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 2024

Spis treści

1. Ogólne określenie wymagań **4**

1.1. Główne funkcje aplikacji 4

2. Określenie wymagań szczegółowych **6**

2.1. Cel aplikacji 6

3. Projektowanie **9**

3.1. Przygotowanie narzędzi 9
3.2. Architektura systemu 10
3.3. Framework Flutter 10
3.4. Architektura systemu 11
3.5. Projekt interfejsu użytkownika 11
3.5.1. Style i motywy 11
3.5.2. Nawigacja 11
3.6. Model danych 12
3.6.1. Główne encje 12
3.7. Bezpieczeństwo 12
3.8. Synchronizacja danych 13
3.9. Diagram UML 13

4. Implementacja **15**

4.1. Struktura projektu 15
4.2. Modele danych 15
4.2.1. Model użytkownika 15
4.2.2. Model wydarzenia 16
4.3. Zarządzanie stanem 16
4.3.1. Dostawca motywu 16
4.4. Usługi API 17
4.4.1. Serwis powiadomień 17
4.5. Komponenty wielokrotnego użytku 17
4.5.1. Przycisk stylizowany 17

4.6. Logika biznesowa 18
4.6.1. Filtrowanie wydarzeń 18
4.7. Integracja z backendem 19

5. Testowanie **20**

6. Podręcznik użytkownika **21**

6.1. Logowanie 21
6.2. Ekran główny 21
6.3. Powiadomienia 21
6.4. Wiadomości 21
6.5. Ustawienia 21
6.5.1. Tryb ciemny 22
6.5.2. Zarządzanie powiadomieniami 22
6.5.3. Profil użytkownika 22
6.5.4. Wylogowanie 22

Literatura **23**

Spis rysunków **23**

Spis tabel **24**

Spis listingów **25**

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, e-dokumentów oraz ułatwiając komunikację z administracją uczelni.

- **Zakres aplikacji:**
  Studenci: dostęp do ocen, planu zajęć, harmonogramu egzaminów, komunikacja z dziekanatem.
  Wykładowcy/Pracownicy: dostęp do harmonogramu zajęć, oceny studentów, komunikacja z dziekanatem.
  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.
  Formularze zgłoszeń: wnioski o zaświadczenia, prośby do dziekanatu.

- **Opis interfejsu użytkownika i elementów interaktywnych:**
  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):** Wyświetlane informacje: skrót do ocen, nadchodzących zajęć, powiadomienia o ważnych wydarzeniach.
  Przyciski:
  „Oceny" – po kliknięciu, przejście do ekranu z listą ocen.
  „Plan zajęć" – po kliknięciu, podgląd planu zajęć (interaktywny kalendarz).
  „Harmonogram egzaminów" – otwiera harmonogram egzaminów z możliwością filtrowania według przedmiotu/semestru.
  Automatyczne zdarzenia: wyświetlanie powiadomień push o nadchodzących zajęciach, ocenach, ważnych wydarzeniach.

- **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:
  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**:
  Lista egzaminów: możliwość filtrowania według przedmiotu, prowadzącego, daty.
  Opcja zapisu: jeśli wymagana rejestracja na egzamin, po kliknięciu „Zapisz się" użytkownik zapisuje się na egzamin.
  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
- **VSCode** – edytor kodu źródłowego, który zapewnia wsparcie dla wielu języków programowania i umożliwia instalację rozszerzeń wspierających programowanie. Dokumentacja narzędzia: https://code.visualstudio.com/docs
- **Android Studio** – środowisko programistyczne wykorzystywane głównie do testowania aplikacji na emulatorze Android oraz zarządzania SDK. Dokumentacja: https://developer.android.com/studio/intro
- **Flutter SDK** – zestaw narzędzi do tworzenia aplikacji wieloplatformowych. Dokumentacja: https://flutter.dev/docs
- **Doxygen** – narzędzie do generowania dokumentacji automatycznej na podstawie komentarzy w kodzie źródłowym. Dokumentacja narzędzia: https://www.doxygen.nl/
- **Doxygen Awesome** – motyw graficzny dostosowujący wygląd strony wygenerowanej przez Doxygen do współczesnych standardów. Więcej informacji: https://github.com/jothepro/doxygen-awesome-css
- **Lefthook** – narzędzie do zarządzania hookami Git, które wspiera automatyczne formatowanie, walidację kodu oraz zgodność wiadomości commitów z konwencją. Dokumentacja: https://github.com/evilmartians/lefthook
- **Commitlint** – narzędzie do sprawdzania zgodności wiadomości commitów z konwencją *Conventional Commits*. Dokumentacja: https://commitlint.js.org/
- **GitHub Actions** – platforma do automatyzacji procesów CI/CD. Umożliwia automatyczną walidację commitów, generowanie dokumentacji oraz wersjonowanie wydań. Dokumentacja: https://docs.github.com/en/actions

