)**



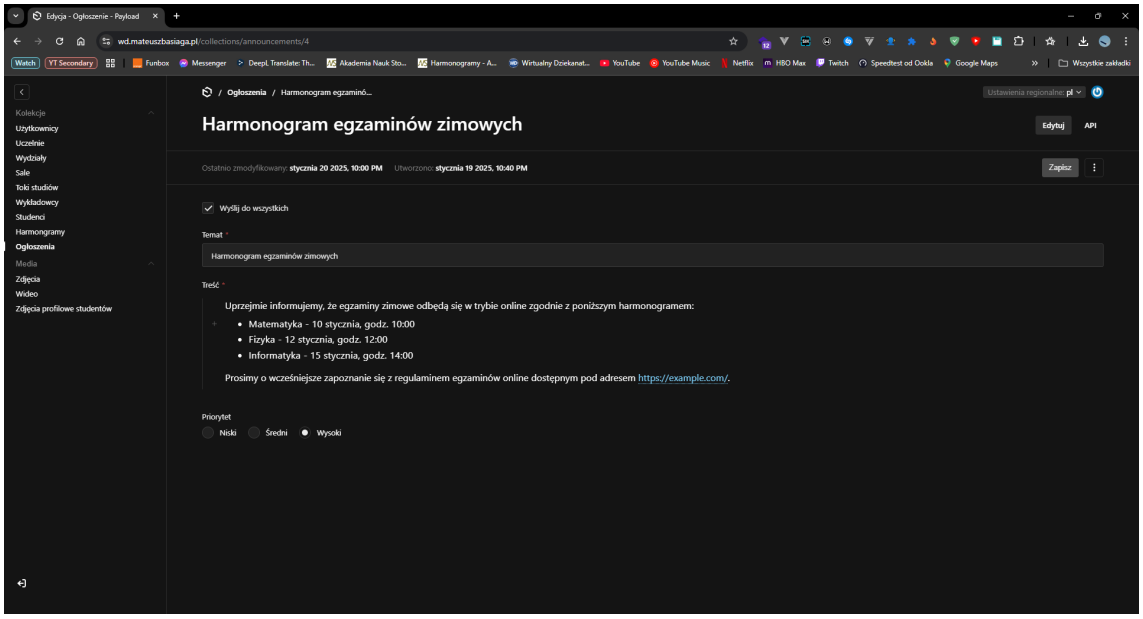
Rys. 6.28. Zdjęcia profilowe studentów - przykład danych multimedialnych

Plan zajęć można przeglądać i edytować za pomocą interfejsu backendu. Administrator może dodawać nowe zajęcia, edytować istniejące oraz usuwać wpisy. Plan zajęć jest synchronizowany z aplikacją mobilną, dzięki czemu studenci mają zawsze aktualne informacje. **Rys. 6.29 (s. 71)**



Rys. 6.29. Widok planu zajęć toku studiów

Ogłoszenia można zarządzać za pomocą interfejsu backendu. Administrator może dodawać nowe ogłoszenia, edytować istniejące oraz usuwać wpisy. Ogłoszenia są synchronizowane z aplikacją mobilną, dzięki czemu studenci są na bieżąco informowani o ważnych wydarzeniach. **Rys. 6.30 (s. 72)**



Rys. 6.30. Widok ogłoszenia

7. Literatura

[1] *Strona internetowa Git*. URL: https://git-scm.com/doc (term. wiz. 11. 01. 2024).

[2] *Strona internetowa Visual Studio Code*. URL: https://code.visualstudio.com/ (term. wiz. 11. 01. 2024).

[3] *Strona internetowa AndroidStudio*. URL: https://developer.android.com/studio/intro?hl=pl (term. wiz. 11. 01. 2024).

[4] *Strona internetowa Flutter docs*. URL: https://docs.flutter.dev/ (term. wiz. 11. 01. 2024).

[5] *Strona internetowa Doxygen*. URL: https://www.doxygen.nl/ (term. wiz. 11. 01. 2024).

[6] *Strona internetowa Doxygen Awesome*. URL: https://github.com/jothepro/doxygen-awesome-css (term. wiz. 11. 01. 2024).

[7] *Strona internetowa LeftHook*. URL: https://github.com/evilmartians/lefthook (term. wiz. 11. 01. 2024).

[8] *Strona internetowa Commitlint*. URL: https://commitlint.js.org/ (term. wiz. 11. 01. 2024).

[9] *Strona internetowa Github Actions*. URL: https://docs.github.com/en/actions (term. wiz. 11. 01. 2024).

[10] *Repozytorium backendu*. URL: https://github.com/Me-Phew/wirtualny-dziekanat-cms (term. wiz. 11. 01. 2024).

[11] *Link do backendu*. URL: https://wd.mateuszbasiaga.pl/ (term. wiz. 11. 01. 2024).

[12] *Strona internetowa Github Copilot*. URL: https://github.com/features/copilot (term. wiz. 11. 01. 2024).

[13] *Strona internetowa RiverPod*. URL: https://riverpod.dev/ (term. wiz. 11. 01. 2024).

[14] *Paczka flutter Google Fonts*. URL: https://pub.dev/packages/google_fonts (term. wiz. 11. 01. 2024).

[15] *Paczka flutter Google Fonts*. URL: https://pub.dev/packages/google_fonts (term. wiz. 11. 01. 2024).

[16] *Paczka flutter Google Nav Bar*. URL: https://pub.dev/packages/google_nav_bar (term. wiz. 11. 01. 2024).

[17] *Paczka flutter Table Calendar*. URL: https://pub.dev/packages/table_calendar (term. wiz. 11. 01. 2024).

[18] *Paczka flutter Intl*. URL: https://pub.dev/packages/intl (term. wiz. 11. 01. 2024).

[19] *Paczka flutter Shared Preferences*. URL: https://pub.dev/packages/shared_preferences (term. wiz. 11. 01. 2024).

[20] *Paczka flutter Image Picker*. URL: https://pub.dev/packages/image_picker (term. wiz. 11. 01. 2024).

[21] *Paczka flutter Local Auth*. URL: https://pub.dev/packages/local_auth (term. wiz. 11. 01. 2024).

Spis rysunków

Spis tabel

Spis listingów