**Programowanie Systemów Mobilnych II**

**Aplikacja mobilna do obsługi urządzenia IoT**

**Spis treści**

**Rozdział 1**

1. Plan realizacji projektu
   1. Interfejs użytkownika……………………………………...4
   2. Backend aplikacji………………...…….………………….5
   3. Sterowanie płytką ESP……………………………………6
   4. Zabezpieczenia aplikacji oraz responsywność………...6
   5. Testowanie aplikacji……………………………………….7

**Rozdział 2**

1. Lista wykorzystanych technologii oraz bibliotek
   1. Lista wykorzystanych bibliotek……………….…………..8
   2. Lista wykorzystanych technologii………………………...9
   3. System kontroli wersji……………………………………..9

**Rozdział 3**

1. Wybór architektury projektu
   1. Architektura projektu………………………….…………..10

**Rozdział 4**

1. Diagramy oraz schemat bazy danych
   1. Diagram klas……………………………………………….11
   2. Diagram przypadków użycia……………………………..12
   3. Scenariusz użycia…………………………………………13
   4. Schemat bazy danych (relacji + encji)...........................14

**Rozdział 5**

1. Główne interfejsy aplikacji (szkice ekranów) wraz z opisem
   1. Szkic ekran głównego…………………………………..15
   2. Szkic ekran statystyk……………………………………16
   3. Szkic ekran ustawień…………………………………....17

**Rozdział 6**

1. Podział projektu na mniejsze podzadania
   1. Jira - tablica………………………………………………18
   2. Jira - harmonogram……………………………………..20

**Rozdział 1**

**Plan realizacji projektu**

**1.1 Interfejs użytkownika**

**Pierwszym założeniem które chcemy zrealizować, będzie utworzenie dla naszej aplikacji Interfejsu użytkownika**

**Do każdego z ekranów aplikacji zostanie dodany**

* Zostanie dodany **przycisk Home** który będzie wracał do strony startowej
* Zostanie dodany **przycisk Statystyk** który będzie wyświetlał odczyty z czujników naszego fizycznego urządzenia
* Zostanie dodany **przycisk Setting** który będzie przenosił do ustawień aplikacji

* Zostanie dodane **pole header** jak holder dla innych podpól
* Zostanie dodane **pole user** wewnątrz header na którym będzie Witaj, oraz nazwa użytkownika.
* Zostanie dodane **pole data** wewnątrz hedera wyświetlające datę dzień tygodnia i miesiąc

* Zostanie dodany **pole awatar** wewnątrz hedera które będzie wyświetlać awatar lub logo użytkownika

**Do strony startowej aplikacji:**

* Zostanie utworzony holder które będzie obsługiwane przez nasze urządzenie np: Lampka przy biurku ( Holder będzie zawierał nazwę urządzenia oraz przełącznik do włączania lub wyłączania urządzenia)

**Do ekranu odpowiedzialnego za statystyki aplikacji:**

* Zostanie dodane pole przechowujące oraz wyświetlające temperaturę pomieszczenia w postaci stopni celsjusza [ °C ] przekazywaną z fizycznego urządzenia
* Zostanie dodane pole przechowujące oraz wyświetlające wilgotność w postaci procentowej [ % ] przekazywaną z fizycznego urządzenia
* Zostanie dodane pole przechowujące oraz wyświetlające temperaturę urządzenia w postaci stopni celsjusza [ °C ] przekazywaną z fizycznego urządzenia
* Zostanie dodane pole przechowujące oraz wyświetlające natężenie pola magnetycznego w postaci amper na metr [ A/m ]
* Zostanie dodane pole przechowujące oraz wyświetlające siłę sygnału WiFi w postaci decybeli na metr [ -dBm]

**1.2 Backend aplikacji**

**Drugim założeniem realizowanym przez nasz zespół będzie utworzenie dla naszej aplikacji backendu**

* Utworzenie bazy danych z odpowiednimi relacjami oraz encjami
* Zostanie zaimplementowana funkcja która będzie pobierać odczyty z fizycznego urządzenia z bazy danych i zapisywać dane do pola **temperatura pomieszczenia**
* Zostanie zaimplementowana funkcja która będzie pobierać odczyty z fizycznego urządzenia z bazy danych i zapisywać dane do pola **wilgotność**