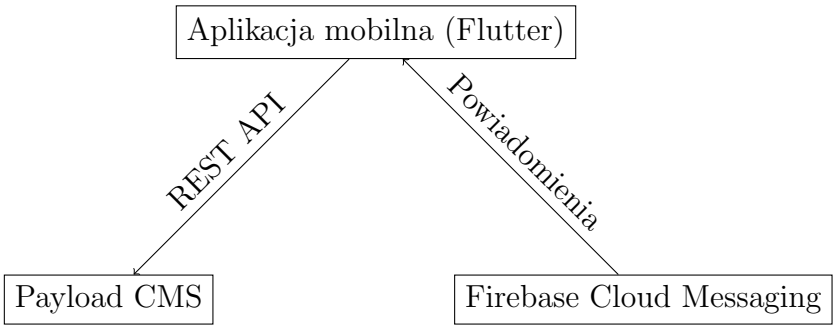
3.2. Architektura systemu

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

3.3. Framework Flutter

System składa się z trzech głównych komponentów:



Rys. 3.1. Architektura systemu

3.4. 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)

3.5. Projekt interfejsu użytkownika

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

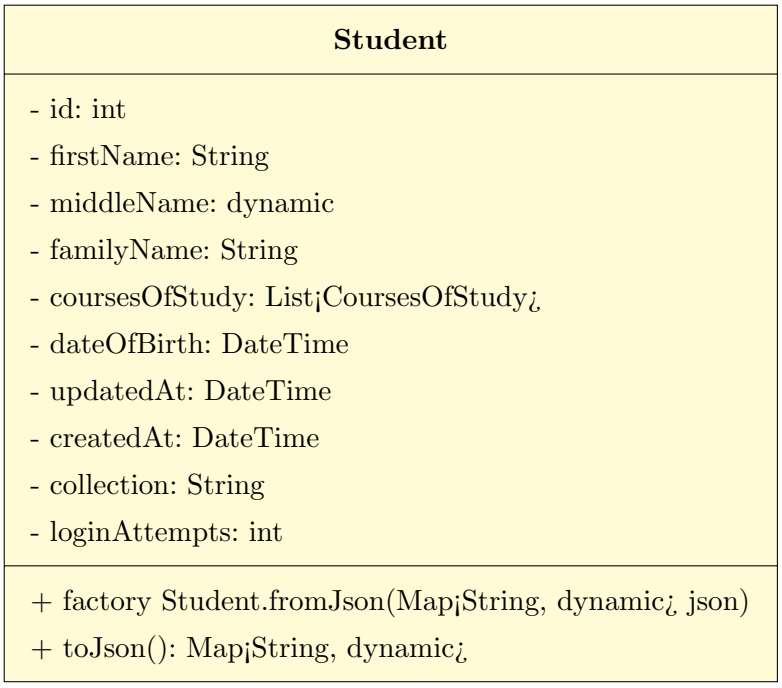
- **Autoryzacja**:
  - JWT tokens
  - Refresh tokens
  - Szyfrowanie danych wrażliwych

- **Walidacja danych**:
  - Walidacja formularzy
  - Sanityzacja danych wejściowych
  - Obsługa błędów

