



**Politechnika  
Śląska**

## **PROJEKT INŻYNIERSKI**

**System zarządzania personelem**

**Piotr MARCOL**

**Nr albumu: 300463**

**Kierunek:** Informatyka

**Specjalność:** Informatyczne Systemy Mobilne i Przemysłowe

**PROWADZĄCY PRACĘ**

**Dr inż. Marcin Połomski**

**KATEDRA Algorytmiki i Oprogramowania**

**Wydział Automatyki, Elektroniki i Informatyki**

**Gliwice 2025**



## **Tytuł pracy**

System zarządzania personelem

## **Streszczenie**

Celem niniejszej pracy było stworzenie systemu zarządzania pracownikami w firmie. Opisano w niej proces, w którym wyznaczono wymagania, wykonano projekt, a na końcu zaimplementowano i przetestowano kompletny system informatyczny. Rozwiązanie składa się z czterech modułów, z których każdy ma swoją ścisłe określona rolę: serwera aplikacyjnego, aplikacji webowej, aplikacji mobilnej oraz układu mikrokontrolerowego. Kolejne części pracy zawierają analizy i szczegóły implementacji projektu. Stworzenie systemu przyniosło wymierne korzyści, a kierunki, jakie można obrać przy jego rozwoju, łączą się z nowymi technologiami, takimi jak sztuczna inteligencja czy Internet Rzeczy.

## **Słowa kluczowe**

System informatyczny, Zarządzanie pracownikami, Harmonogramowanie, Aplikacja bazodanowa, Układ mikrokontrolerowy

## **Thesis title**

Workforce management system

## **Abstract**

The aim of this thesis was to develop a workforce management system for a company. It describes the process by which the requirements were set, the design was carried out, and finally the complete IT system was implemented and tested. The solution consists of four modules, each with its own well-defined role: an application server, a web application, a mobile application, and a microcontroller unit. Subsequent sections of the thesis provide analysis and implementation details of the project. The creation of the system has brought tangible benefits, and the directions that can be taken in its development are linked to new technologies such as artificial intelligence or the Internet of Things.

## **Key words**

IT system, Workforce management, Scheduling, Database application, Microcontroller unit



# Spis treści

<b>1 Wstęp</b>	<b>1</b>
1.1 Wprowadzenie do problematyki . . . . .	1
1.2 Cel pracy . . . . .	2
1.3 Zakres pracy . . . . .	3
1.4 Charakterystyka rozdziałów . . . . .	3
<b>2 Analiza tematu</b>	<b>5</b>
2.1 Przegląd dostępnych sposobów kontroli czasu . . . . .	5
2.1.1 Metody tradycyjne . . . . .	5
2.1.2 Systemy zarządzania kapitałem ludzkim . . . . .	6
2.1.3 Systemy kontroli dostępu . . . . .	7
2.1.4 Podsumowanie . . . . .	7
2.2 Najważniejsze funkcjonalności systemu . . . . .	8
<b>3 Wymagania i narzędzia</b>	<b>9</b>
3.1 Założenia projektowe . . . . .	9
3.1.1 Wymagania funkcjonalne . . . . .	9
3.1.2 Wymagania niefunkcjonalne . . . . .	10
3.2 Projekt systemu . . . . .	10
3.2.1 Technologie . . . . .	10
3.2.2 Narzędzia . . . . .	15
<b>4 Specyfikacja zewnętrzna</b>	<b>17</b>
4.1 Użytkownicy i ich role . . . . .	17
4.1.1 Użytkownik . . . . .	17
4.1.2 Manager . . . . .	17
4.1.3 Administrator . . . . .	18
4.2 Wygląd interfejsu użytkownika . . . . .	18
4.2.1 Kolorystyka . . . . .	19
4.2.2 Struktura widoku . . . . .	20
4.2.3 Strona logowania . . . . .	20

4.2.4	Panel główny . . . . .	21
4.2.5	Widok użytkowników . . . . .	21
4.2.6	Widok karty pracy . . . . .	23
4.2.7	Widok jednostek organizacyjnych . . . . .	24
4.2.8	Widok harmonogramu . . . . .	26
4.2.9	Widok dodawania karty . . . . .	27
4.3	Obsługa czytnika kart . . . . .	28
4.3.1	Uruchomienie . . . . .	28
4.3.2	Odczyt karty . . . . .	29
4.4	Dostępność . . . . .	29
4.4.1	Dostępność cyfrowa . . . . .	29
4.4.2	Dostępność językowa . . . . .	29
<b>5</b>	<b>Specyfikacja wewnętrzna</b>	<b>31</b>
5.1	Architektura systemu . . . . .	31
5.1.1	Aplikacja webowa . . . . .	31
5.1.2	Serwer . . . . .	32
5.1.3	Aplikacja mobilna . . . . .	32
5.1.4	Układ mikroprocesorowy . . . . .	32
5.2	Struktura bazy danych . . . . .	32
5.2.1	Tabele użytkowników . . . . .	33
5.2.2	Tabele jednostek organizacyjnych . . . . .	34
5.2.3	Tabele harmonogramów . . . . .	34
5.2.4	Tabele kart dostępowych . . . . .	35
5.2.5	Tabele słownikowe . . . . .	36
5.3	Modele i struktury danych . . . . .	36
5.3.1	Klasy encji . . . . .	36
5.3.2	Klasy pomocnicze . . . . .	37
5.3.3	Klasy DTO . . . . .	38
5.3.4	Klasy kontrolerów . . . . .	38
5.3.5	Interfejsy i klasy serwisów . . . . .	39
5.3.6	Repozytoria . . . . .	39
5.4	Wybrane algorytmy . . . . .	39
5.4.1	Rejestracja i pierwsze logowanie użytkownika . . . . .	39
5.4.2	Logowanie . . . . .	40
5.4.3	Uwierzytelnianie użytkownika . . . . .	41
5.4.4	Harmonogramowanie . . . . .	41
5.4.5	Przypisanie karty dostępowej . . . . .	43
5.4.6	Uruchomienie czytnika kart . . . . .	44

5.5	Połączenie modułów czytnika kart . . . . .	45
<b>6</b>	<b>Weryfikacja i walidacja</b>	<b>47</b>
6.1	Walidacja danych . . . . .	47
6.1.1	Walidacja po stronie klienta . . . . .	47
6.1.2	Walidacja po stronie serwera . . . . .	48
6.2	Testy . . . . .	48
6.2.1	Testy punktów końcowych API . . . . .	48
6.2.2	Testy czytnika kart zbliżeniowych . . . . .	49
6.2.3	Testy systemowe . . . . .	49
6.2.4	Testy akceptacyjne . . . . .	49
6.3	Bezpieczeństwo . . . . .	49
<b>7</b>	<b>Podsumowanie i wnioski</b>	<b>51</b>
7.1	Wnioski . . . . .	51
7.1.1	Problemy napotkane podczas pracy . . . . .	52
7.1.2	Ocena dobranych technologii po zakończeniu pracy . . . . .	52
7.2	Perspektywy rozwoju . . . . .	52
7.3	Podsumowanie . . . . .	53
<b>Bibliografia</b>		<b>57</b>
<b>Spis skrótów i symboli</b>		<b>61</b>
<b>Lista dodatkowych plików, uzupełniających tekst pracy</b>		<b>63</b>
<b>Spis rysunków</b>		<b>66</b>
<b>Spis tabel</b>		<b>67</b>



# Rozdział 1

## Wstęp

### 1.1 Wprowadzenie do problematyki

W każdym przedsiębiorstwie zatrudniającym pracowników musi zostać ustalona wewnętrzna hierarchia nazywana strukturą organizacyjną. Jest ona oficjalnym podziałem jednostek, komórek, stanowisk i pracowników w organizacji. W nowych przedsiębiorstwach najczęściej występuje nieformalny podział obowiązków, ale z czasem zaczyna klarować się jasny zakres działań poszczególnych osób. Następnie zaczyna się tworzyć struktura organizacyjna, która w pierwszym okresie działania firmy jest spłycona do dwóch poziomów: kierownictwa oraz działów. Taka hierarchia niestety nie sprzyja efektywnemu zarządzaniu, ponieważ obejmuje zbyt szerokie zakresy odpowiedzialności oraz powiązania między działami [6]. W takiej sytuacji konieczne jest wprowadzenie bardziej skomplikowanej struktury organizacyjnej, która umożliwi lepsze zarządzanie. Poniżej wymieniono najczęściej spotykane struktury organizacyjne [46].

- Struktura płaska — występuje w niej centralizacja władzy, w której wszyscy pracownicy podlegają jednemu kierownikowi. Najczęściej występuje w małych lub nowych firmach. Zapewnia szybki przepływ informacji i jest elastyczna.
- Struktura liniowa (prosta) — charakteryzuje się występowaniem trzech rodzajów stanowisk: kierowników, pracowników oraz dyrektorów. Każdy pracownik ma przydzielonego jednego przełożonego, który odpowiada przed swoim dyrektorem. W tej strukturze informacje są przekazywane tzw. drogą służbową. Aby pracownik mógł przekazać informacje do dyrekcji, muszą one przejść przez wszystkie szczeble zarządzania. Ten rodzaj struktury jest najbardziej popularny w przedsiębiorstwach franczyzowych.
- Struktura funkcjonalna — w tej strukturze każdy z pracowników odpowiada przed kilkoma przełożonymi. Pozwala ona zmniejszyć ilość obowiązków pojedynczego kierownika dzieląc je na kilka osób, które są specjalistami w danej dziedzinie. Każdy

specjalista odpowiada przed dyrektorem lub menedżerem, odpowiedzialnym za całość działu.

- Struktura sztabowo-liniowa — jest połączeniem struktury liniowej oraz funkcjonalnej. Zespół składa się z kierownika wspieranego przez kilku specjalistów oraz reszty pracowników. Nie istnieje pojedynczy sposób budowy tej struktury, ponieważ każdy z zespołów może mieć inną strukturę wewnętrzną.
- Struktura macierzowa (problemowa) — struktura rozdziela kierowników na dwa rodzaje: kierowników projektów oraz kierowników działów. Pracownicy podlegają jednocześnie kilku przełożonym, co zwiększa elastyczność firmy. W zespołach mogą powstawać się podgrupy pracowników pracujących nad jednym zadaniem.
- Struktura dywizjonalna — struktura określająca dokładną hierarchię w firmie. Najczęściej spotyka się ją w dużych firmach i korporacjach. Charakteryzuje się wyodrębnieniem działów, które są niemal samodzielne. Każdy z działów może mieć własną strukturę organizacyjną, odpowiednią do swoich potrzeb.

Struktura organizacyjna, jaką należy przyjąć w danym przedsiębiorstwie zależy ściśle od jego specyfiki oraz wielkości, jednakże w każdym przypadku powinna być ona jasno określona, aby umożliwić jak najskuteczniejsze zarządzanie firmą.

Kolejną krytyczną kwestią w zarządzaniu firmą jest kontrola czasu pracy pracowników. Wpływa ona na ich efektywność pracy i zadowolenie. W zależności od wielkości firmy oraz jej specyfiki, metody kontroli czasu pracy mogą się różnić. W firmach gdzie liczba pracowników jest niewielka, kontrola czasu pracy może być prowadzona w sposób tradycyjny; z kolei duże firmy mogą zastosować bardziej zaawansowane systemy informatyczne. Niezależnie od wybranego podejścia, jej celem jest zapewnienie, aby pracownicy byli obecni w miejscu pracy, w określonym czasie. Prowadzenie kontroli czasu pracy jest obowiązkowe dla pracodawców, a nieprzestrzeganie przepisów jest uznawane za naruszenie praw pracowniczych, co może skutkować karą finansową wysokości zgodnej z art. 281 pkt 6 Kodeksu Pracy [49].

## 1.2 Cel pracy

Celem pracy jest opracowanie systemu informatycznego, który wesprze pracę działu kadr w zarządzaniu pracownikami przedsiębiorstwa. Docelowo ma się on składać z czterech głównych części:

- serwera odpowiadającego za część biznesową,
- aplikacji webowej do zarządzania systemem,

- systemu mikroprocesorowego do autoryzacji i przyznania dostępu poprzez odczyt kart zbliżeniowych,
- aplikacji mobilnej umożliwiającej przypisywanie nowych kart dostępowych.

System ma umożliwiać zarządzanie pracownikami, ich zadaniami, czasem pracy i strukturą firmy, a także ma umożliwiać rejestrację czasu pracy pracowników oraz kontrolę dostępu do pomieszczeń. Wykonany projekt - implementując szereg funkcjonalności - ma być narzędziem wspierającym dział kadr automatyzując wiele procesów. Jednocześnie, jego architektura ma umożliwiać łatwą rozbudowę i integrację nowych funkcji w przyszłości.

W trakcie pracy zrealizowano również szereg zadań pobocznych takich jak analiza dziedziny, wyznaczenie wymagań oraz wykonanie projektu graficznego.

## 1.3 Zakres pracy

W ramach pracy przeprowadzono analizę dziedziny, na podstawie której został stworzony projekt systemu. Następnie zostały wyznaczone wymagania funkcjonalne i niefunkcjonalne, które system musi spełniać, oraz wykonany został jego projekt graficzny. Po zakończeniu fazy projektowania rozpoczęto prace nad implementacją. Zakończeniu prac towarzyszyło przeprowadzenie testów oraz ewentualne poprawki.

## 1.4 Charakterystyka rozdziałów

Niniejsza praca inżynierska ukazuje proces tworzenia systemu do zarządzania pracownikami. Jej treść została podzielona na sześć rozdziałów, z których każdy opisuje kluczowe części projektu. Poniżej przedstawiono krótką charakterystykę każdego z nich.

W rozdziale drugim przedstawiono analizę dziedziny oraz przegląd dostępnych rozwiązań. Wskazano w nim zasadność stworzenia systemu oraz rolę, jaką ma on spełniać. Sformułowano i opisano również najważniejsze funkcjonalności, jakie powinien posiadać.

Rozdział trzeci zawiera w sobie szczegółowe dotyczące założień, jakie musi spełniać system, aby był użyteczny. Opisano wymagania funkcjonalne i niefunkcjonalne, a także narzędzia i technologie, które zostały użyte do implementacji systemu.

Rozdział czwarty obejmuje wszelkie informacje dotyczące wyglądu i działania systemu, które są widoczne dla użytkownika. Zostały w nim opisane role użytkowników, dostępne funkcjonalności oraz sposób ich użytkowania. Znajduje się w nim również opis interfejsu graficznego oraz obsługi czytnika kart zbliżeniowych.

W piątym rozdziale zawarto informacje dotyczące implementacji systemu oraz jego architektury. Znajdują się w nim opisy najważniejszych klas, bazy danych, wybranych algorytmów i struktury połączeń układu czytnika kart zbliżeniowych.

W rozdziale szóstym skupiono się na opisie sposobów walidacji danych wejściowych oraz przeprowadzonych testach.

W ostatnim rozdziale przedstawiono wnioski wynikające z przeprowadzonych prac, problemy, jakie w ich trakcie wystąpiły, oraz propozycje możliwości rozwoju systemu w przyszłości.

# Rozdział 2

## Analiza tematu

### 2.1 Przegląd dostępnych sposobów kontroli czasu

#### 2.1.1 Metody tradycyjne

Wiele małych przedsiębiorstw nie potrzebuje wprowadzenia zaawansowanych systemów kontroli pracowników i wciąż korzysta z metod tradycyjnych. Polegają one w większości na ręcznym wypełnianiu papierowych dokumentów, które następnie wymagają przetworzenia. Przykłady takich metod wymieniono poniżej.

- **Indywidualne karty czasu pracy** [16] — pracownik zaznacza swoją obecność na kartce papieru, a następnie podpisuje ją. Zaletą tej metody jest jej prostota i niski koszt, jednakże jest ona mało efektywna i podatna na błędy. Dodatkowo wymagane jest przechowywanie dużej ilości papierowych dokumentów oraz ich archiwizacji.
- **Arkusz kalkulacyjny** — metoda polegająca na prowadzeniu arkusza kalkulacyjnego, w którym - podobnie jak w przypadku kart czasu - zapisywane są godziny pracy pracowników oraz zadania, jakie w tym czasie wykonali. Arkusze pozwalają na jednoczesną kontrolę obecności pracowników oraz na monitorowanie ich postępów w pracy. Niestety, brak automatyzacji procesu skutkuje koniecznością ręcznego wypełniania arkuszy, co zwiększa ryzyko popełnienia błędów. Można wykorzystywać do tego programy takie jak Microsoft Excel czy Google Sheets [48].
- **Harmonogram pracy** — pozwala przełożonym na ustalenie grafiku pracy pracowników. Jest on następnie przekazywany podwładnym, którzy muszą go przestrzegać. Ta metoda wymaga wykorzystania dodatkowych narzędzi, np. kart pracy. Jej największą wadą jest brak możliwości przekazania informacji o zmianach w grafiku w czasie rzeczywistym, co może prowadzić do nieporozumień i konfliktów.