* Zostanie zaimplementowana funkcja która będzie pobierać odczyty z fizycznego urządzenia z bazy danych i zapisywać dane do pola **temperaturę urządzenia**
* Zostanie zaimplementowana funkcja która będzie pobierać odczyty z fizycznego urządzenia z bazy danych i zapisywać dane do pola **natężenie pola magnetycznego**
* Zostanie zaimplementowana funkcja która będzie pobierać odczyty z fizycznego urządzenia z bazy danych i zapisywać dane do pola **siła sygnału WiFi**
* Zostanie zaimplementowana funkcjonalność **przycisku Home**, aby przekierowywał do strony startowej
* Zostanie zaimplementowana funkcjonalność **przycisku Settings**, aby przekierowywał do strony z ustawieniami aplikacji
* Zostanie zaimplementowana funkcjonalność **przycisku Statystyki**, aby przekierowywał do strony zawierającej dane z czujników
* Zostanie zaimplementowany wewnętrzna funkcja headera który będzie pobierał nazwę użytkownika i wyświetla odpowiedni komunikat wewnątrz **pola user**

* Zostanie zaimplementowany wewnętrzna funkcja headera który będzie pobierał aktualną datę jaki jest to dzień tygodnia oraz jaki jest miesiąc następnie funkcja będzie wyświetlać odpowiednie dane w **polu data**

**1.3 Sterowanie płytką ESP**

**Trzecim założeniem realizowanym przez nasz zespół będzie utworzenie dla naszej aplikacji funkcjonalności odpowiedzialnej za sterowanie płytką ESP która będzie sterować już urządzeniem zewnętrznym z poziomu aplikacji**

* Przygotowanie urządzenia które będzie sterowane przez płytę ESP
* Zostanie dokonana konfiguracja płytki ESP
* Zostanie dokonana konfiguracja serwer na płytce ESP zbierającego dane z otoczenia
* Zostanie zaimplementowana funkcji połączeniowej między serwerem płytki ESP a bazą danych
* Zostanie zaimplementowana funkcji połączeniowej między bazą danych a naszą aplikacją

* Zostanie zaimplementowana funkcji połączeniowej między serwerem płytki ESP a naszą aplikacją
* Zostanie zaimplementowane połączenie zdarzeń serwer z przyciskami w aplikacji mobilnej aby oddziaływały na fizyczne urządzenia ( włączanie i wyłączanie )

**1.4 Zabezpieczenia aplikacji oraz responsywność**

**Czwarty, założeniem realizowanym przez nasz zespół będzie utworzenie dla naszej aplikacji funkcjonalności odpowiedzialnej za zabezpieczenia aplikacji oraz responsywność**

* Zostanie zaimplementowane zabezpieczenia odpowiedzialne za sprawdzenie czy aplikacja ma połączenie z WiFi
* Zostanie zaimplementowane zabezpieczenia odpowiedzialne za sprawdzenie czy aplikacja ma połączenie z Bazą Danych
* Zostanie zaimplementowane zabezpieczenia odpowiedzialne za sprawdzenie czy aplikacja ma połączenie z Płytką ESP
* Zostanie zaimplementowane zabezpieczenia odpowiedzialne za sprawdzenie czy aplikacja przyjmuje odpowiednie dane z bazy danych oraz czy baza danych przyjmuje odpowiednie dane z serwera płytki ESP
* Optymalizacja responsywności aplikacji aby dostosowała się do danych ekranów

**1.5 Testowanie aplikacji**

**Piątym, założeniem realizowanym przez nasz zespół będzie testowanie oraz ewentualne poprawy działania aplikacji**

* Testowanie aplikacji poprzez Espresso lub inną bibliotekę
* Wprowadzanie ewentualnych poprawek do aplikacji i ponowne testowanie

**Rozdział 2**

**Lista wykorzystanych technologii oraz bibliotek**

**2.1 Lista wykorzystanych bibliotek**

* **Android SDK** - zestaw narzędzi, bibliotek i dokumentacji, które są niezbędne do tworzenia aplikacji mobilnych dla systemu Android. Biblioteka ta zawiera narzędzia do budowy interfejsu użytkownika, zarządzania siecią, obsługi danych, multimediów oraz innych funkcjonalności.
* **Retrofit** - biblioteka, która umożliwia tworzenie interfejsów dla komunikacji z serwerem HTTP. Biblioteka ta może być używana do komunikacji z serwerem HTTP