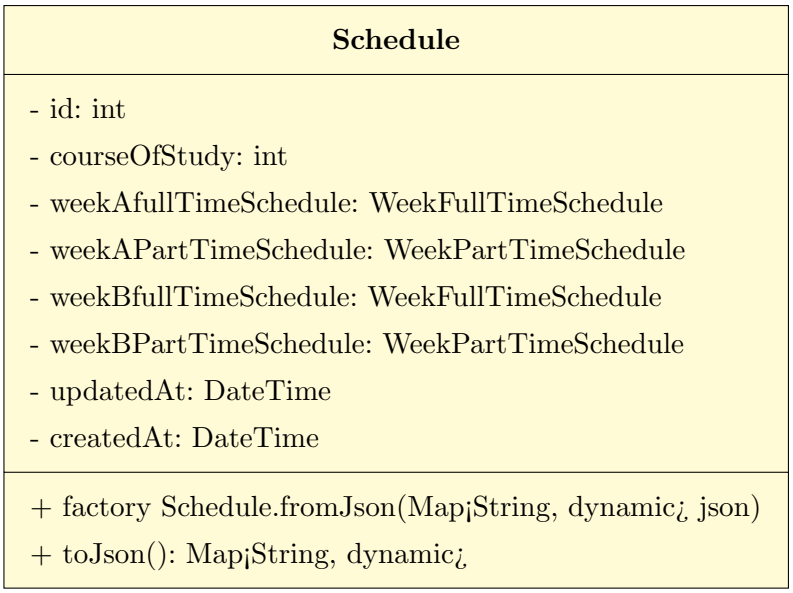
3.8. Synchronizacja danych

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

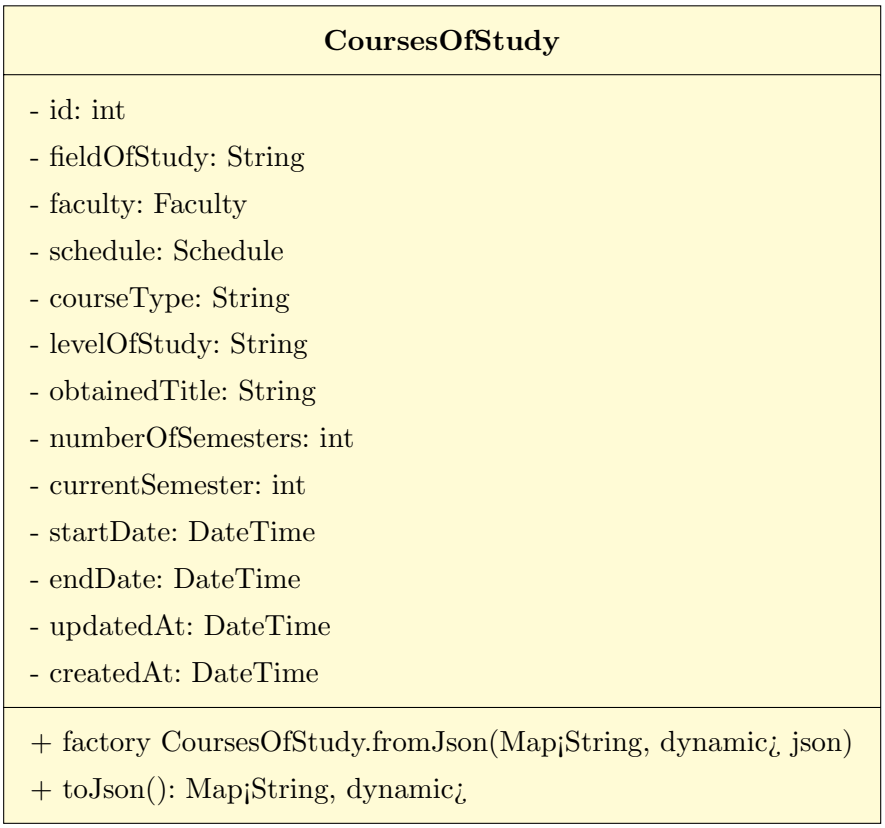
3.9. Diagram UML



Rys. 3.2. Klasa Student



Rys. 3.3. Klasa Schedule



Rys. 3.4. Klasa CoursesOfStudy

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

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

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

Listing 1. Model użytkownika

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

4.2.2. Model wydarzenia

```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 2. 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

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

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

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

Listing 3. 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

```dart
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 4. 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

```dart
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 5. 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

```
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 6. Filtrowanie wydarzeń

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

4.7. 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/messages` - system wiadomości
- `/api/notifications` - zarządzanie powiadomieniami

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

5. Testowanie

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".

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 konkretne zajęcia wyświetli szczegóły, takie jak sala, wykładowca i godziny.

6.3. Powiadomienia

Aby zobaczyć powiadomienia, należy kliknąć na ikonę powiadomień w dolnym pasku nawigacyjnym. Wyświetli się lista powiadomień, takich jak nadchodzące zajęcia, nowe oceny oraz ważne ogłoszenia. Kliknięcie na powiadomienie wyświetli szczegóły.

6.4. Wiadomości

Aby zobaczyć wiadomości, należy kliknąć na ikonę wiadomości w dolnym pasku nawigacyjnym. Wyświetli się lista wiadomości od wykładowców i administracji. Kliknięcie na wiadomość wyświetli jej treść.

6.5. Ustawienia

Aby przejść do ustawień, należy kliknąć na ikonę ustawień w dolnym pasku nawigacyjnym. W ustawieniach użytkownik może:

- Włączyć lub wyłączyć tryb ciemny
- Zarządzać powiadomieniami
- Kliknąć na swój avatar, aby zmienić zdjęcie profilowe, zobaczyć informacje o sobie lub wylogować się

6.5.1. Tryb ciemny

Aby włączyć tryb ciemny, należy przełączyć odpowiedni przełącznik w ustawieniach. Aplikacja automatycznie zmieni motyw na ciemny.

6.5.2. Zarządzanie powiadomieniami

W ustawieniach można włączyć lub wyłączyć powiadomienia dla różnych typów zdarzeń, takich jak nadchodzące zajęcia czy nowe oceny.

6.5.3. Profil użytkownika

Kliknięcie na avatar w ustawieniach przenosi użytkownika do ekranu profilu, gdzie można:

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

6.5.4. Wylogowanie

Aby się wylogować, należy kliknąć na przycisk "Wyloguj się"na ekranie profilu. Użytkownik zostanie wylogowany i przekierowany na ekran logowania.

Spis rysunków

3.1. Architektura systemu 10
3.2. Klasa Student 13
3.3. Klasa Schedule 14
3.4. Klasa CoursesOfStudy 14

Spis tabel

Spis listingów

1. Model użytkownika 15
2. Model wydarzenia 16
3. Provider motywu 16
4. Serwis powiadomień 17
5. Komponent przycisku 17
6. Filtrowanie wydarzeń 18