Połączenie kilku tradycyjnych metod pozwala na skutecną kontrolę czasu pracy, lecz bez wykorzystania systemów informatycznych, proces ten jest bardzo czasochłonny i po-

datny na błędy. Korzystanie z papierowych dokumentów jest obarczone bardzo dużym ryzykiem utraty danych, nieautoryzowanego dostępu oraz błędów ludzkich [41].

### 2.1.2 Systemy zarządzania kapitałem ludzkim

Wraz z rozwojem technologii informatycznych wiele firm zdecydowało się na wprowadzenie rozwiązań HCM (ang. *Human Capital Management*). Pozwalają one na pełną automatyzację procesów związanych z działaniem kadr i płac. Oferują szereg funkcji związanych z zarządzaniem czasem pracy, rekrutacją, szkoleniami, wynagrodzeniami oraz rozwojem pracowników. Ich główną częścią jest moduł *Workforce Management* [31]. Zazwyczaj dostęp do systemu odbywa się poprzez aplikację internetową, co umożliwia łatwy dostęp z dowolnego miejsca na świecie. Przykłady dostępnych systemów HCM wymieniono poniżej.

- **Oracle HCM Cloud** [30] — kompletne rozwiązanie chmurowe firmy Oracle, które łączy w sobie funkcje zarządzania personelem, procesami kadrowymi, rekrutacyjnymi i płacowymi. Jest używany m.in. przez FUJIFILM, Deutsche Bahn, Fujitsu.
- **SAP SuccessFactors HCM** [39] — rozwiązanie chmurowe firmy SAP, które oferuje szereg funkcji w zakresie HR (ang. *Human Resources*). Zawiera w sobie moduły do zarządzania procesami kadrowymi, rekrutacyjnymi, szkoleniowymi, płacowymi i analitycznymi. Jest używany m.in. przez Microsoft, Nestlé, Allianz.
- **MintHCM** [24] — oprogramowanie firmy eVolpe oparte na otwartoźródłowych systemach CRM (ang. *Customer Relationship Management*). Oferuje szereg funkcji związanych z zarządzaniem personelem, takich jak: rekrutacja, szkolenia, oceny pracownicze czy zarządzanie czasem pracy i urlopami. Korzystają z niego m.in. Empik, Poczta Polska, Asseco.

Głównym powodem decyzji firm o wdrożeniu systemów zarządzania kapitałem ludzkim jest ich pozytywny wpływ na efektywność pracy i wzrost przychodów. Sprawnie działający system umożliwiający załatwienie wielu spraw w jednym miejscu jest ogromnym udogodnieniem dla pracowników, pozwalając na szybsze reagowanie na zmiany w organizacji i dając jasny wgląd do danych dotyczących ich wydajności. Dla kadry zarządzającej użycie takich systemów daje wymierne korzyści, umożliwiając bardzo szybkie przeglądanie statystyk pracowników i generowanie raportów [25]. Niestety, ich wdrożenie wiąże się najczęściej z ogromnymi kosztami oraz koniecznością przeprowadzenia szkoleń pracowników, co może być problematyczne dla małych przedsiębiorstw.

Często w rozwiązaniach HCM brakuje funkcji związanej z przyznawaniem dostępów oraz kontrolą wejść i wyjść pracowników. W takich przypadkach konieczne jest zintegrowanie ich z systemem kontroli dostępu, co może zwiększać koszta i skomplikowanie

całości. Dodatkowo, systemy HCM są zazwyczaj dostępne jedynie w formie chmurowej, co może okazać się problemem dla firm, które chcą mieć pełną kontrolę nad danymi swoich pracowników.

### 2.1.3 Systemy kontroli dostępu

Celem systemów kontroli dostępu jest zapewnienie bezpieczeństwa w firmie poprzez kontrolę wejścia i wyjścia pracowników oraz gości. Zapewniają one bezpieczeństwo pracownikom oraz ochronę mienia firmy. Pozwalają również na identyfikację osób przemieszczających się po budynku oraz na ograniczenie dostępu do poszczególnych pomieszczeń. Zdarza się, że systemy są zintegrowane z alarmami oraz monitoringiem. Zazwyczaj spotyka się je w dużych firmach, gdzie kontrola dostępu jest kluczowym elementem bezpieczeństwa [20]. Takie systemy dostarczają m. in. firmy:

- **Satel** [40] — polska firma zajmująca się produkcją systemów alarmowych, monitoringowych i kontroli dostępu, której rozwiązania opierają się o technologię RFID (ang. *Radio-Frequency Identification*). Możliwe jest wdrożenie zarówno lokalne, jak i rozproszone,
- **Avigilon** [1] — firma zajmująca się produkcją systemów monitoringu i kontroli dostępu. Ich rozwiązania opierają się głównie na technologiach bezprzewodowych oraz pinpadach.

Systemy kontroli dostępu są zazwyczaj stosowane w firmach, w których bezpieczeństwo jest kluczowym elementem. Dla pracowników ich użytkowanie jest proste i intuicyjne, a dostęp do poszczególnych pomieszczeń jest szybki i wygodny. Niestety, systemy te nie oferują funkcji związanych z zarządzaniem personelem i śledzeniem czasu pracy. W takich przypadkach konieczne jest zintegrowanie ich z systemem HCM, co stanowczo zwiększa koszta i skomplikowanie systemu.

### 2.1.4 Podsumowanie

Analiza dziedziny pozwala stwierdzić, że istnieje zapotrzebowanie na system łączący w sobie funkcje zarządzania personelem, kontroli czasu pracy oraz kontroli dostępu. Obecne rozwiązania są albo nieefektywne i przestarzałe, albo nie zawierają w sobie wszystkich funkcjonalności. W związku z tym zaprojektowanie i wdrożenie nowego systemu, który pozwoli na automatyzację wymienionych procesów, może przynieść wymierne korzyści dla firm. Taki system pozwoli na zwiększenie efektywności pracy, bezpieczeństwa oraz ułatwi pracownikom codzienną pracę. Dodatkowo, wykluczy on koszty ponoszone na utrzymanie kilku systemów oraz zintegrowanie ich ze sobą.

## 2.2 Najważniejsze funkcjonalności systemu

Dobrze zaprojektowany system zarządzania personelem powinien zawierać szereg modułów związanych z jego kluczowymi częściami. Poniżej wymieniono najważniejsze z nich.

- **Zarządzanie pracownikami** — pozwala na przechowywanie danych pracowników i zarządzanie nimi. W systemie powinna być możliwość dodawania, edycji oraz usuwania pracowników, a także przypisywania im odpowiednich ról i uprawnień.
- **Zarządzanie czasem pracy** — pozwala na kontrolę czasu pracy pracowników. System powinien umożliwiać zarządzanie grafikami pracy, kontrolę obecności pracowników oraz monitorowanie ich postępów w pracy.
- **Zarządzanie dostępem** — umożliwia kontrolę dostępu osób do budynku. Moduł powinien umożliwiać zarządzanie kartami dostępowymi, kontrolować wejścia i wyjścia pracowników, a także umożliwiać nadawanie uprawnień dostępu do poszczególnych pomieszczeń.
- **Zarządzanie strukturą firmy** — udostępnia możliwość zarządzania strukturą wewnętrzną przedsiębiorstwa. System powinien pozwalać na tworzenie i zarządzanie działami, zespołami oraz stanowiskami pracy.
- **Zarządzanie wnioskami** — pracownicy powinni mieć możliwość składania wszelkich wniosków w systemie. Moduł powinien umożliwiać zarządzanie wnioskami urlopowymi, zwolnieniami lekarskimi czy wnioskami o zmianę grafiku pracy.

# Rozdział 3

## Wymagania i narzędzia

### 3.1 Założenia projektowe

Podczas tworzenia systemu przyjęto zbiór założeń, które miały na celu określenie zakresu projektu oraz jego funkcjonalności.

#### 3.1.1 Wymagania funkcjonalne

Projekt systemu zakłada spełnienie następujących wymagań funkcjonalnych:

- zarządzanie pracownikami: dodawanie, usuwanie, edycja, przypisywanie ról,
- zarządzanie zadaniami: dodawanie, usuwanie, edycja, przegląd, wymagana akceptacja przez przełożonego,
- zarządzanie czasem pracy: harmonogramowanie czasu pracy,
- zarządzanie strukturą firmy: dodawanie, usuwanie, edycja działów, stanowisk i ról,
- rejestracja czasu pracy: odczyt kart zbliżeniowych, zapisywanie czasu pracy,
- kontrola dostępu: autoryzacja kart zbliżeniowych, przyznawanie dostępu,
- generowanie raportów: raporty z czasu pracy, zadań, działania systemu,
- wnioski: składanie, akceptacja, odrzucanie, komunikacja,
- śledzenie czasu pracy pracowników: rozpoznawanie aktywności pracownika.

### 3.1.2 Wymagania niefunkcjonalne

Projekt systemu zakłada spełnienie wymienionych wymagań niefunkcjonalnych.

- System dostępny jest przez 24 godziny na dobę, 7 dni w tygodniu, z wyjątkiem przerw na konserwację. Nie powinny one przekraczać 1 godziny w tygodniu, a w wypadku konieczności dłuższej przerwy, powinny być wcześniej zapowiedziane. Konserwacja obejmuje prace serwisowe oraz aktualizacje. W wypadku awarii, czas naprawy nie powinien przekraczać 1 dnia od zgłoszenia.
- Aplikacja webowa działa na najpopularniejszych przeglądarkach internetowych: Google Chrome, Mozilla Firefox, Microsoft Edge, Safari itp.
- Aplikacja reaguje na interakcje użytkownika w czasie krótszym niż 0.1 sekundy. Serwer typowo odpowiada na zapytania w czasie nieprzekraczającym jednej sekundy, a w wypadku wzmożonego ruchu nie dłużej niż 5 sekund.
- Serwer jest w stanie obsłużyć do 200 zapytań na sekundę.
- Aplikacja mobilna działa na systemach Android i iOS.
- Komunikacja w systemie jest szyfrowana i bezpieczna.
- System jest odporny na ataki typu SQL Injection oraz Cross-Site Scripting.
- Aplikacja jest intuicyjna i łatwa w obsłudze. Podstawowi użytkownicy powinni móc korzystać z systemu bez konieczności przeprowadzania szkoleń.
- System jest łatwy w utrzymaniu i rozbudowie. Dodanie nowych funkcjonalności nie powinny wpływać na działanie pozostałych.

## 3.2 Projekt systemu

### 3.2.1 Technologie

Poniżej przedstawiono główne technologie wraz z uzasadnieniem ich wyboru oraz spis pozostałych technologii.

#### MySQL

MySQL [27] to relacyjna baza danych rozwijana aktualnie przez firmę Oracle. Jej najważniejszymi cechami są: popularność, szybkość, niezawodność oraz łatwość obsługi. Najnowsze wersje wspierają transakcje, widoki, procedury składowane i wiele innych funkcji, które zbliżają ją do baz danych typu Enterprise. Dodatkowo oferuje lepsze prędkości odczytu danych, niż konkurencyjne rozwiązania takie jak PostgreSQL.

Baza danych MySQL została wybrana ze względu na możliwości jakie oferuje, a także jej przejrzystość, co może ułatwić migrację na inne rozwiązanie w przyszłości.

## Spring Framework

Spring [43] to platforma, która dostarcza rozwiązań do tworzenia aplikacji typu Enterprise w języku Java. Jego asynchroniczna, nieblokująca architektura umożliwia tworzenie rozwiązań obsługujących duże ilości danych, przy jednoczesnym zachowaniu wysokiej wydajności. Składa się z wielu modułów obsługujących kluczowe aspekty działania systemu, takich jak: transakcje, bezpieczeństwo, obsługa danych, integracja z bazami danych, obsługa REST API, itp.

Spring Framework został wybrany ze względu na swoją popularność, wsparcie społeczności oraz bogatą dokumentację, co ułatwiło pracę nad projektem.

## Vue.js

Vue.js [51] jest progresywnym frameworkiem JavaScript przeznaczonym do budowania interfejsów użytkownika. Charakteryzuje się lekkością, prostotą oraz wydajnością. Jego architektura oparta na komponentach umożliwia tworzenie skomplikowanych interfejsów, które są łatwe w utrzymaniu i rozbudowie. Dodatkowo oferuje udostępnienie aplikacji w formie SPA (ang. *Single Page Application*), dzięki czemu użytkownik widzi ją jako pojedynczą stronę internetową. Reakcje na interakcje użytkownika są szybkie i płynne, co zwiększa komfort korzystania, przypominając niejako aplikacje desktopowe.

Vue.js został wybrany ze względu na chęć poznania tej technologii, względnie niski próg wejścia oraz dużą popularność wśród programistów i firm.

## Expo

Expo [7] to platforma, która umożliwia tworzenie wieloplatformowych aplikacji mobilnych w językach JavaScript i TypeScript. Została stworzona na bazie `React Native` i oferuje wiele gotowych rozwiązań dotyczących zarówno interfejsu użytkownika, jak i ustawień aplikacji. Platforma umożliwia również szybkie wdrożenie aplikacji dzięki wbudowanemu serwerowi deweloperskiemu oraz narzędziom do kompilacji i publikacji aplikacji na platformach Android i iOS używając rozwiązania EAS (ang. *Expo Application Service*) [8].

Expo zostało wybrane ze względu na wcześniejsze doświadczenie autora pracy z tą technologią, wieloplatformowość oraz możliwość szybkiego utworzenia i wdrożenia aplikacji na platformy mobilne.

## Raspberry Pi Pico W

Raspberry Pi Pico W [33] jest najmniejszym modułem z rodziny Raspberry Pi, który może łączyć się z siecią. Posiada wbudowany kontroler Raspberry RP4020 oparty na architekturze ARM (ang. *Advanced RISC Machine*), co czyni go idealnym wyborem do zastosowań IoT (ang. *Internet of Things*). Układ wyposażony jest w 264 KB pamięci RAM, 2 MB pamięci flash oraz 26 pinów GPIO (ang. *General Purpose Input/Output*), dzięki którym bez problemu można podłączyć do niego wiele różnych czujników i urządzeń. Pico W posiada wbudowany moduł Wi-Fi oraz Bluetooth, co pozwala na łatwe łączenie się z siecią oraz innymi urządzeniami.

Raspberry Pi Pico W został wybrany ze względu na możliwość połączenia z siecią oraz duże możliwości rozbudowy.

## MicroPython

MicroPython [22] jest implementacją języka Python przeznaczoną dla mikrokontrolerów i systemów wbudowanych. Jest zoptymalizowany pod kątem niskiego zużycia zasobów - co czyni go idealnym wyborem dla urządzeń o ograniczonej mocy obliczeniowej i pamięci. MicroPython oferuje większość funkcji standardowego Pythona, umożliwiając szybkie i efektywne tworzenie oprogramowania dla mikrokontrolerów. Dzięki temu można korzystać z dobrze znanych narzędzi i bibliotek znaczaco przyspieszając proces tworzenia programu i jego testowania.

MicroPython został wybrany ze względu na jego prostotę, elastyczność oraz możliwość szybkiego wdrożenia mikrokontrolera.

## Inne technologie

W projekcie wykorzystano również mniej znaczące technologie, które przedstawiono w tabelach 3.1, 3.2 oraz 3.3.