* **Espresso** - framework służący do testowania interfejsu użytkownika aplikacji mobilnej. Framework ten umożliwia pisanie testów, które symulują interakcje użytkownika z aplikacją.
* **Picasso** - biblioteka służąca do ładowania obrazów z sieci. Biblioteka ta może ułatwić ładowanie obrazów na interfejsie użytkownika
* **Mockito** - biblioteka służąca do pisanie testów jednostkowych w języku Java. Biblioteka ta umożliwia tworzenie obiektów testowych, stymulowanie zachowań obiektów zależnych i weryfikację zachowania kodu
* **Room** - biblioteka służąca do tworzenia i obsługi bazy danych SQLite w aplikacjach mobilnych. Biblioteka ta pozwala na łatwe przechowywanie danych i obsługę ich w aplikacji.
* **Firebase** - platforma usług sieciowych Google, która oferuje szereg narzędzi i usług, takich jak autoryzacja użytkowników, baza danych w czasie rzeczywistym, powiadomienia push, analityka i wiele innych.
* **EventBus** - biblioteka służąca do przekazywania zdarzeń między komponentami aplikacji. Biblioteka ta pozwala na łatwe komunikowanie się między różnymi elementami aplikacji.
* **Timber** - biblioteka służąca do logowania w aplikacjach mobilnych. Biblioteka ta umożliwia wygodne logowanie i debugowanie aplikacji.
* **ESPAsyncWebServer.h** - biblioteka do obsługi protokołu HTTP w płytkach ESP. Biblioteka ta może pomóc w budowie serwera HTTP, który będzie obsługiwał zapytania z aplikacji mobilnej i włączał lampkę.
* **WiFi.h** - biblioteka do obsługi modułu Wi-Fi w płytkach ESP. Biblioteka ta umożliwia zestawienie połączenia z siecią Wi-Fi i obsługę komunikacji za pomocą protokołu Wi-Fi.
* **ButterKnife** - biblioteka służąca do wstrzykiwania widoków Androida i innych zasobów w aplikacji mobilnej. Biblioteka ta pozwala na szybsze i wygodniejsze wstrzykiwanie zasobów w kodzie aplikacji.

**2.2 Lista wykorzystanych Technologii**

* **Mikrokontroler ESP32** - płyta ta jest mikrokontrolerem z wbudowanym modułem Wi-Fi, który umożliwia łączenie się z Internetem. Płyta ta będzie pełniła rolę urządzenia IoT, które będzie komunikować się z aplikacją mobilną
* **ESPHome** - system kontroli dla mikrokontrolera ESP32
* **Język programowania aplikacji** -Java
* **Baza danych** - PostgreSQL
* **IDE -** Android Studio
* **Protokoły komunikacyjne** - w projekcie wykorzystywane będą różne protokoły komunikacyjne, takie jak HTTP, MQTT lub CoAP.
* **Urządzenia mobilne** - aby przetestować aplikację, potrzebne będą urządzenia mobilne, takie jak smartfony lub jego emulator.
* **System kontroli wersji Git** - platforma github.com, [repozytorium projektu](https://github.com/ANS-TRN/IoT_App_ANS).
* **Szkice ekranów aplikacji** - [Figma](https://www.figma.com/file/aZUUHZLr8bIIJIxDCbi7jf/Untitled?node-id=0-1).

**Rozdział 3**

**Wybór architektury projektu**

**3.1 Architektura projektu**

Wybraną przez zespół architekturą projektu jest: **Architektura warstwowa**

Ponieważ pozwala na podzielenie aplikacji na kilka warstw, każda z nich pełniąca swoją specyficzną rolę. Na przykład, jedną z warstw może być warstwa prezentacji, która odpowiada za interakcję z użytkownikiem poprzez interfejs graficzny aplikacji. Inną warstwą może być warstwa logiki biznesowej, która odpowiada za przetwarzanie danych i wykonywanie operacji biznesowych.