Tabela 3.1: Biblioteki i frameworki wykorzystane w części backend

Technologia	Opis	Autor	Licencja
Spring Boot	Framework do tworzenia aplikacji w języku Java [42]	Spring	Apache License 2.0
Spring Security	Framework do zarządzania bezpieczeństwem aplikacji [44]	Spring	Apache License 2.0

Springdoc OpenAPI	Biblioteka do generowania dokumentacji oraz dostępu do Swagger-ui [45]	Springdoc	Apache License 2.0
MySQL Connector	Sterownik do łączenia się z bazą danych MySQL [28]	Oracle	GPL 2.0
Lombok	Biblioteka do generowania kodu Java [19]	Project Lombok	MIT
JJWT	Biblioteka do obsługi tokenów JWT [15]	jwtk	Apache License 2.0

Tabela 3.2: Biblioteki wykorzystane w programie układu mikroprocesorowego

Technologia	Opis	Autor	Licencja
mfrc522	Biblioteka do obsługi modułu RFID [32]	Daniel Perron	MIT
picozero	Biblioteka do obsługi modułów peryferyjnych z Raspberry Pi Pico [10]	Raspberry Pi Foundation	MIT
Requests	Biblioteka do wykonywania zapytań HTTP w Python [37]	Python Software Foundation	Apache License 2.0

Tabela 3.3: Biblioteki i frameworki wykorzystane w części frontend oraz aplikacji mobilnej

<b>Technologia</b>	<b>Opis</b>	<b>Autor</b>	<b>Licencja</b>
Axios	Biblioteka do wykonywania zapytań HTTP [2]	Axios	MIT
Heroicons	Zestaw ikon SVG [17]	Tailwind Labs	MIT
PrimeVue	Biblioteka komponentów [36]	PrimeTek	MIT
Luxon	Biblioteka do obsługi dat [26]	Moment.js	MIT
Valibot	Biblioteka do walidacji [12]	Fabian Hiller	MIT
Vue i18n	Biblioteka do obsługi wielojęzyczności [13]	Intlify	MIT
Vue Router	Biblioteka do zarządzania trasami w aplikacji Vue.js [50]	Vue	MIT
TailwindCSS	Biblioteka do stylowania aplikacji internetowych [18]	Tailwind Labs	MIT
Nativewind	Biblioteka umożliwiająca używanie TailwindCSS w aplikacjach opartych o React Native [29]	Nativewind	MIT
React Native Gesture Handler	Biblioteka do obsługi gestów w aplikacjach React Native [21]	Software Mansion	MIT
react native nfc manager	Biblioteka do obsługi NFC w aplikacjach React Native [38]	RevtelTech	MIT

### 3.2.2 Narzędzia

Podczas tworzenia systemu wykorzystano narzędzia, które przedstawiono w tabeli 3.4.

Tabela 3.4: Narzędzia wykorzystane podczas tworzenia systemu

Narzędzie	Opis	Producent
Docker	Narzędzie do uruchamiania kontenerów, w projekcie wykorzystane do uruchomienia serwera bazy danych [5]	Docker Inc.
Git	System kontroli wersji [11]	Linus Torvalds
Visual Studio Code	Edytor kodu wykorzystany przy tworzeniu części frontend oraz aplikacji mobilnej [23]	Microsoft
IntelliJ IDEA	Zintegrowane środowisko programistyczne do języków Java oraz Kotlin, wykorzystane przy tworzeniu części backend [14]	JetBrains
Figma	Narzędzie do projektowania interfejsów użytkownika [9]	Figma
Zeplin	Narzędzie do współpracy nad projektami interfejsów użytkownika, umożliwiające generowanie stylów CSS [52]	Zeplin
Thonny	Zintegrowane środowisko programistyczne dla języka Python na mikrokontrolery [47]	Thonny
PlantUML	Narzędzie do tworzenia diagramów UML [35]	PlantUML



# Rozdział 4

## Specyfikacja zewnętrzna

### 4.1 Użytkownicy i ich role

W systemie zostały zdefiniowane trzy role użytkowników:

- Użytkownik,
- Manager,
- Administrator.

Każda rola posiada inne uprawnienia i możliwości. Do każdej z nich został przypisany kolor, który umożliwia ich łatwe rozróżnienie. Widoczny jest on na pasku nawigacyjnym aplikacji mobilnej i webowej, a także w tabeli użytkowników.

Dodatkowo, każde konto użytkownika może zostać zarchiwizowane, blokując dostęp użytkownika do systemu. Wiąże się to również z usunięciem danych określonych przez ustawę RODO. Poniżej zostały przedstawione informacje na temat każdej z ról.

#### 4.1.1 Użytkownik

**Użytkownik** jest podstawową rolą w systemie i posiada najmniej uprawnień. Ma dostęp wyłącznie do tego, co jest związane z jego kontem i obowiązkami. Może przeglądać swoje dane, harmonogram pracy oraz dodawać wpisy do karty pracy (ang. *Timesheet*).

#### 4.1.2 Manager

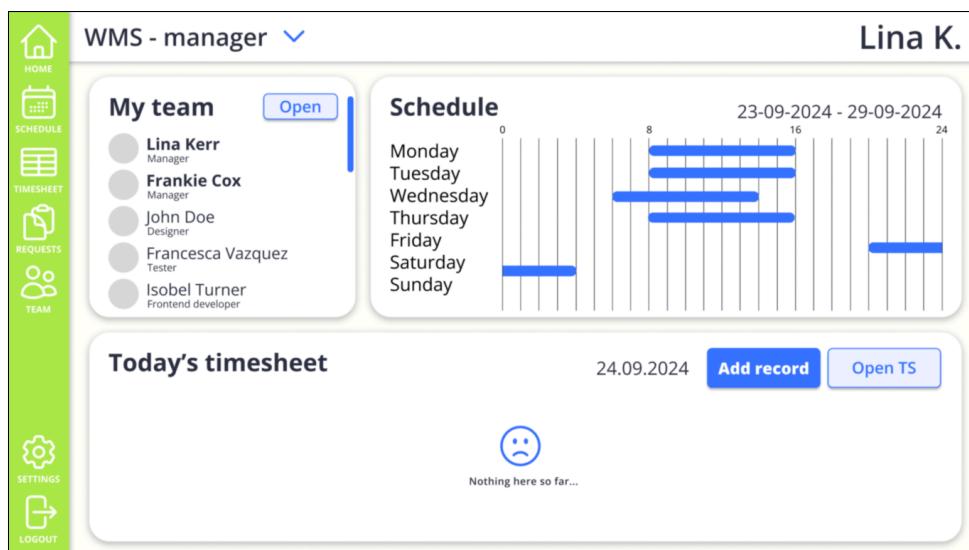
**Manager** posiada większe uprawnienia niż użytkownik. Oprócz możliwości przeglądania swoich danych, harmonogramu pracy, oraz dodawania wpisów do karty pracy, manager może również zarządzać harmonogramem pracy zespołu oraz przeglądać i akceptować karty pracowników. Manager nie ma możliwości zarządzania innymi managerami, administratorami oraz strukturą organizacyjną firmy.

### 4.1.3 Administrator

**Administrator** jest użytkownikiem o największych uprawnieniach w systemie. Rozszerza możliwości managera o możliwość zarządzania poszczególnymi użytkownikami i ich rolami, strukturą organizacyjną firmy - dodawanie, edycja i usuwanie działów oraz stanowisk - oraz konfigurację systemu. Administrator ma dostęp do wszystkich danych i może nimi dowolnie zarządzać.

## 4.2 Wygląd interfejsu użytkownika

Interfejs użytkownika został zaprojektowany w taki sposób, aby osoba korzystająca z systemu mogła od razu zrozumieć, jakie funkcje są dla niej dostępne i co może zrobić. Wszystkie elementy interfejsu, z którymi użytkownik może wchodzić w interakcję, wyróżniają się na tle reszty, co ułatwia ich zauważenie. Skorzystano przy tym z zasady "mniej znaczy więcej", aby nie przytłoczyć użytkownika zbyt dużą ilością informacji. Wszystko, czego użytkownik może potrzebować, jest dostępne w jednym miejscu, a dostęp do mniej znaczących funkcjonalności jest dostępny poprzez dedykowane im widoki. Makieta głównego widoku dla managera została przedstawiona na rysunku 4.1.



Rysunek 4.1: Makieta głównego widoku dla managera

W całej aplikacji zdecydowano użyć jednolitego kroju bezszeryfowej czcionki OpenSans, która została zaprojektowana z myślą o czytelności na ekranach. Ta sama myśl przyświecała przy jej wyborze. Pomimo jej silnej inspiracji krojami klasycznymi, posiada nowoczesny charakter i zachowuje czytelność. Elementy nagłówkowe korzystają z jej wersji pogrubionej lub półpogrubionej. Interfejs użytkownika automatycznie dostosowuje wielkość czcionek do preferencji użytkownika.

Przy projektowaniu interfejsu skorzystano z podejścia Agile UX - metodologii, która zakłada tworzenie wysokiej jakości produktu w sposób iteracyjny. Stąd też interfejs użytkownika miał kilka wersji, zanim osiągnął obecną formę. Wszystkie zmiany były konsultowane z osobami niezwiązanyymi z projektem, aby uzyskać jak największą użyteczność i zadowolenie użytkowników. Przykład zmian w karcie harmonogramu, na widoku głównym został przedstawiony na rysunku 4.2.



Rysunek 4.2: Zmiany w interfejsie użytkownika

Pierwsza wersja - mimo, że bogatsza - była uznana za zbyt skomplikowaną i nieczytelną, zwłaszcza jeśli chodzi o użyte kolory. Wersja końcowa została pozbawiona części elementów, uzyskując w zamian bardziej przejrzysty i czytelny interfejs.

#### 4.2.1 Kolorystyka

Przy projektowaniu interfejsu użytkownika zdecydowano się na użycie ciepłych kolorów, które są przyjemne dla oka i nie męczą wzroku. Wśród nich wybrano trzy, które reprezentują poszczególne role w systemie oraz dodają charakteru interfejsowi. Użyto również kilku odcieni do oznaczania elementów informacyjnych. W aplikacji używane są następujące kolory:

- #FCAB10 — pomarańczowy, kolor użytkownika,
- #ACE849 — zielony, kolor managera,
- #3772FF — niebieski, kolor administratora,
- #FFFFFF — kolor tła,
- #4BB543 — kolor potwierdzenia,
- #DC3545 — kolor błędu,
- #FFC107 — kolor ostrzeżenia,
- #E6EDFF — kolor uzupełniający.

#### 4.2.2 Struktura widoku

Każdy widok aplikacji dla zalogowanego użytkownika składa się z trzech głównych elementów:

- paska nawigacyjnego umieszczonego po lewej stronie ekranu, dzięki któremu użytkownik może szybko i łatwo poruszać się po aplikacji,
- górnej belki z skróconą nazwą użytkownika i jego zespołem,
- głównego obszaru, w którym wyświetlane są poszczególne karty z informacjami.

Makieta struktury widoku została przedstawiona na rysunku 4.3.

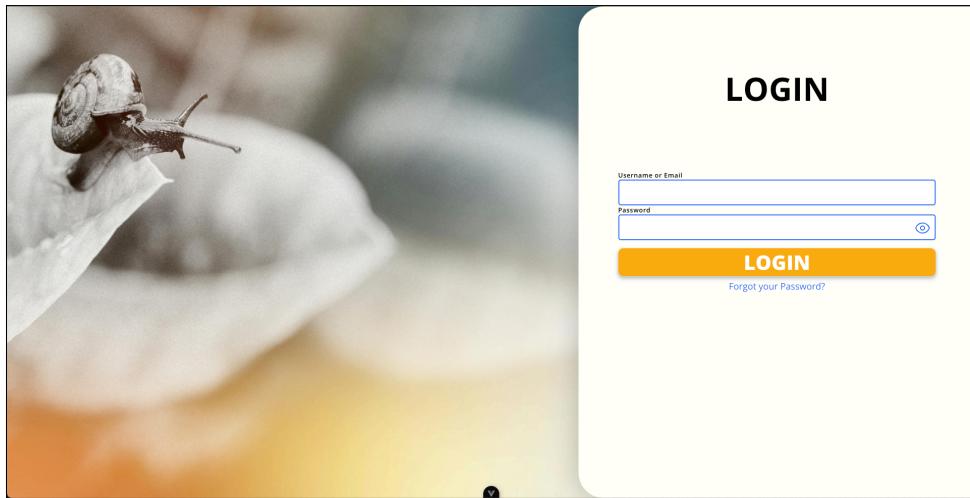


Rysunek 4.3: Makieta struktury widoku

Komponenty są umieszczane na obszarze głównym z wykorzystaniem siatki CSS Grid, o 12 kolumnach i 10 wierszach, dzięki czemu możliwe jest ich łatwe i elastyczne pozycjonowanie na stronie.

#### 4.2.3 Strona logowania

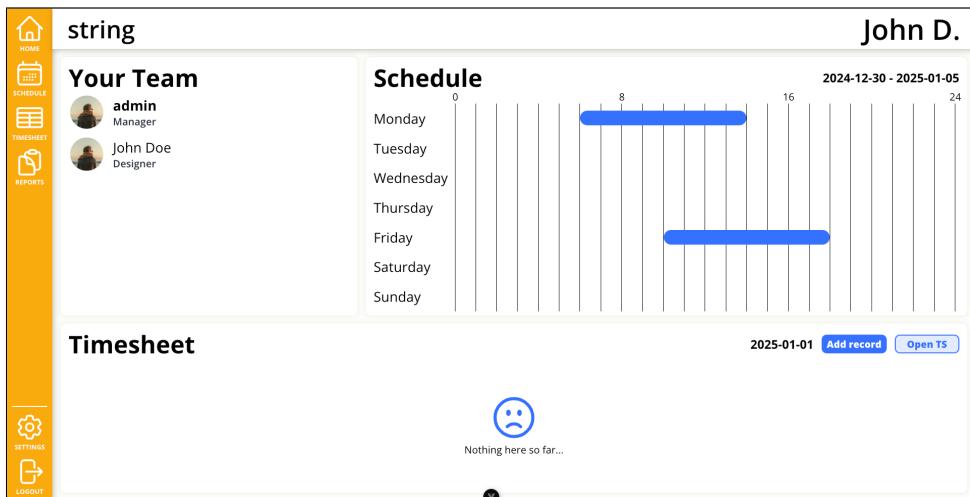
Logowanie do systemu odbywa się poprzez formularz, w którym użytkownik musi podać swoją nazwę użytkownika oraz hasło. W przypadku, gdy użytkownik nie ma jeszcze ustawionego hasła pojawi się dodatkowe pole. Użytkownik musi potwierdzić wcześniej wpisane hasło, a to - jeżeli jest zgodne - zostanie zapamiętane w systemie. Wszelkie błędy związane z logowaniem są wyświetlane w formie komunikatów pod polami formularza. Widok strony logowania został przedstawiony na rysunku 4.4.



Rysunek 4.4: Widok strony logowania

#### 4.2.4 Panel główny

Panel główny jest pierwszym widokiem, który widzi zalogowany użytkownik. Przy projektowaniu zdecydowano, że na panelu głównym znajdują się informacje, które użytkownik będzie najczęściej potrzebował. W związku z tym umieszczono na nim skład aktualnego zespołu, harmonogram pracy w bieżącym tygodniu, oraz szybki dostęp do karty pracy w danym dniu. Przykładowy panel użytkownika został przedstawiony na rysunku 4.5.



Rysunek 4.5: Panel użytkownika

#### 4.2.5 Widok użytkowników

Po przejściu do widoku użytkowników, administratorowi wyświetlana jest lista wszystkich kont zarejestrowanych w systemie. Istnieje możliwość filtrowania wyników wpisując dane użytkownika w pole wyszukiwania. W tabeli znajdują się skrócone informacje o użytkowniku, przedostatnia kolumna przedstawia aktualną rolę użytkownika, a ostat-

nia kolumna umożliwia wykonywanie akcji otwarcia szczegółów, edycji, archiwizacji oraz usunięcia konta. Widoczność poszczególnych akcji zależy od statusu danego konta i jest ustalana przez serwer. Widok użytkowników został przedstawiony na rysunku 4.6. Kliknięcie w ikony akcji otwiera odpowiednie okno dialogowe, szufladę lub wykonuje ją bezpośrednio. Akcje krytyczne z punktu widzenia bezpieczeństwa, wymagają potwierdzenia w osobnym oknie dialogowym.

ID	Username	First name	Last name	Email	Role	Actions	
2	john	John	Doe		USER		
3	admin			admin@admin.pl	ADMIN		
4	manager	manager	manager	man@ger.pl	MANAGER		
5	adam			adam@mowak.pl	USER		
6	Jacek				USER		
7	new				USER		
11	newUser				NEW USER		

Rysunek 4.6: Widok użytkowników

Otwarcie widoku edycji użytkownika umożliwia zmianę jego danych. Obowiązkowe pola są oznaczone gwiazdką, a przesłanie formularza z niepoprawnymi danymi skutkuje wyświetleniem komunikatu o błędzie. W przypadku poprawnego przesłania formularza, użytkownik zostaje poinformowany o sukcesie operacji, okno dialogowe zamknięte, a tabela użytkowników jest odświeżana.

(a) Szczegóły użytkownika

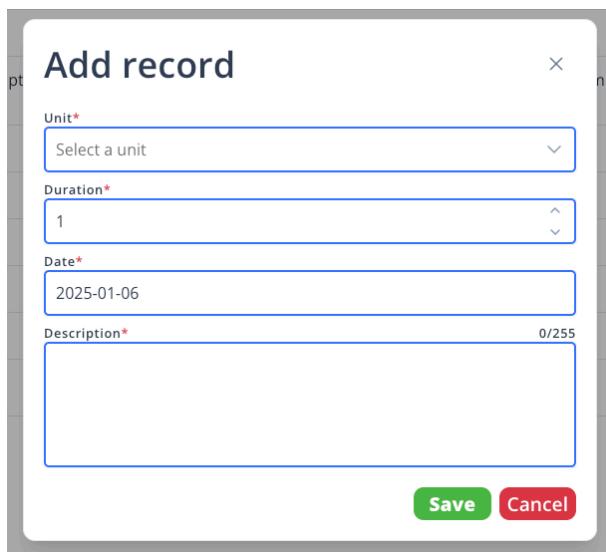
(b) Edycja użytkownika

Rysunek 4.7: Okna dialogowe widoku użytkowników

#### 4.2.6 Widok karty pracy

Widok karty pracy rozszerza funkcjonalności karty z panelu głównego. Oprócz możliwości dodawania wpisów, użytkownik może przeglądać swoje wpisy z wybranego przez siebie zakresu czasu. Informacje są sortowane zgodnie z chronologią ich dodawania. Wspomniany widok został przedstawiony na rysunku 4.8. Dodawanie wpisów odbywa się poprzez formularz otwierany w oknie dialogowym po kliknięciu odpowiedniego przycisku. Został on ukazany na rysunku 4.9.

Rysunek 4.8: Widok karty pracy



Rysunek 4.9: Okno dialogowe dodawania wpisu do karty pracy

Administratorzy i managerowie mają możliwość przejścia do widoku wpisów oczekujących na akceptację. Mogą z tego miejsca ręcznie zatwierdzać lub odrzucać wpisy, lub automatycznie zaakceptować wszystkie z nich. W zależności od statusu wpisu, w tabeli kart pracy pojawiają się odpowiednie oznaczenia. Widok wpisów oczekujących na akceptację został przedstawiony na rysunku 4.10.

The screenshot shows a software interface with a green sidebar on the left containing icons for Home, Schedule, Timesheet, Reports, Users, Units, Settings, and Logout. The main title is 'AIPSUM1' and the section is 'Pending'. A table lists three entries:

Date	Short name	Unit	Duration	Description
2025-01-05	John D.	IPSUM	1	Task #5550
2025-01-05	manager m.	BIPSUM1	1	Task #9987
2025-01-05	manager m.	BIPSUM1	1	Task #9988

On the right, there are buttons for 'Approve all' and 'Show my'. Below them is a table for actions:

Actions
Approved
Rejected

At the bottom, there are navigation arrows and a page number '1'.

Rysunek 4.10: Widok wpisów oczekujących na akceptację

#### 4.2.7 Widok jednostek organizacyjnych

Widok został skonstruowany w taki sposób, aby nie ukazywać wszystkich informacji na raz. Hierarchiczna struktura jednostek organizacyjnych skutkuje wyświetleniem jedynie tych, które są bezpośrednio zależne od wybranej jednostki. Te, które wyświetlają się zaraz po przejściu do widoku nie posiadają żadnych jednostek nadzędnych. Przejście do podjednostek odbywa się poprzez kliknięcie w odpowiednią ikonę w kolumnie akcji. Przejście do wyższego poziomu jest możliwe klikając nazwę jednostki w nagłówku karty. Widok jednostek organizacyjnych przedstawiono na rysunku 4.11.

The screenshot shows a software interface with a blue sidebar on the left containing icons for Home, Schedule, Timesheet, Reports, Users, Units, Settings, and Logout. The main title is 'IPSUM' and the section is 'Units'. A table lists three units:

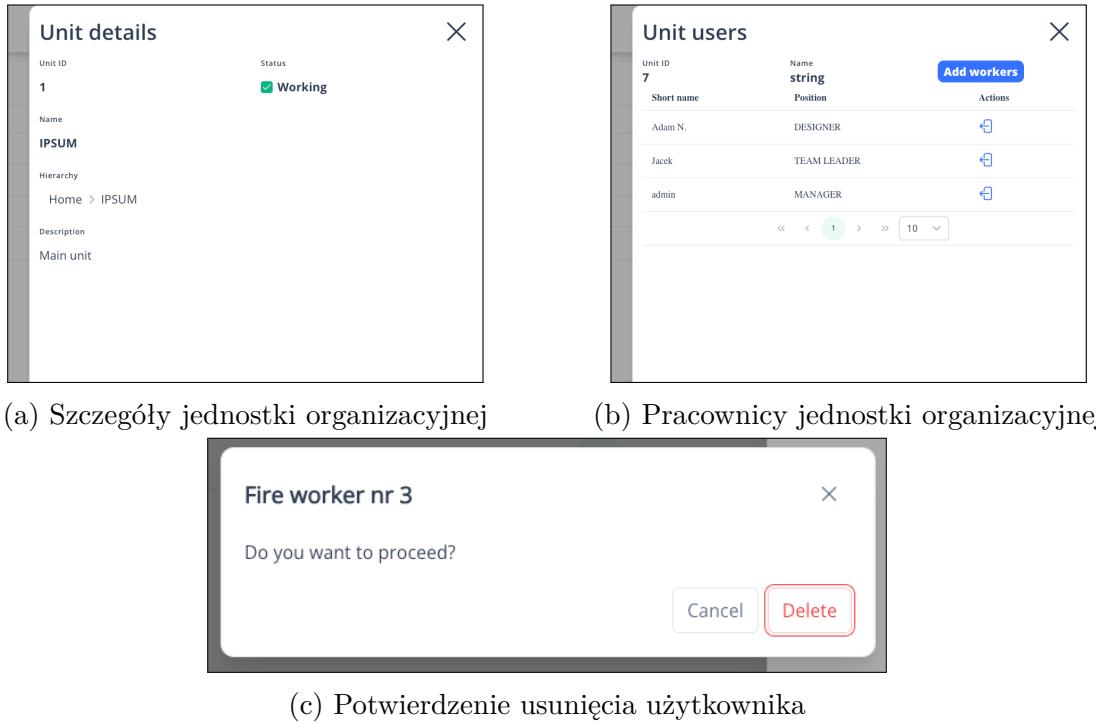
ID	Name	Description	Workers	Subunits	Actions
2	AIPSUM1	Subunit #1	1	1	
3	AIPSUM2	Subunit #2	0	0	
8	ATR1	string	0	0	

On the right, there is a button for 'New unit'. At the bottom, there are navigation arrows and a page number '1'.

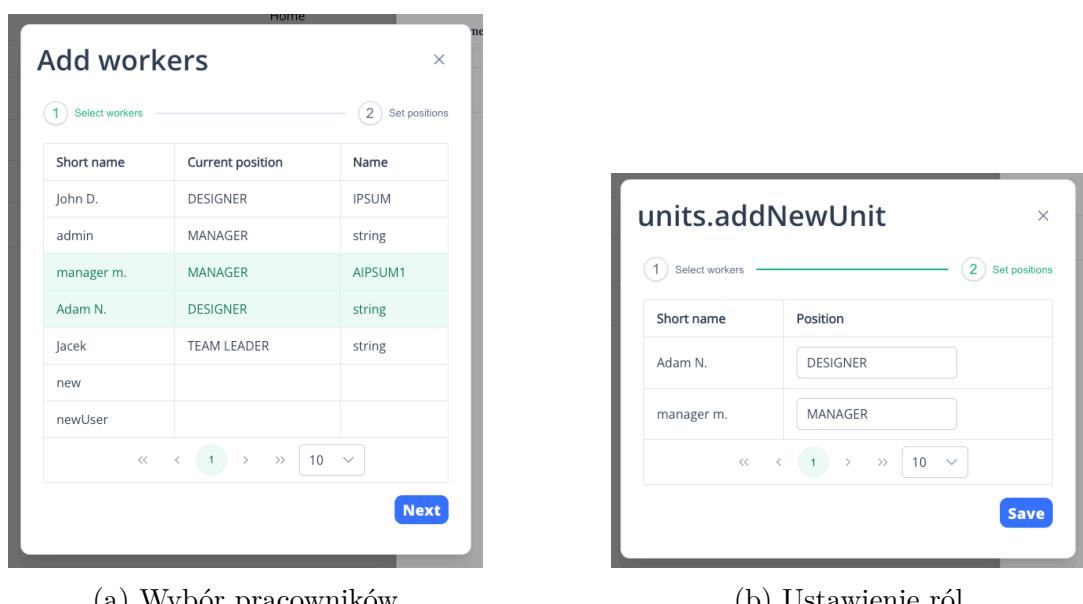
Rysunek 4.11: Widok jednostek organizacyjnych

Kolumna akcji może zawierać maksymalnie cztery ikony: otwarcie szczegółów, edycję, przejście do podjednostek oraz otwarcie widoku pracowników (rys. 4.12). Ostatnia z akcji umożliwia przeglądanie użytkowników pracujących w danej jednostce wraz z ich rolami. W przypadku braku pracowników, tabela jest pusta. Kliknięcie przycisku w górnej części

szuflady otwiera okno dialogowe, w którym możliwe jest dodanie nowych użytkowników do jednostki. Po wybraniu użytkowników aplikacja prosi o nadanie ról, a następnie dodaje ich do jednostki. Po zakończeniu operacji okno dialogowe zamknie się, a tabela jest odświeżana. Otwierane okna są ukazane na rysunku 4.13.



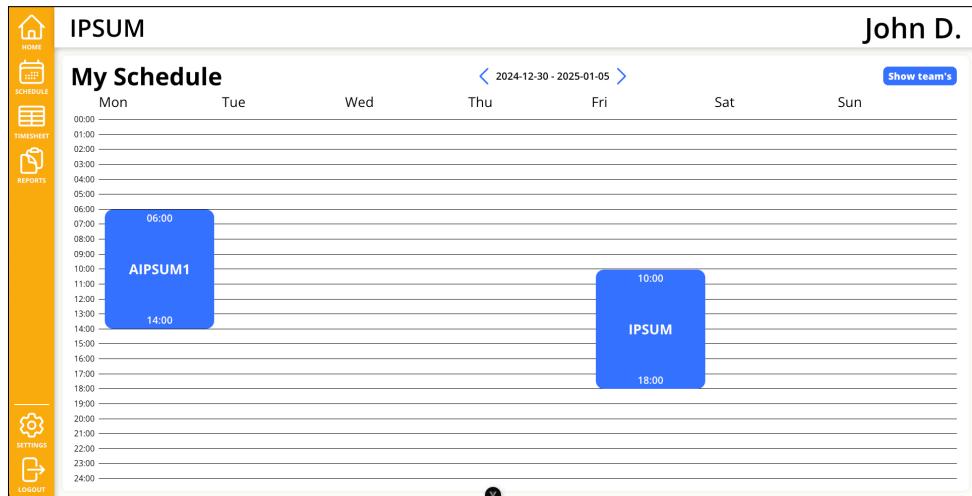
Rysunek 4.12: Okna dialogowe widoku jednostek organizacyjnych



Rysunek 4.13: Okna dialogowe dodawania użytkowników do jednostki organizacyjnej

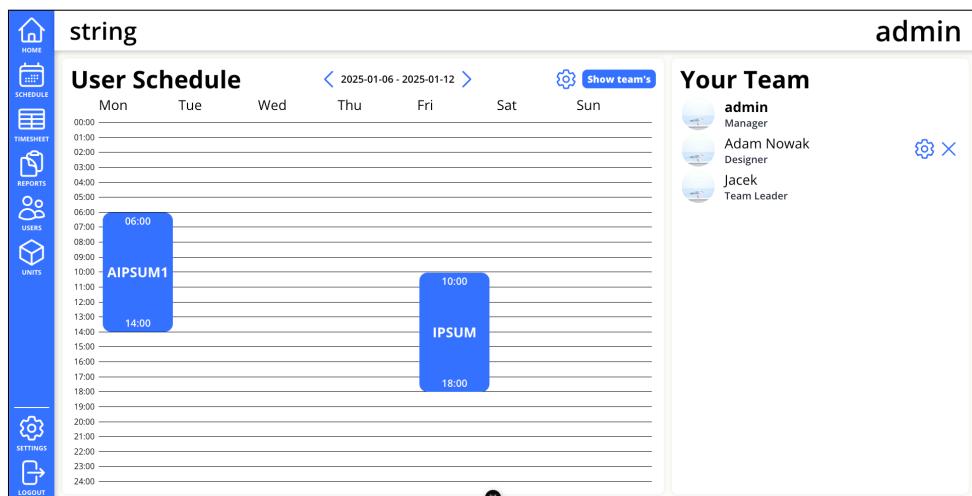
#### 4.2.8 Widok harmonogramu

Widok harmonogramu jest rozszerzoną wersją karty z panelu głównego. Użytkownik może przeglądać harmonogram swój oraz swojego zespołu. Wyświetlany jest w formie siatki, na której umieszczane są bloki reprezentujące poszczególne zadania. Są one przypisane do konkretnych dni i godzin, co umożliwia ich łatwe odczytanie. Dodatkowo, każdy z nich posiada informacje o godzinach rozpoczęcia i zakończenia pracy oraz nazwę zespołu. Widok harmonogramu użytkownika został przedstawiony na rysunku 4.14.



Rysunek 4.14: Widok harmonogramu

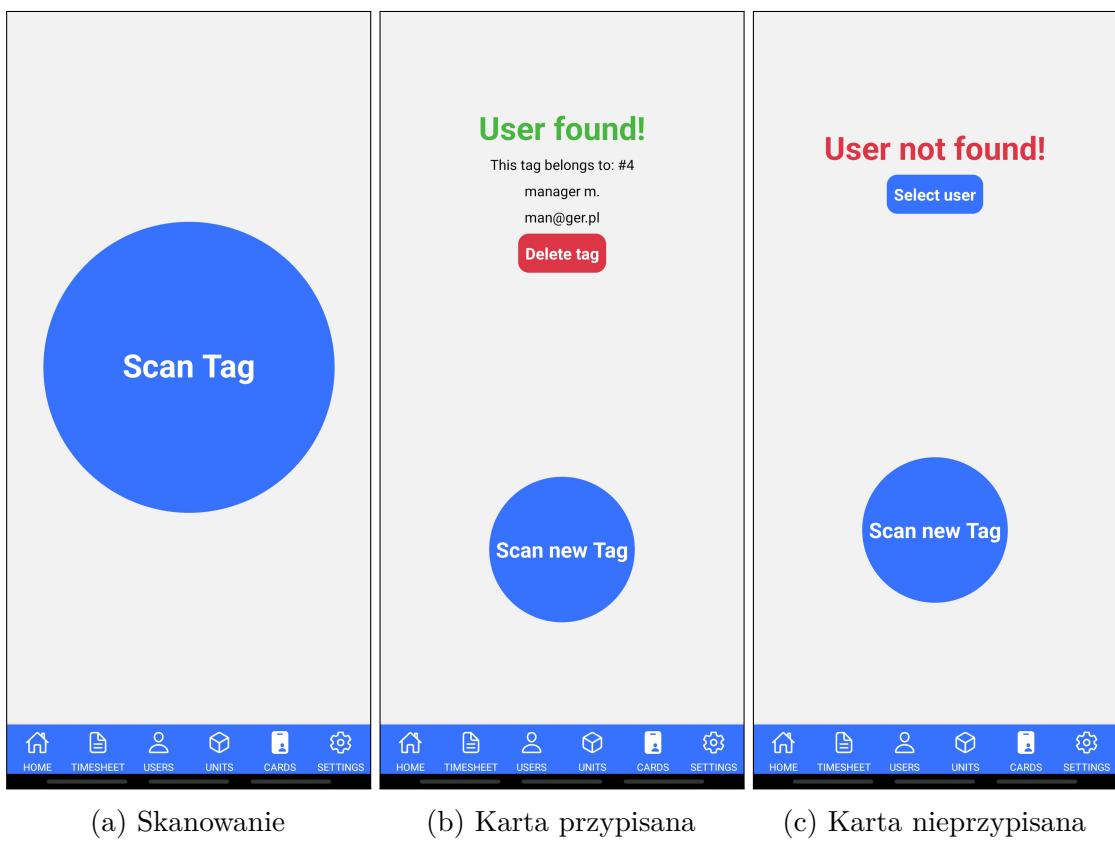
Administratorzy oraz managerowie mają możliwość podglądu harmonogramu użytkowników należących do ich zespołu. Odbiera się to poprzez wybór odpowiedniego użytkownika z dodatkowej karty wyświetlanej po prawej stronie widoku. Ikony kół zębatych umożliwiają edycję harmonogramu. Widok harmonogramu użytkownika przed administratorem został przedstawiony na rysunku 4.15.