Architektura czysta jest oparta na zasadzie oddzielenia zależności, co pozwala na większą elastyczność i testowalność aplikacji. W tym podejściu, aplikacja jest podzielona na cztery warstwy:

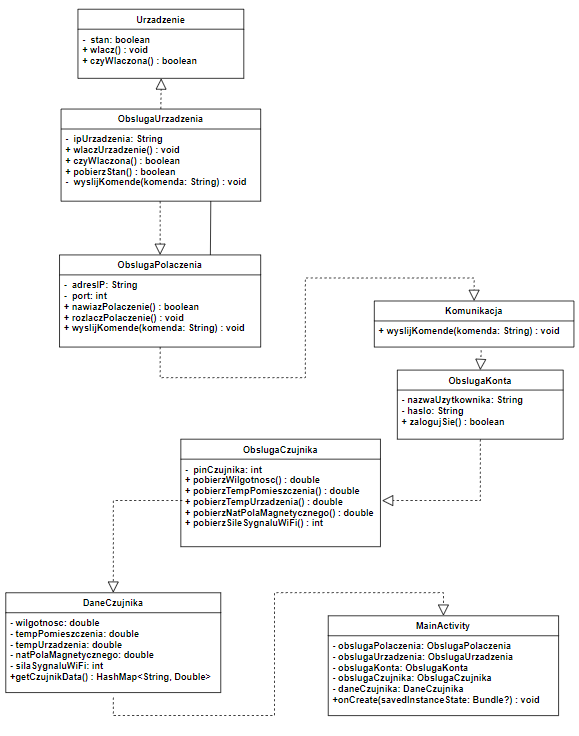
* interfejs użytkownika,
* warstwa aplikacji,
* warstwa domeny
* warstwa infrastruktury.

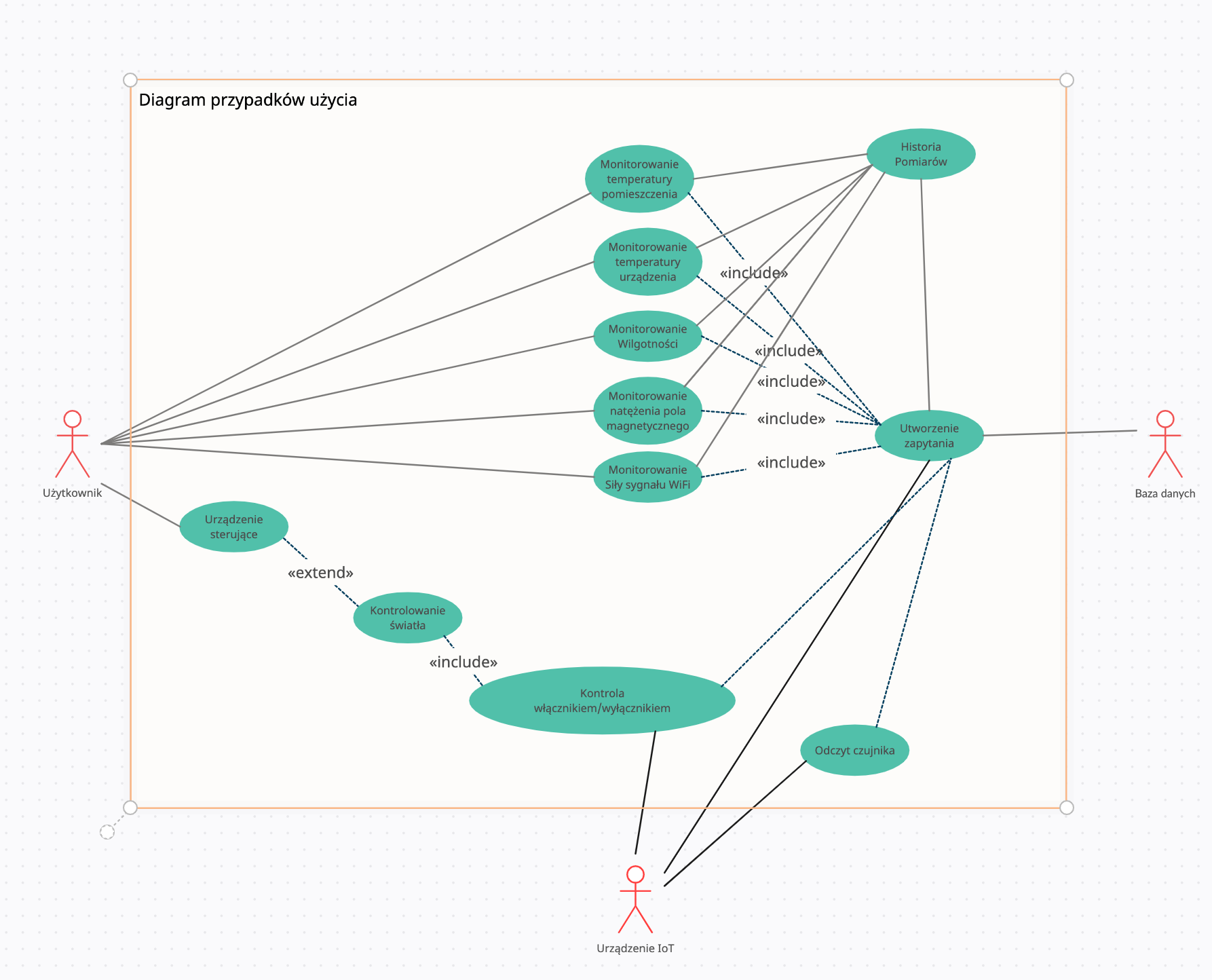
Każda z tych warstw ma swoje specyficzne zadania, co pozwala na lepszą organizację i utrzymanie aplikacji.