Rysunek 4.15: Widok harmonogramu dla managera

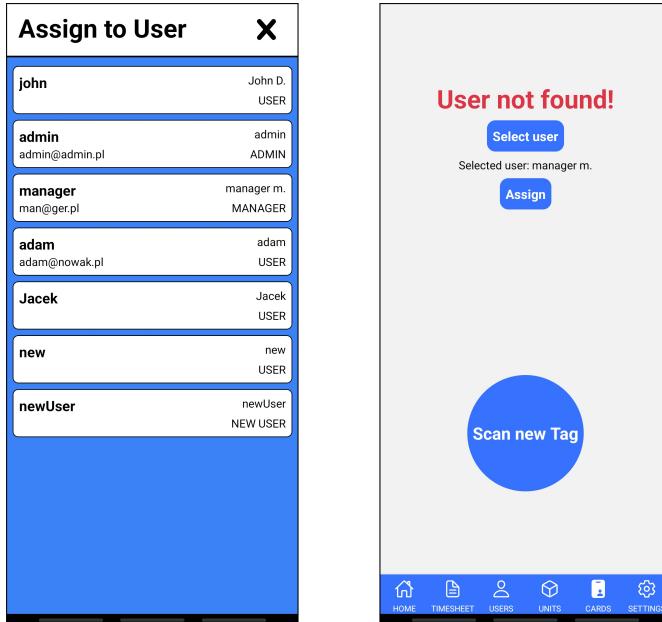
#### 4.2.9 Widok dodawania karty

Karty zbliżeniowe może dodawać administrator wyłącznie za pomocą aplikacji mobilnej. Po przejściu do odpowiedniej zakładki użytkownik musi zeskanować kartę, a następnie aplikacja wysyła do serwera zapytanie na jej temat. Widok jest różny w zależności od tego, czy karta jest już przypisana do użytkownika, czy nie. Jeżeli w urządzeniu jest wyłączona funkcja NFC (ang. *Near Field Communication*), aplikacja wyświetla komunikat o konieczności jej włączenia ukazany na rysunku 4.18a. Widoki zostały przedstawione na rysunkach 4.16.



Rysunek 4.16: Warianty widoku dodawania kart

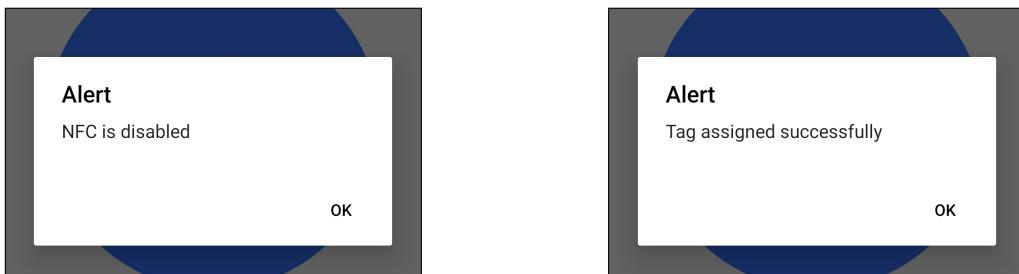
Następnie - w zależności od wyniku zapytania - możliwe jest usunięcie karty z systemu lub przypisanie jej do użytkownika. W przypadku przypisania, administrator musi wybrać użytkownika z rozwijanej listy. Po zatwierdzeniu operacji, karta zostaje przypisana, a administrator otrzymuje informację zwrotną o powodzeniu operacji. Widoki przypisania karty zostały przedstawione na rysunkach 4.17, a komunikat o powodzeniu operacji na rysunku 4.18b.



(a) Lista użytkowników

(b) Wybrany użytkownik

Rysunek 4.17: Widoki dodawania kart



(a) Komunikat o konieczności włączenia NFC

(b) Komunikat o powodzeniu operacji

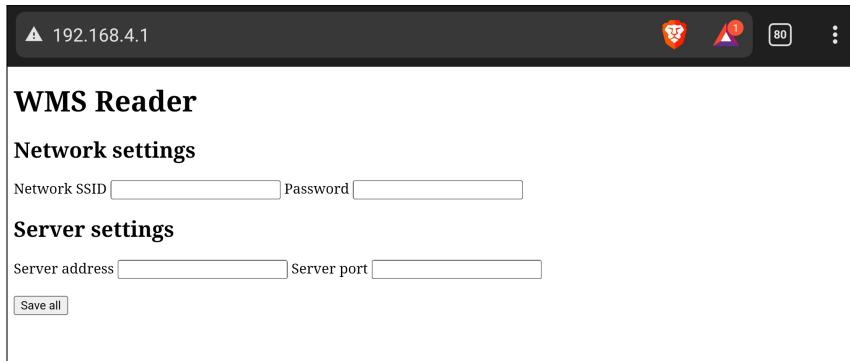
Rysunek 4.18: Komunikaty widoku dodawania kart

## 4.3 Obsługa czytnika kart

Czytnik kart jest prostym układem mikroprocesorowym, który kontaktuje się z serwerem w celu autoryzacji użytkowników.

### 4.3.1 Uruchomienie

Czytnik uruchamia się automatycznie po podłączeniu do zasilania. Podczas tego procesu dioda LED (ang. *Light Emitting Diode*) świeci się na żółto. Jeżeli czytnik napotka jakikolwiek problem podczas uruchamiania, dioda LED zmieni kolor na zielony, a mikrokontroler udostępnii sieć bezprzewodową, do której można się połączyć i otworzyć stronę konfiguracyjną. Widok strony konfiguracyjnej został przedstawiony na rysunku 5.13.



Rysunek 4.19: Widok strony konfiguracyjnej czytnika.

Po poprawnym uruchomieniu, dioda LED zmienia kolor na niebieski.

### 4.3.2 Odczyt karty

Podczas normalnego trybu pracy czytnik oczekuje na zbliżenie karty. Po jej odczycie dioda LED zmienia kolor na żółty, kiedy to czytnik oczekuje na odpowiedź serwera. Następnie przez krótką chwilę świeci na zielono - jeśli odczyt się powódł - lub na czerwono - jeśli wystąpił błąd. Po zakończeniu procesu, czytnik wraca do trybu oczekiwania na odczyt karty.

## 4.4 Dostępność

### 4.4.1 Dostępność cyfrowa

Aplikacja została zaprojektowana w taki sposób, aby możliwe było korzystanie z niej przy użyciu klawiatury. Użycie biblioteki komponentów PrimeVue pozwoliło na zapewnienie dostępności dla osób o ograniczeniach. Komponenty umieszczone na stronie internetowej są zgodne z wytycznymi WCAG 2.1 [4].

### 4.4.2 Dostępność językowa

W celu zapewnienia dostępności systemu użytkownikom z różnych krajów i regionów, aplikacja dostarcza możliwość zmiany języka interfejsu użytkownika. W chwili obecnej dostępne są 2 języki: polski i angielski, jednakże dodanie kolejnego nie wymaga od programisty dużego nakładu pracy. Każdy z języków jest przechowywany w oddzielnym pliku `.json`, co pozwala na jego łatwą modyfikację lub dodanie nowego. Część pliku zawierającego tłumaczenia na język polski została przedstawiona na listingu 4.1.

```
1 "form": {  
2     "save": "Zapisz",  
3     "cancel": "Anuluj",  
4     "fieldRequired": "To pole jest wymagane",  
5     "invalidFormat": "Nieprawidłowy format"  
6 },
```

---

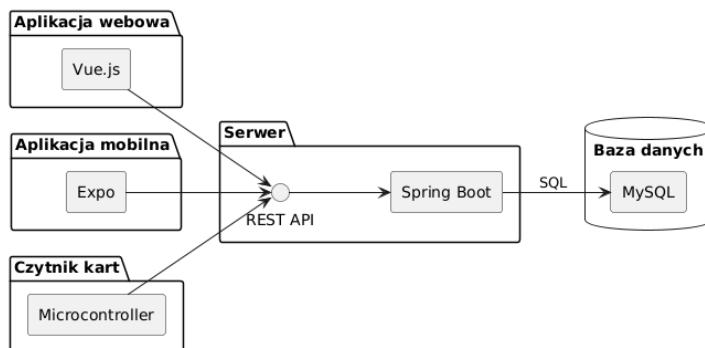
Listing 4.1: Fragment pliku z tłumaczeniami na język polski

# Rozdział 5

## Specyfikacja wewnętrzna

### 5.1 Architektura systemu

Kompletny system składa się z czterech głównych komponentów oraz bazy danych, które komunikują się ze sobą w celu zapewnienia pełnej funkcjonalności. Diagram architektury został przedstawiony na rysunku 5.1. Kolejne podrozdziały opisują każdy z komponentów oraz ich relacje.



Rysunek 5.1: Diagram architektury systemu

#### 5.1.1 Aplikacja webowa

Głównym interfejsem użytkownika jest aplikacja webowa stworzona przy użyciu frameworka `Vue.js`. Zapewnia użytkownikowi dostęp do większości funkcji systemu, a w zależności od jego roli, umożliwia wykonanie innych czynności. Autoryzacja odbywa się poprzez tokeny JWT, które aplikacja przechowuje w pamięci przeglądarki. Wykorzystywana jest biblioteka `Vue Router` odpowiedzialna za ustalanie tras i widoków, a także `Axios` do prostszego wykonywania zapytań do serwera. Frontend komunikuje się z serwerem aplikacyjnym wysyłając żądania HTTP (ang. *Hypertext Transfer Protocol*) i odbierając odpowiedzi w formacie JSON (ang. *JavaScript Object Notation*).

### 5.1.2 Serwer

Serwer aplikacyjny został stworzony przy użyciu frameworka Spring Boot korzystając z architektury Controller-Service-Repository [3]. Zapewnia on komunikację między bazą danych, a pozostałymi komponentami systemu. Jest odpowiedzialny za przetwarzanie żądań klienckich oraz zwracanie odpowiedzi w formacie JSON. Serwer aplikacyjny odpowiada również za autoryzację i autentykację użytkowników oraz zarządzanie sesjami. Użytkownicy nie mają bezpośredniego dostępu do serwera.

### 5.1.3 Aplikacja mobilna

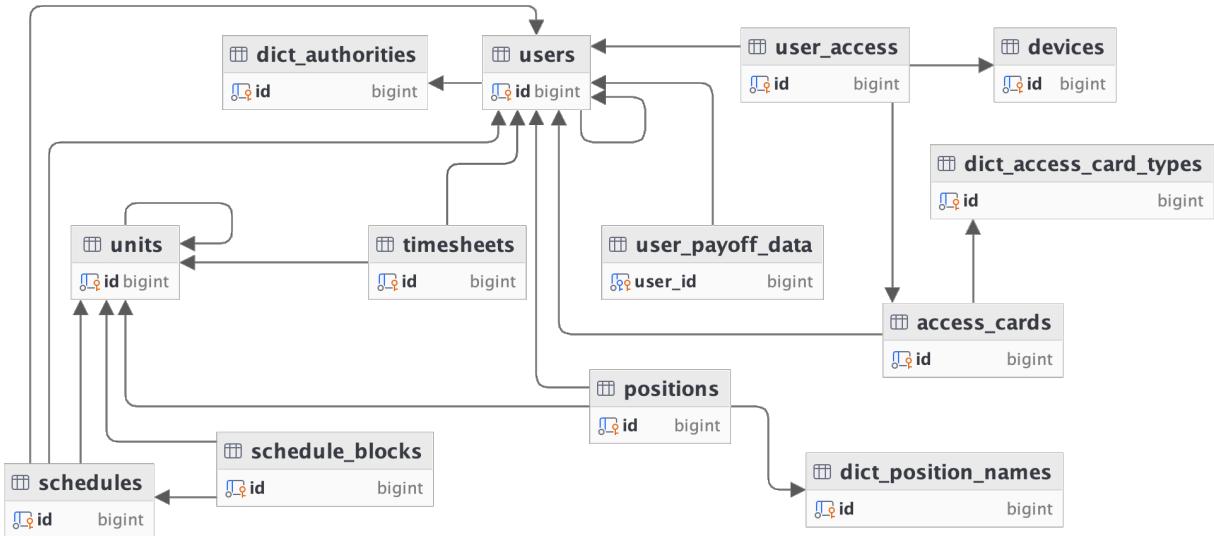
Aplikacja mobilna umożliwia użytkownikom przeglądanie swoich danych bez konieczności korzystania z przeglądarki internetowej. Jej głównym zadaniem jest jednak możliwość dodawania kart dostępowych dla poszczególnych użytkowników. Odbywa się to poprzez przyłożenie tagu NFC do telefonu z zainstalowaną aplikacją, a następnie przypisanie go do konkretnego użytkownika. Szczegółowy opis tego procesu został opisany w rozdziale 5.4.5.

### 5.1.4 Układ mikroprocesorowy

Układ mikroprocesorowy jest odpowiedzialny za odczytywanie tagów NFC oraz przesyłanie informacji do serwera aplikacyjnego. Następnie serwer zwraca informację o przyznaniu dostępu. Układ mikroprocesorowy jest zasilany poprzez wbudowany w płytę port Micro USB, dzięki czemu podłączenie go do zasilania jest bardzo proste. Dokumentacja techniczna mikrokontrolera [34] precyzuje również podłączenie innego źródła zasilania bez użycia portu.

## 5.2 Struktura bazy danych

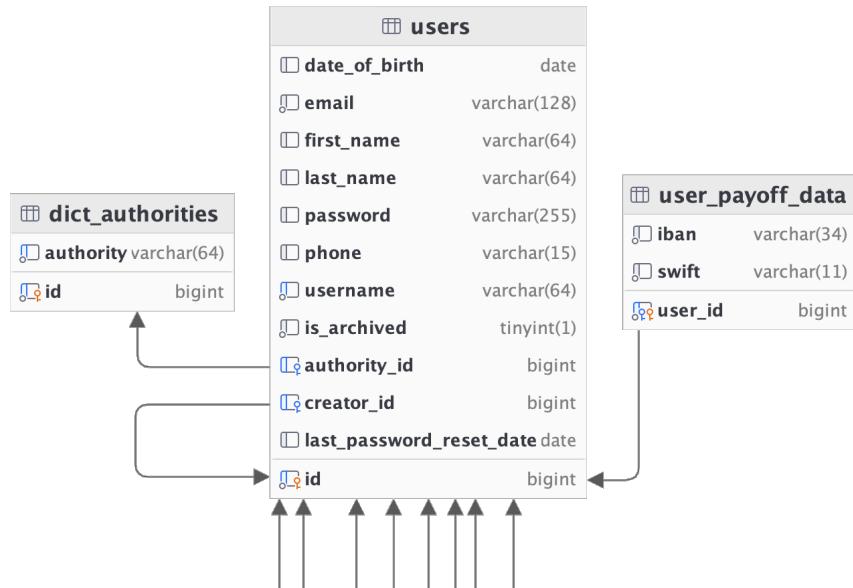
Baza danych obsługująca system składa się z kilkunastu tabel, które przechowują informacje o wszelkich danych w systemie. Na rysunku 5.2 został przedstawiony jej uproszczony schemat, a w kolejnych podrozdziałach zostaną opisane najważniejsze tabele oraz relacje, jakimi są połączone.



Rysunek 5.2: Uproszczony schemat bazy danych

### 5.2.1 Tabele użytkowników

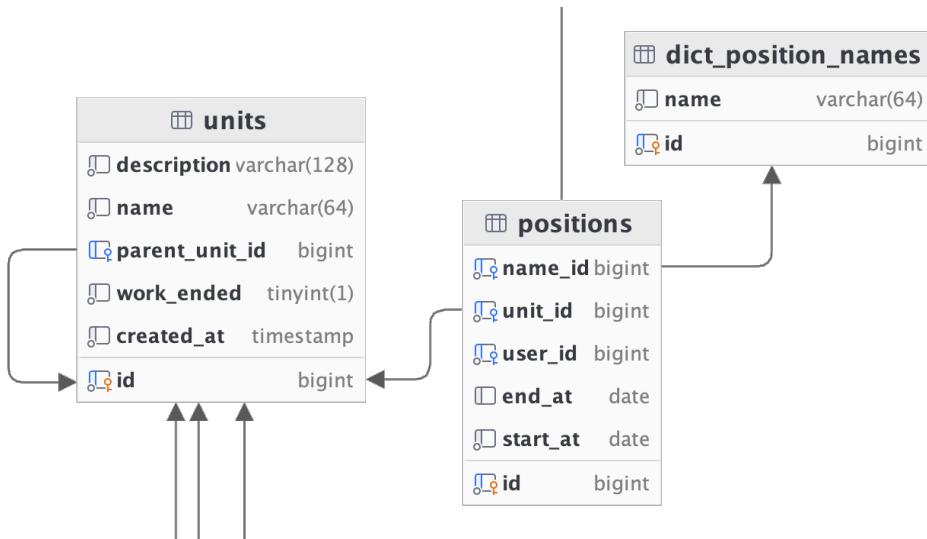
Główna tabelą przechowującą informacje o użytkownikach jest tabela **USERS**. Zawiera ona dane personalne, takie jak imię, nazwisko, adres e-mail i numer telefonu. Dodatkowo do tabeli wpisane są dane do logowania: nazwa użytkownika i hasło; oraz informacja o użytkowniku, który utworzył dany wpis. Każdy użytkownik ma przypisaną jedną z rol z tabeli **DICT\_AUTHORITIES**, która określa jego uprawnienia w systemie. Relacją jeden do jednego jest połączona tabela **USER\_PAYOFF\_DATA** zawierającej dane o koncie bankowym użytkownika. W tabeli **USERS** znajduje się również pole **is\_archived**, które określa, czy użytkownik jest aktywny w systemie. Przedstawienie graficzne tabel użytkowników znajduje się na rysunku 5.3.



Rysunek 5.3: Schemat tabel użytkowników

### 5.2.2 Tabele jednostek organizacyjnych

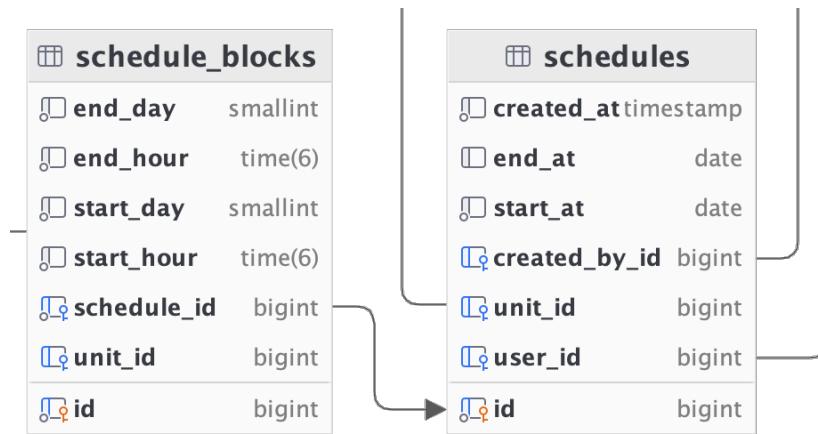
Wszystkie jednostki organizacyjne są przechowywane w tabeli **UNITS** zawierającej dane o nazwie jednostki, jej opisie, jednostce nadzędnej oraz dacie utworzenia. Aktywność jednostki określa pole **work\_ended**. Użytkownicy są przypisani do jednostki organizacyjnej poprzez tabelę **POSITIONS** będącą tabelą łącznikową między tabelami **USERS** i **UNITS**. Znajdują się w niej dane o dacie rozpoczęcia pracy na danym stanowisku oraz dacie zakończenia pracy, a także odwołanie do tabeli słownikowej, zawierającej nazwy stanowisk. Tabela **POSITIONS** jest szczególnie ważna przy odczytywaniu harmonogramów pracy. Schemat tabel jednostek organizacyjnych został przedstawiony na rysunku 5.4.



Rysunek 5.4: Schemat tabel jednostek organizacyjnych

### 5.2.3 Tabele harmonogramów

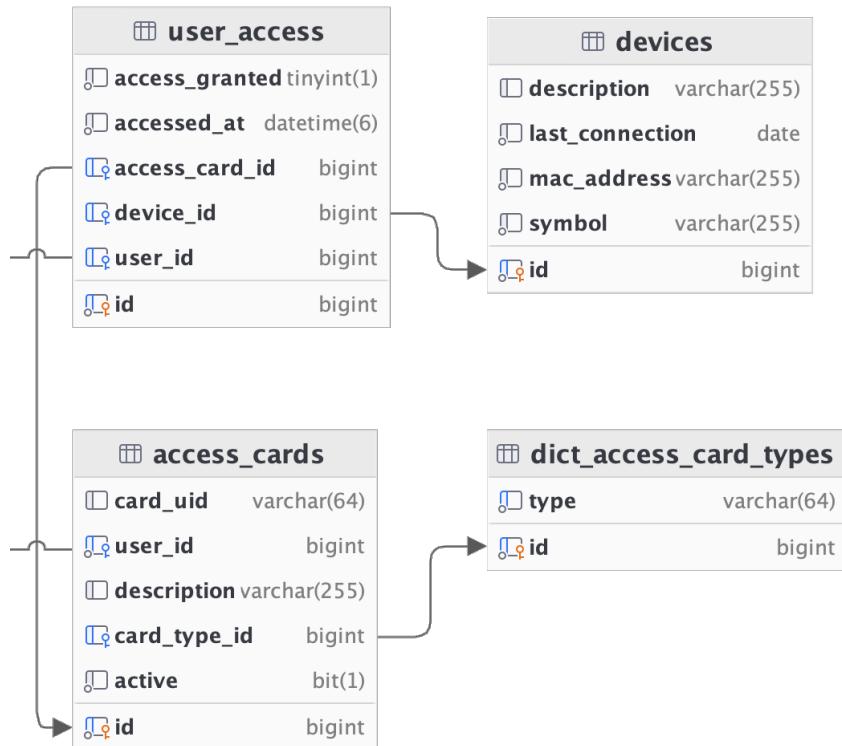
Na każdy z harmonogramów składa się pojedynczy rekord w tabeli **SCHEDULES** i pewna liczba rekordów w tabeli **SCHEDULE\_BLOCKS**. Pierwsza z nich zawiera informacje o dacie rozpoczęcia i zakończenia harmonogramu, jego utworzenia oraz odwołanie do jednostki organizacyjnej lub użytkownika, którego dotyczy. Tabela **SCHEDULE\_BLOCKS** odpowiada za przechowywanie pojedynczych bloków czasowych w harmonogramie opisując jednostkę, której dotyczą, a także ich dzień i godzinę rozpoczęcia oraz zakończenia. Blok nie musi odpowiadać jednostce organizacyjnej, do której przypisany jest cały harmonogram. Tabele są powiązane relacją jeden do wielu - jeden harmonogram może zawierać wiele bloków czasowych. Przedstawienie tabel harmonogramów widoczne jest na rysunku 5.5.



Rysunek 5.5: Schemat tabel harmonogramów

#### 5.2.4 Tabele kart dostępowych

Karty dostępowe użytkowników są przechowywane w tabeli **ACCESS\_CARDS** zawierającej informacje o numerze seryjnym, właścicielu, opisie, typie karty oraz statusie. Dzięki ostatniej z właściwości możliwe jest przypisanie jednej karty dwóm użytkownikom w różnych okresach czasu. Każda autoryzacja użytkownika jest zapisywana w tabeli **USER\_ACCESS** wpisując do niej datę i godzinę, id użytkownika, id czytnika, id karty dostępowej oraz status autoryzacji. Umożliwia to późniejsze analizowanie historii dostępu. Wizualizacja tabel kart dostępowych znajduje się na rysunku 5.6.



Rysunek 5.6: Schemat tabel kart dostępowych

Tabela **DEVICES** przechowuje informacje o czytnikach kart - ich opis, datę ostatniego uruchomienia, adres MAC (ang. *Media Access Control address*) oraz symbol, którym identyfikuje się w systemie.

### 5.2.5 Tabele słownikowe

W bazie danych znajdują się trzy tabele słownikowe zawierające dane, które nie zmieniają się w czasie działania systemu. Nazwa każdej z nich poprzedzona jest prefikssem **DICT\_**. Są to:

- **DICT\_AUTHORITIES** zawierająca role użytkowników, równoznaczne z uprawnieniami,
- **DICT\_ACCESS\_CARD\_TYPES** zawierająca typy kart dostępowych dla łatwiejszego rozróżnienia tagów NFC,
- **DICT\_POSITION\_NAMES** zawierająca nazwy stanowisk przypisanych pracownikom.

Do tabel **DICT\_ACCESS\_CARD\_TYPES** i **DICT\_POSITION\_NAMES** mogą zostać dodane nowe rekordy, lecz nie jest możliwe ich usunięcie. Takie ograniczenie zapewnia integralność danych w systemie i zapobiega błędem w działaniu aplikacji.

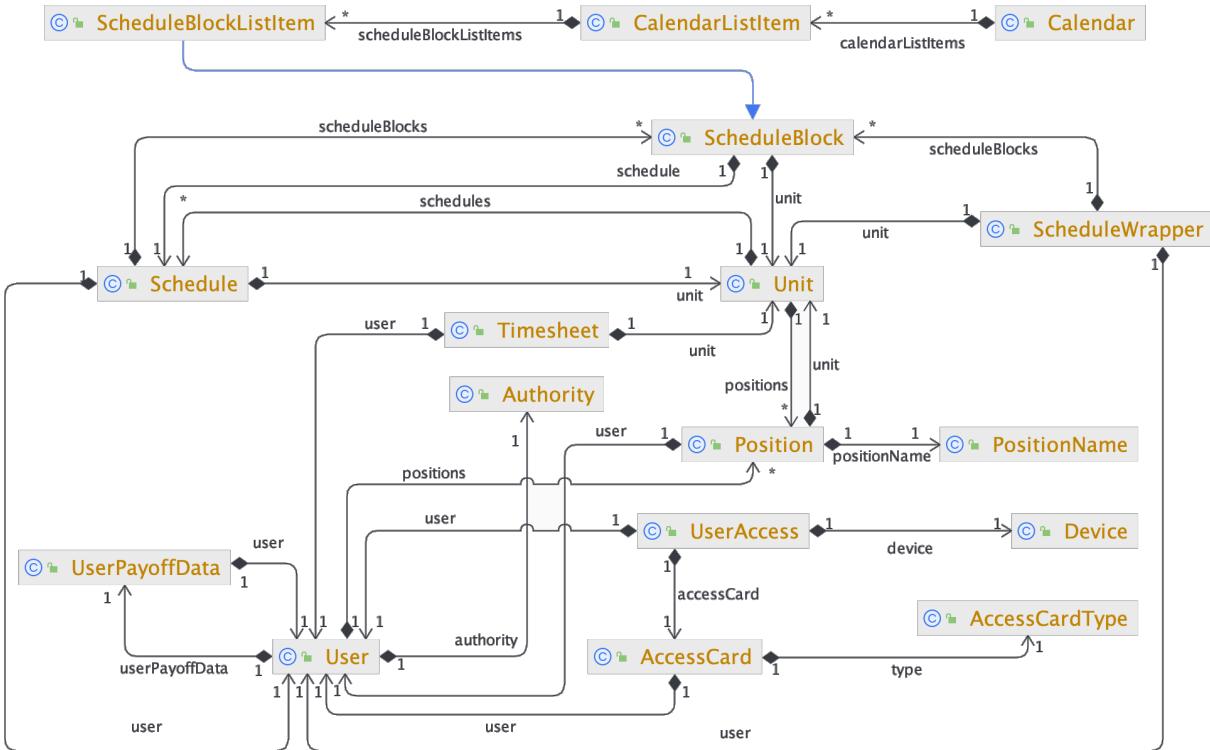
Dane mogą dodawać jedynie administratorzy systemu.

## 5.3 Modele i struktury danych

Dzięki użyciu w projekcie JPA (ang. *Java Persistence API*) oraz Hibernate, struktury danych w systemie odpowiadają strukturom tabel w bazie danych - każda z tabel bazy danych jest mapowana na odpowiadającą jej klasę w systemie. Oprócz tego zostały zaimplementowane klasy pomocnicze oraz klasy DTO (ang. *Data Transfer Object*), które upraszczają logikę biznesową. Diagram klas został przedstawiony na rysunku 5.7.

### 5.3.1 Klasy encji

Zimportowane zależności w projekcie umożliwiają tworzenie klas encji, które są mapowane na tabele w bazie danych. Każda z nich posiada adnotację **@Entity** - informującą JPA o tym, że klasa jest encją - oraz **@Table** z nazwą tabeli, z którą jest mapowana. Każde pole klasy, które odpowiada kolumnie w tabeli jest odpowiednio oznaczone. Do generowania metod **get** i **set** używane są adnotacje **@Getter** i **@Setter** z biblioteki **lombok**. W klasach znajdują się również proste, publiczne metody przetwarzające dane, dzięki którym możliwe było uproszczenie logiki biznesowej.



Rysunek 5.7: Diagram klas modeli

### 5.3.2 Klasy pomocnicze

Aby zapewnić poprawne działanie systemu, niezbędne było zaimplementowanie klas pomocniczych. Nie są one mapowane na tabele w bazie danych, a ich celem jest uproszczenie logiki biznesowej oraz zwiększenie czytelności kodu. Dzielą się na dwie kategorie: narzędziowe oraz danych.

Klasy należące do pierwszej z kategorii zawierają jedynie metody statyczne i nie wymagają tworzenia instancji ich obiektu. Istnieją dwie takie klasy:

- **DateUtils** — zawierająca metody do zmiany dat z formatu `java.time.LocalDate` na `java.util.Date` oraz `java.sql.Date`,
- **JwtTokenUtils** — zawierająca metody do generowania, weryfikacji i wyłuskiwania danych z tokenów JWT (ang. *JSON Web Token*).

Druga kategoria klas pomocniczych to klasy danych, które istnieją w celu uproszczenia przechowywania danych w systemie. Służą najczęściej jako kontenery na dane, które następnie zostaną przetworzone na obiekty klas DTO. Najważniejszą z nich jest klasa **Calendar**, reprezentująca harmonogramy w formie kalendarza. Przechowuje ona dane dla każdego dnia tygodnia w ustalonym przedziale czasowym i dodatkowo posiada metody do sortowania, nadpisywania dni oraz uzupełniania pustych miejsc w kalendarzu.

### 5.3.3 Klasy DTO

W pakiecie `apimodels`, odrębnym od głównego pakietu serwera znajdują się definicje klas DTO. Służą one do ustalania struktury obiektów przesyłanych między serwerem, a klientem, zapewniając poprawne działanie systemu. Każda z nich jest zaimplementowana jako klasa Java, która posiada jedynie publiczne pola dostępowe. Nie jest na nich wykonywana żadna logika biznesowa, a dane są wpisywane bezpośrednio przed przesaniem ich do odbiorcy.

Pakiet `apimodels` zawiera subpakiety odpowiadające kontrolerom, które wymagają przesłania pełnego obiektu. Są to:

- `access_card` — modele kart dostępowych,
- `auth` — modele autoryzacji i autentykacji,
- `position` — modele stanowisk,
- `schedule` — modele harmonogramów,
- `timesheet` — modele kart pracy,
- `unit` — modele jednostek organizacyjnych,
- `user` — modele użytkowników.

### 5.3.4 Klasy kontrolerów

Klasy kontrolerów odpowiadają za obsługę żądań HTTP przesyłanych do serwera. Realizują one komunikację między klientem a serwerem wywołując odpowiednie metody i zwracając dane. Każda została oznaczona adnotacją `@RestController` oraz przypisano jej ścieżkę. Nie zawierają one logiki biznesowej, a jedynie wywołania odpowiednich metod serwisów. Przykład metody kontrolera został przedstawiony na listingu 5.1.