**Rozdział 4**

**Diagramy oraz schemat bazy danych**

**4.1 Diagram klas**

****

**4.2 Diagram przypadków użycia**

**4.3 Scenariusz użycia**

Scenariusz użycia do urządzenia smart home zrobionego z ESP32:

1. Włączenie urządzenia

Po włączeniu urządzenia ESP32 podłączonego do sieci WiFi, urządzenie automatycznie nawiąże połączenie z aplikacją na telefonie.

2. Wyświetlanie informacji o temperaturze, wilgotności, natężeniu pola magnetycznego i siły sygnału WiFi

Po nawiązaniu połączenia, aplikacja wyświetli informacje o aktualnej temperaturze urządzenia i pomieszczenia, wilgotności, natężeniu pola magnetycznego oraz siły sygnału WiFi w decybelach na metr.

3. Włączanie i wyłączanie światła

W aplikacji będzie możliwość włączenia i wyłączenia światła w pomieszczeniu. Po wybraniu opcji, ESP32 wyśle sygnał do modułu oświetleniowego, aby włączyć lub wyłączyć światło.

4. Automatyczne sterowanie światłem na podstawie natężenia światła

Urządzenie ESP32 może być wyposażone w czujnik światła, który może mierzyć natężenie światła w pomieszczeniu. Na podstawie tej informacji, ESP32 może automatycznie sterować oświetleniem, aby utrzymać stałe natężenie światła w pomieszczeniu.

5. Konfiguracja parametrów

W aplikacji będzie możliwość konfiguracji różnych parametrów, takich jak minimalne i maksymalne wartości temperatury i wilgotności, próg natężenia pola magnetycznego, który aktywuje alarm, czy też poziom jasności światła, który jest ustawiony automatycznie.

6. Powiadomienia alarmowe

Jeśli natężenie pola magnetycznego przekroczy ustalony próg, ESP32 wyśle powiadomienie do aplikacji na telefonie, aby użytkownik mógł natychmiast zareagować.

7. Włączanie i wyłączanie urządzenia

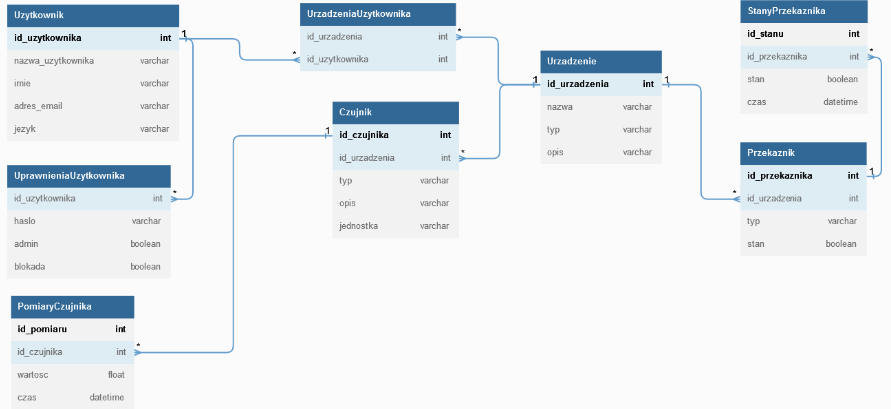
W aplikacji będzie możliwość włączenia i wyłączenia urządzenia ESP32. Po wyłączeniu, ESP32 nie będzie się łączyć z aplikacją na telefonie i nie będzie wysyłać informacji o stanie urządzenia ani oświetlenia.