---

```
1 @PostMapping("/add")
2 public ResponseEntity<?> addUnit(@RequestBody AddUnitRequest request) {
3     try {
4         unitService.addUnit(request);
5         return ResponseEntity.ok("Unit added successfully");
6     } catch (Exception e) {
7         return ResponseEntity.badRequest().body("An error occurred");
8     }
9 }
```

---

Listing 5.1: Metoda kontrolera jednostek organizacyjnych dodająca nową jednostkę

### 5.3.5 Interfejsy i klasy serwisów

Serwisy są odpowiedzialne za logikę biznesową systemu. Każda z klas serwisów implementuje jego interfejs umożliwiając stworzenie kilku wariacji rozwiązania problemu. Dzięki temu możliwe jest łatwe dodanie nowych funkcji do systemu bez konieczności zmiany istniejącego już kodu. Metody interfejsów są publiczne, a niektóre klasy zawierają metody prywatne służące do przetwarzania danych. Adnotacją informującą platformę Spring o tym, że klasa jest serwisem jest `@Service`.

### 5.3.6 Repozytoria

Częścią, która odpowiada za komunikację pomiędzy serwerem, a bazą danych są repozytoria. Są one interfejsami rozszerzającymi interfejs `JpaRepository` udostępniany przez Spring Data JPA. Metody w nich zawarte są nazywane zgodnie z konwencją określoną przez twórców, co umożliwia automatyczne generowanie zapytań SQL. Kod repozytorium harmonogramów ukazano na listingu 5.2.

---

```

1 public interface iScheduleRepository extends JpaRepository<Schedule,
2     ↳ Long> {
3     List<Schedule> findAllByUnitId(Long unitId);
4     Schedule findFirstByUnitIdOrderByStartDateDesc(Long unitId);
5     Schedule findFirstByUserIdOrderByStartDateDesc(Long userId);
6     List<Schedule> findAllByUserId(Long id);
7 }
```

---

Listing 5.2: Przykład interfejsu repozytorium harmonogramów

## 5.4 Wybrane algorytmy

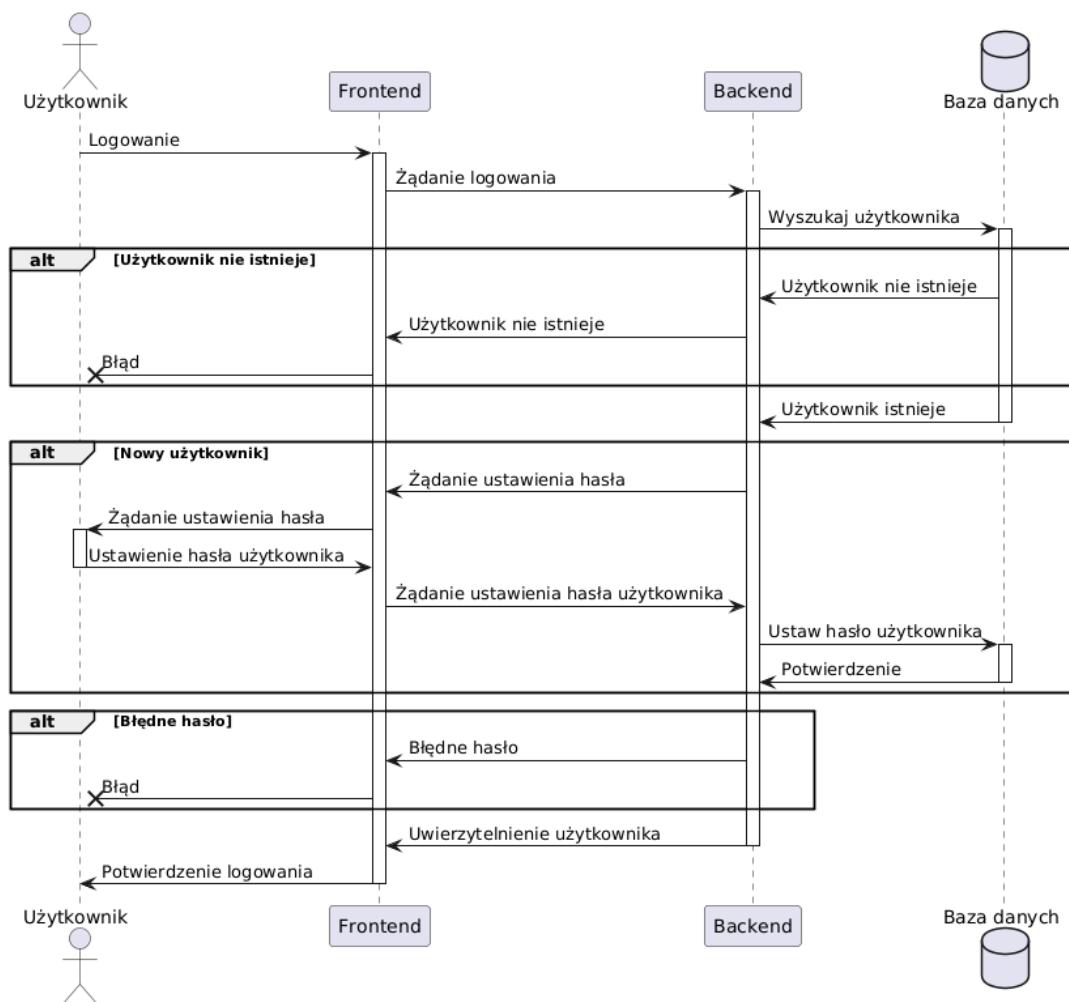
### 5.4.1 Rejestracja i pierwsze logowanie użytkownika

Rejestracja użytkownika może zostać dokonana jedynie przez administratora systemu. W tym celu powinien on wypełnić formularz rejestracyjny wprowadzając nazwę użytkownika oraz jego adres email. Po zatwierdzeniu formularza, w widoku wszystkich użytkowników pojawi się nowy rekord z danymi właśnie utworzonego użytkownika. Następnie administrator powinien przekazać nowo zarejestrowanemu użytkownikowi jego nazwę, którą musi wpisać na ekranie logowania - nie jest wymagane przy tym wpisywanie hasła. Po zatwierdzeniu formularza, i wysłaniu danych do serwera, zostanie sprawdzone czy użytkownik o podanej nazwie istnieje w bazie danych i czy ma przypisane hasło. Jeżeli nie,

zwróci kod 206 - *Partial Content*, a aplikacja udostępnii użytkownikowi możliwość ustawienia hasła. Po jego wpisaniu, potwierdzeniu i zatwierdzeniu formularza, użytkownik zostanie zalogowany do systemu. Diagram sekwencji pierwszego logowania użytkownika został przedstawiony na rysunku 5.8, a szczegóły procesu zostały opisane w rozdziale 5.4.2

## 5.4.2 Logowanie

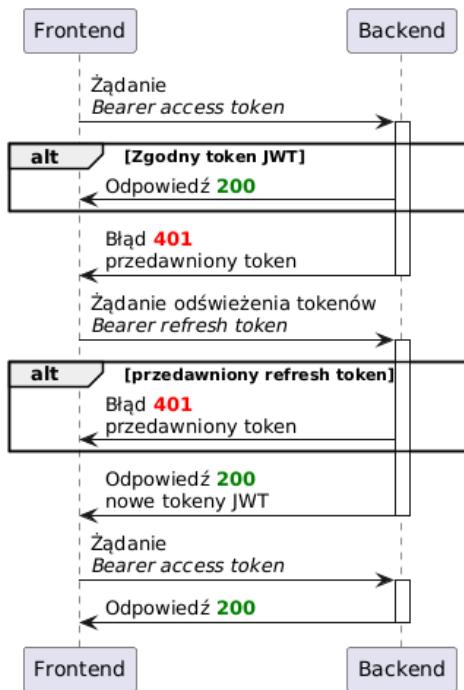
Logowanie do systemu odbywa się poprzez przesłanie żądania HTTP z danymi logowania do serwera aplikacyjnego. Po jego otrzymaniu serwer zwraca się do bazy danych w celu znalezienia użytkownika o podanej nazwie. Jeżeli użytkownik nie istnieje, serwer zwraca kod błędu 401. W przeciwnym wypadku sprawdza, czy zahashowane hasło użytkownika jest zgodne z zapisanym w bazie. Jeżeli hasła się zgadzają, serwer generuje dwa tokeny JWT i zwraca je w odpowiedzi. Aplikacja kliencka zapisuje otrzymane tokeny w pamięci lokalnej przeglądarki i przechodzi do widoku panelu głównego. Diagram czynności logowania został przedstawiony na rysunku 5.8.



Rysunek 5.8: Diagram sekwencji logowania

### 5.4.3 Uwierzytelnianie użytkownika

Klient po zalogowaniu się do systemu otrzymuje od serwera dwa tokeny JWT, które są przechowywane w pamięci lokalnej przeglądarki. Aby mógł korzystać z zasobów serwera, musi dołączać pierwszy z nich - token dostępowy - do nagłówka każdego żądania HTTP pod kluczem `Authorization` poprzedzając go słowem `Bearer`. Token ten jest ważny przez określony czas, po którym traci ważność - serwer symbolizuje to kodem błędu 401. Jeżeli zaistnieje taka sytuacja, klient musi wysłać żądanie odświeżenia tokena do serwera dołączając drugi token - odświeżający - który ma dłuższy czas przedawnienia. Serwer następnie sprawdza, czy token odświeżający jest poprawny i zwraca nowe tokeny. W przeciwnym wypadku klient musi ponownie zalogować się do systemu. Diagram sekwencji procesu uwierzytelniania użytkownika został przedstawiony na rysunku 5.9.



Rysunek 5.9: Diagram sekwencji procesu uwierzytelniania użytkownika

### 5.4.4 Harmonogramowanie

Harmonogramowanie jest częścią systemu, która pochłania najwięcej zasobów serwera, ponieważ dane za każdym razem są przetwarzane na nowo. Odejście od praktyki wstępnego generowania harmonogramów i zapisywania ich w bazie danych umożliwiło zredukowanie ilości danych przechowywanych w bazie oraz konieczności ich aktualizacji w przypadku zmian w grafiku pracy.

#### 5.4.4.1 Zapis harmonogramu

Harmonogramowanie pracy użytkowników odbywa się poprzez wysłanie żądania HTTP z danymi harmonogramu do serwera aplikacyjnego. Struktura wysyłana wraz z żądaniem musi być zgodna z odpowiadającym jej DTO i zawierać informacje o dacie rozpoczęcia, jednostce lub użytkowniku, którego dotyczy oraz blokach czasowych, jakie mają się w nim znaleźć. Następnie tworzona jest struktura danych odpowiadająca tabelom opisanym w rozdziale 5.2.3 i zapisywana w bazie danych. Po zakończeniu procesu, serwer zwraca kod 200 - OK, a aplikacja kliencka przechodzi do widoku harmonogramu. W przypadku błędu, serwer zwraca kod 400 - Bad Request.

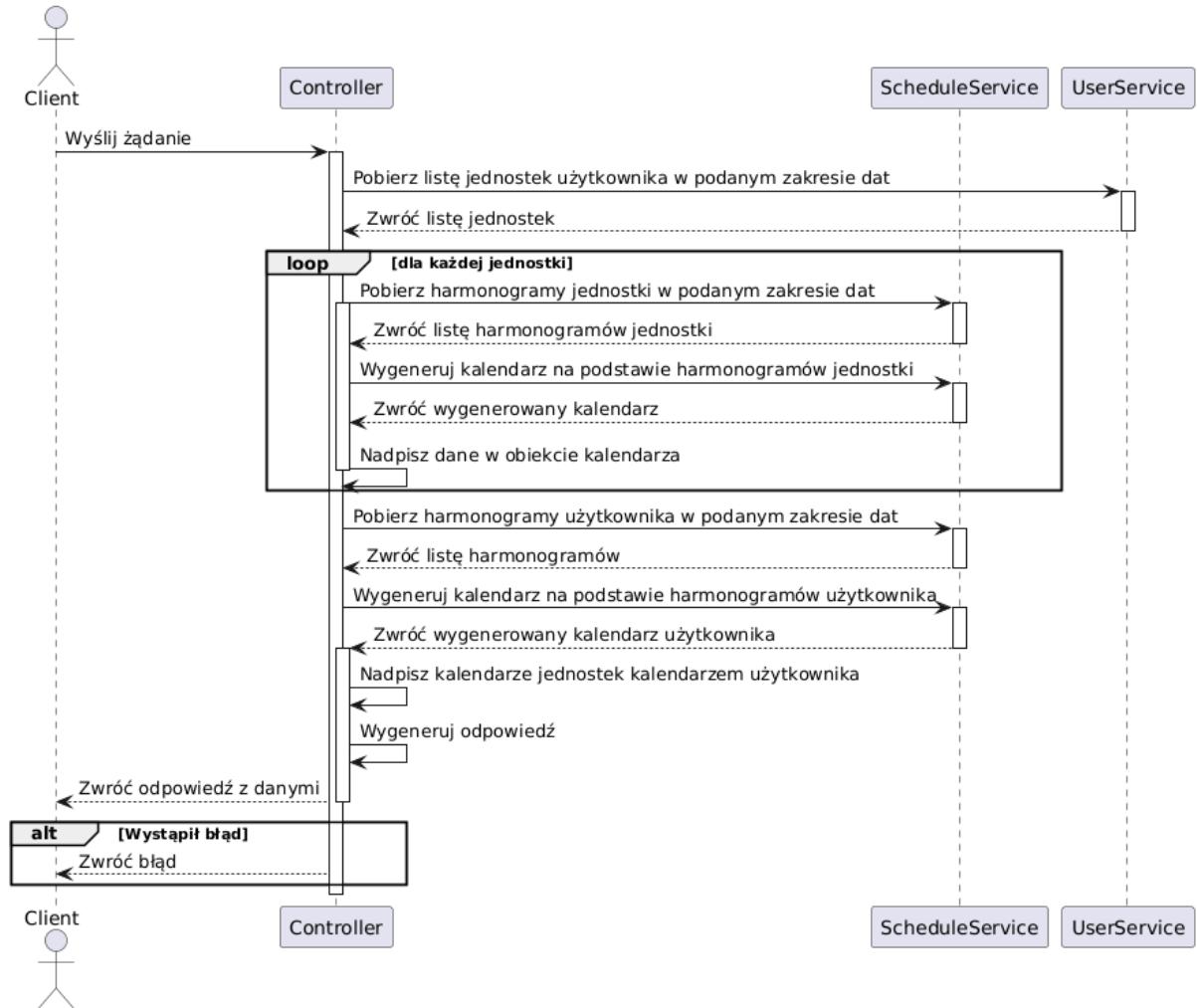
#### 5.4.4.2 Odczyt harmonogramu

Odczyt harmonogramu odbywa się poprzez wysłanie żądania GET na adres odpowiednio:

- `/schedule/getUnit/{id}/{startDate}/{endDate}` — dla harmonogramu jednostki organizacyjnej,
- `/schedule/getUser/{id}/{startDate}/{endDate}` — dla harmonogramu użytkownika,

uzupełniając odpowiednio parametry w nawiasach klamrowych. Oba żądania działają w podobny sposób - pierwsze z nich uwzględnia wyłącznie harmonogramy jednostki, natomiast drugie łączy harmonogramy wszystkich jednostek przypisanych do użytkownika, a następnie nadpisuje je harmonogramami użytkownika. W odpowiedzi serwer zwraca dane w formacie JSON, które są przetwarzane przez aplikację kliencką i wyświetlane w odpowiednim widoku.