8. Ustawienia czasowe

W aplikacji będzie możliwość skonfigurowania czasowych ustawień, takich jak włączenie lub wyłączenie oświetlenia o określonej godzinie.

Taki scenariusz użycia pozwoli użytkownikowi na łatwe i wygodne sterowanie urządzeniem smarthome z ESP32 za pomocą aplikacji na telefonie.

**4.4 Schemat bazy danych (relacji + encji)**



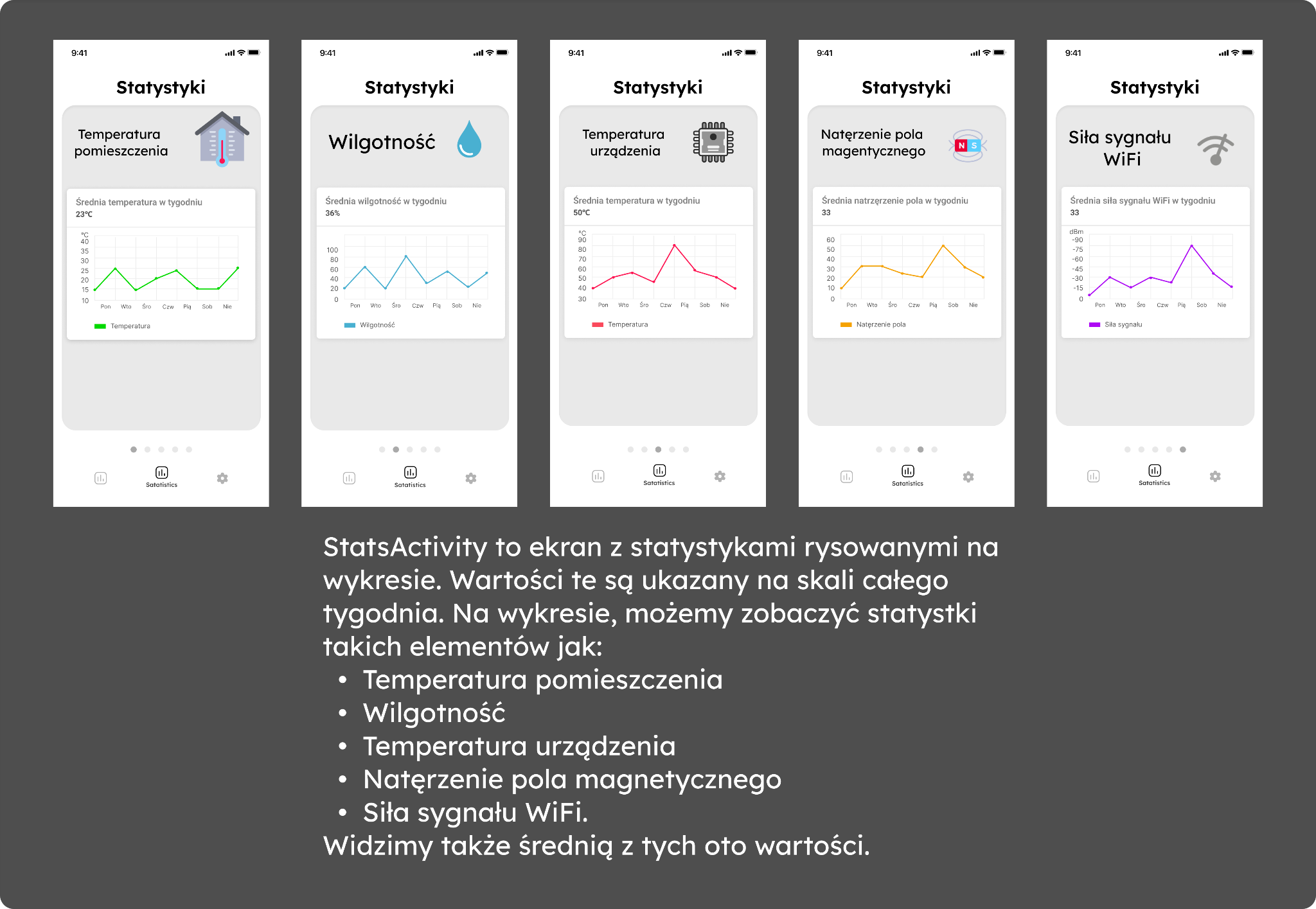
**Rozdział 5**

Główne interfejsy aplikacji (szkice ekranów) wraz z opisem

**5.1 Szkic ekran głównego**

****

**5.2 Szkic ekran statystyki**

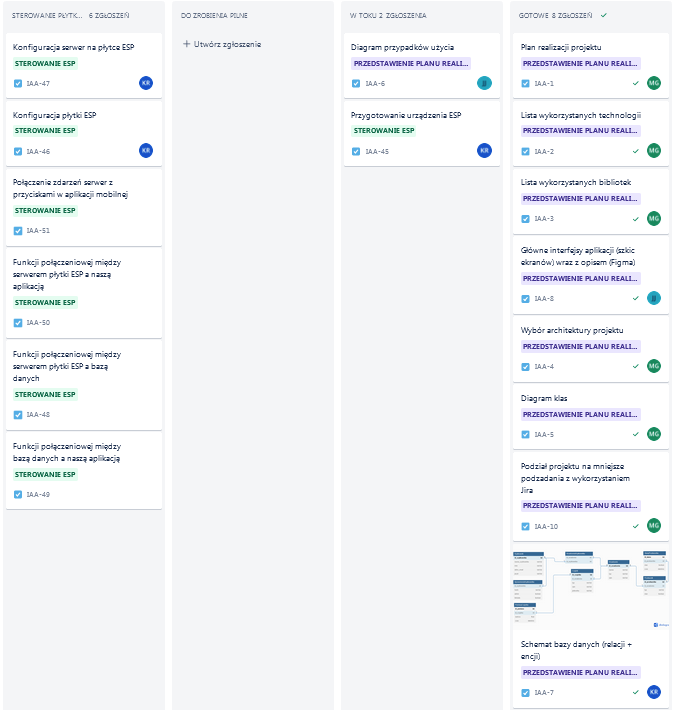


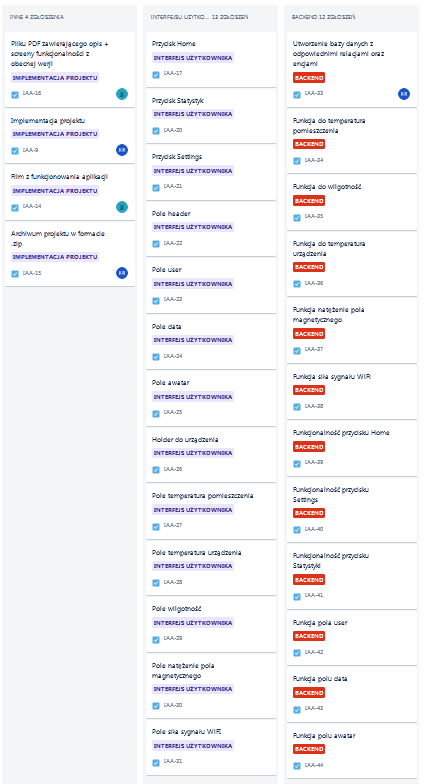
**5.3 Szkic ekran ustawień**

****

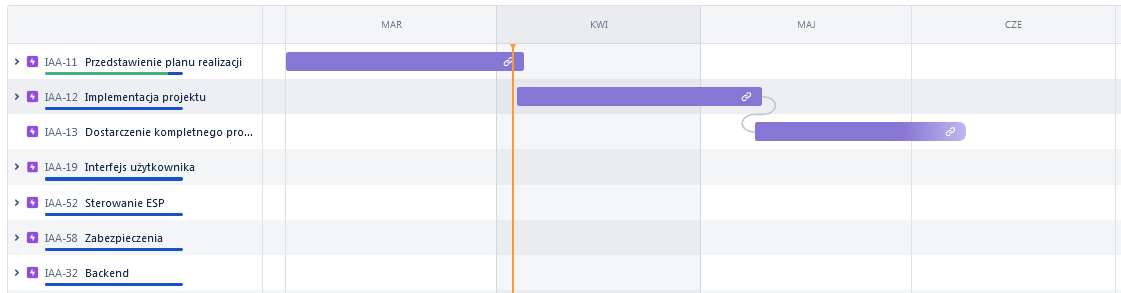
**Rozdział 6**

**6.1 Jira - tablica**

****

****

**6.2 Jira - harmonogram**

****