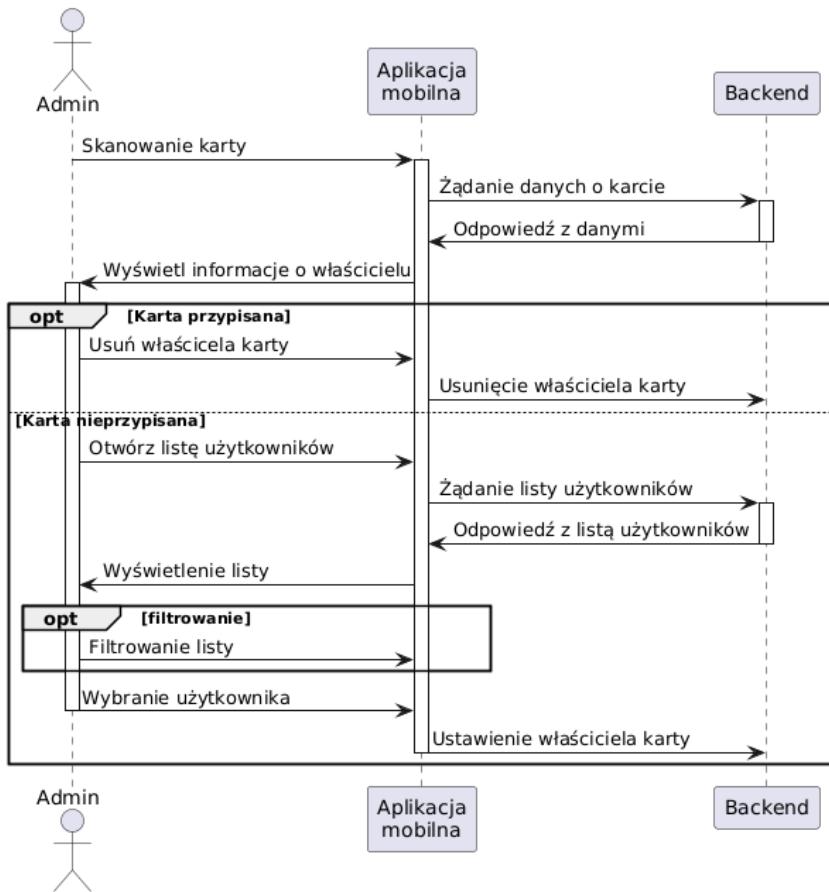
Odczyt harmonogramu użytkownika jest dużo bardziej złożony niż jednostki organizacyjnej. Wymaga on pobrania wszystkich jednostek, do których należał użytkownik w danym okresie czasu, następnie dla każdej z nich odczytania harmonogramu i połączenia ich w jeden widok przechowywany w klasie pomocniczej `Calendar`. Jeżeli użytkownik posiada również własny harmonogram, serwer generuje jego widok i nadpisuje dane przechowywane w klasie. Diagram sekwencji procesu został przedstawiony na rysunku 5.10.



Rysunek 5.10: Diagram sekwencji odczytu harmonogramu użytkownika

#### 5.4.5 Przypisanie karty dostępowej

Przypisanie karty dostępowej odbywa się przy wykorzystaniu aplikacji mobilnej. Po zalogowaniu się do systemu administrator powinien wybrać zakładkę CARDS, a następnie rozpoczęć skanowanie tagu NFC. Po pozytywnym odczytaniu danych aplikacja wyśle do serwera żądanie o informacje na temat karty, a w kolejnym kroku wyświetli informacje o jej właścicielu. Jeżeli karta jest już przypisana do użytkownika, aplikacja umożliwi jej usunięcie, a w przeciwnym wypadku udostępni listę rozwijaną z jego wyborem. Po zatwierdzeniu, aplikacja wyśle do serwera żądanie przypisania karty do użytkownika i otrzyma odpowiedź. Diagram sekwencji przypisania karty dostępowej do użytkownika został przedstawiony na rysunku 5.11.



Rysunek 5.11: Diagram sekwencji odczytu i przypisania karty dostępowej

#### 5.4.6 Uruchomienie czytnika kart

Uruchomienie czytnika kart odbywa się poprzez podłączenie go do zasilania. Następnie czytnik stara się połączyć z siecią bezprzewodową, która jest zapisana w jego pamięci. Po pozytywnym połączeniu, czytnik wysyła do serwera aplikacyjnego żądanie o informacje na swój temat dołączając do niego adres MAC. Serwer zwraca informacje o czytniku, a ten przechodzi do trybu oczekiwania na odczytanie karty. Jeżeli któryś z kroków nie powiedzie się, czytnik przechodzi w stan błędu i udostępnia sieć Wi-Fi o nazwie WMS-AP z hasłem 123456789 w celu ręcznego skonfigurowania połączenia. Po połączeniu się innym urządzeniem z siecią czytnika, należy wpisać w przeglądarce adres 192.168.4.1 i wprowadzić dane dostępowe do sieci ogólnej. Widok strony konfiguracyjnej czytnika został przedstawiony na rysunku 5.13. Po zatwierdzeniu formularza czytnik zapisuje dane, restartuje się i ponawia opisany wcześniej proces.

## 5.5 Połączenie modułów czytnika kart

Czytnik kart składa się z dwóch modułów oraz kilku elementów pasywnych. Pierwszy z modułów to mikrokontroler **Raspberry Pi Pico W** odpowiedzialny za przetwarzanie danych i komunikację z serwerem aplikacyjnym. Drugi moduł - oznaczony symbolem **RFID-RC522** - odpowiada za odczytanie tagów NFC i przekazanie ich do mikrokontrolera. Oba układy są połączone ze sobą za pomocą interfejsu SPI (ang. *Serial Peripheral Interface*). Schemat połączeń czytnika kart został przedstawiony na rysunku 5.12.

Do elementów pasywnych układu należą:

- Rezystory ograniczające prąd,
- Dioda LED RGB (ang. *Red, Green, Blue*) ze wspólną katodą, która sygnalizuje stan wewnętrzny czytnika.

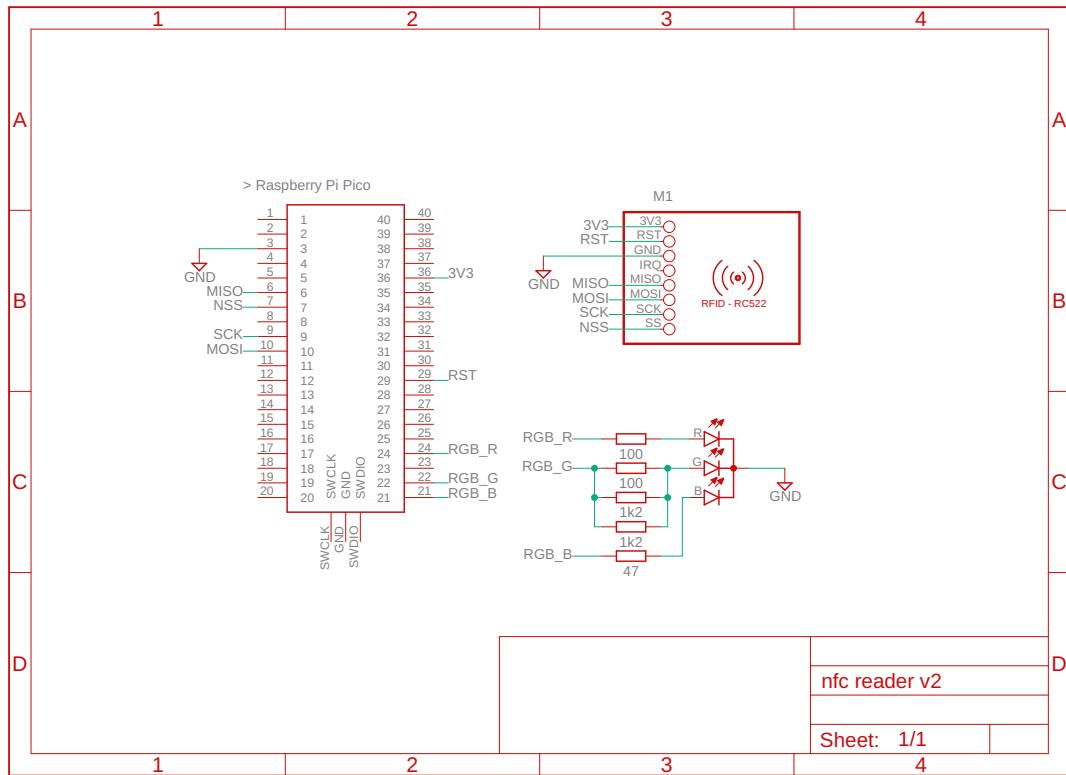
Zgodnie z sposobem podłączania diod do każdej anody powinien być dołączony rezistor ograniczający prąd. Producent zaleca następujące wartości rezystorów:

$$\begin{cases} R_R = 100\Omega \\ R_G = 82\Omega \\ R_B = 47\Omega \end{cases} \quad (5.1)$$

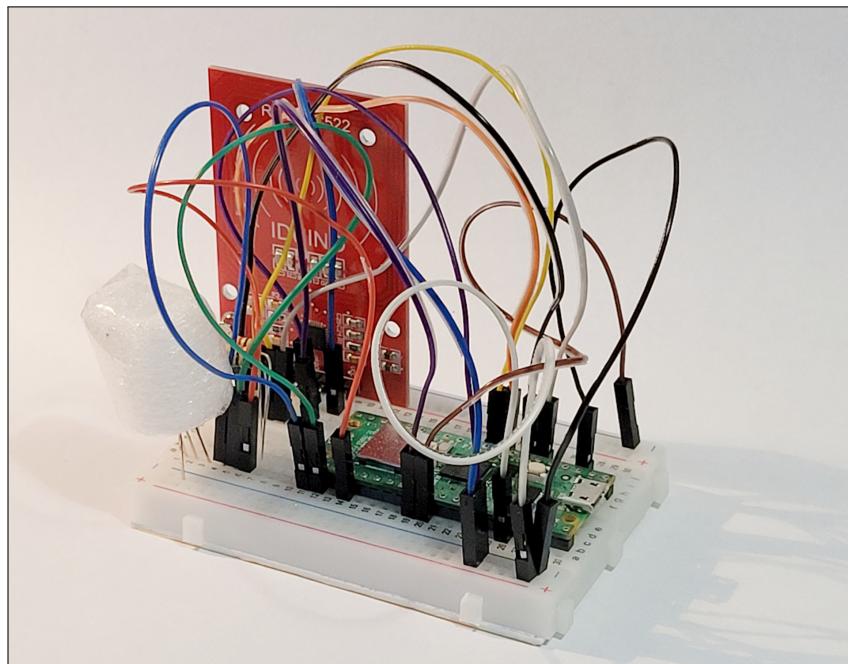
z dokładnością do 5%. Niemożność uzyskania rezystora dla kanału zielonego spowodowała, że połączono równolegle dostępne rezystory o wartościach  $100\Omega$ ,  $1.2k\Omega$  i  $1.2k\Omega$ . Dzięki temu uzyskano rezystancję równą:

$$R_G = \frac{1}{\frac{1}{100} + \frac{1}{1200} + \frac{1}{1200}} \approx 85.71\Omega \quad (5.2)$$

która odbiega od wartości zalecanej o mniej niż 5%.



Rysunek 5.12: Schemat połączeń czytnika kart NFC



Rysunek 5.13: Prototyp czytnika kart na płytce stykowej

# Rozdział 6

## Weryfikacja i walidacja

### 6.1 Walidacja danych

Walidacja danych przesyłanych pomiędzy serwerem, a klientem jest jednym z kluczowych elementów w procesie tworzenia aplikacji internetowych. Jej brak może doprowadzić do poważnych problemów związanych z bezpieczeństwem. Szczególnie ważne jest to w przypadku aplikacji, które wymagają od użytkownika wprowadzenia danych.

#### 6.1.1 Walidacja po stronie klienta

Walidacja po stronie klienta jest realizowana przy pomocy biblioteki `Valibot`. Opiera się ona na obiektach walidujących, którymi sprawdzany jest formularz przed jego wysłaniem. Tworzenie obiektów jest bardzo proste i intuicyjne - polega na tworzeniu potoków, które zawierają kolejne funkcje walidujące. Przykład walidacji z użyciem biblioteki został przedstawiony na listingu 6.1.

---

```
1 const newRecordSchema = v.object({
2     unitId: v.required(v.number()),
3     hours: v.required(v.pipe(v.number(), v.minLength(0.25),
4         v.maxLength(24))),
5     date: v.required(v.date()),
6     description: v.required(v.pipe(v.string(), v.nonEmpty(),
7         v.maxLength(255))),
8 })
```

---

Listing 6.1: Przykład walidacji formularza przy pomocy biblioteki Valibot

### 6.1.2 Walidacja po stronie serwera

Do walidacji danych wykorzystano rozwiązanie udostępniane przez **Spring Framework**. Opiera się ono na zbiorze adnotacji walidujących, które umieszcza się nad polami klas. Można wykorzystać gotowe adnotacje lub wyrażenia regularne. Przykład walidacji danych został przedstawiony na listingu 6.2.

---

```
1 @DateTimeFormat(pattern = "yyyy-MM-dd")
2 private LocalDate dateOfBirth;
```

---

Listing 6.2: Przykład walidacji danych przy pomocy adnotacji w Spring Framework

## 6.2 Testy

Testowanie aplikacji jest kluczowym elementem wytwarzania oprogramowania. Pozwala ono na wczesne wykrycie błędów, które na etapie produkcji mogłyby prowadzić do poważnych problemów, czy nawet awarii systemu. Nie powinno się jednak zapominać o ciągłej konieczności testowania aplikacji, nawet po dłuższym czasie od jej wdrożenia.

System został przetestowany aby upewnić się, że działa on poprawnie. Zdecydowano się na przeprowadzenie ręcznych testów punktów końcowych, systemowych oraz akceptacyjnych w symulowanym środowisku produkcyjnym. Przetestowano również czytnik kart zbliżeniowych pod kątem poprawności odczytu danych.

### 6.2.1 Testy punktów końcowych API

Testy punktów końcowych pozwalają na sprawdzenie poprawności działania serwera pod kątem: obsługi danych, zależności czasowych, utraty danych czy innych niepożądanych efektów ubocznych. Poprawność działania wszystkich punktów końcowych API została sprawdzona przy użyciu narzędzia **Swagger**. Umożliwia ono automatyczną generację zapytań bazując na kodzie źródłowym. Szczególną uwagę przyłożono do warunków brzegowych. W trakcie ręcznego testowania napotkano kilka błędów, które zostały od razu poprawione. Dotyczyły one głównie przypadków, gdy do serwera trafiały wartości brakujące, które nie były obsługiwane przez aplikację.

Często zdarzało się, że serwer poprawnie obsługiwał żądania wysłane przy pomocy narzędzia **Swagger**, ale zwracał błędy w aplikacji klienckiej. Było to najczęściej spowodowane pośpiechem przy wysyłaniu kolejnych żądań. W takich sytuacjach aplikacja, nie mając jeszcze danych, wysyłała zapytania do serwera uzupełniając je wartością `null` lub `undefined`. W celu uniknięcia takich sytuacji dodano kolejkowanie żądań. Największym

błędem, jaki przy tym wystąpił był brak odświeżenia widoku po wylogowaniu i zalogowaniu się innego użytkownika. Rozwiązano go dopiero po ręcznym debugowaniu aplikacji.

### 6.2.2 Testy czytnika kart zbliżeniowych

Działanie czytnika kart zostało przetestowane przy użyciu tagów MifareClassic oraz MifareUltralight. Oba typy tagów działały poprawnie i były rozpoznawane przez czytnik. Przy próbie emulacji tagów na urządzeniu z systemem Android, czytnik nie był w stanie odczytać numeru UID. Problem ten nie został rozwiązany, ponieważ wiązałoby się to z koniecznością zmiany sposobu autoryzacji lub modyfikacji systemu operacyjnego urządzenia mobilnego, co w ogólnym przypadku nie jest zalecane.

### 6.2.3 Testy systemowe

Testy systemowe miały na celu weryfikację działającego systemu jako całości. Sprawdzono przede wszystkim, czy poszczególne moduły współpracują ze sobą oraz czy poprawnie zrealizowano kluczowe procesy. Przeprowadzono je w symulowanym środowisku produkcyjnym. W ich trakcie sprawdzono: integralność modułów, obsługę błędów oraz kompatybilność z różnymi urządzeniami i przeglądarkami. Potwierdziły one poprawność działania wszystkich modułów w standardowych przypadkach użycia.

### 6.2.4 Testy akceptacyjne

W celu weryfikacji, czy aplikacja spełnia wymagania użytkowników, wykonano serię testów akceptacyjnych w symulowanym środowisku produkcyjnym. W ich trakcie sprawdzano, czy aplikacja w poprawny sposób obsługuje scenariusze: rejestracji użytkownika, logowania, zarządzania pracownikami, zadaniami, czasem pracy i strukturą firmy. Testy akceptacyjne potwierdziły, że aplikacja poprawnie spełnia postawione wymagania.

## 6.3 Bezpieczeństwo

Kluczowym aspektem aplikacji operującej na danych osobowych, jest zabezpieczenie ich przed nieautoryzowanym dostępem. W tym celu podjęto szereg działań, których celem było zabezpieczenie aplikacji na różnych poziomach. Wdrożono również mechanizmy, które zapobiegają częścią ataków. Zastosowano następujące rozwiązania:

- szyfrowanie komunikacji między modułami za pomocą protokołu HTTPS,
- haszowanie danych uwierzytelniających użytkownika,
- autoryzacja przy pomocy tokenów JWT o krótkim okresie ważności, zapobiegając atakom typu *replay*,

- zapytania generowane przez zaufane biblioteki oraz JPA, zapobiegając atakom typu *SQL Injection*,
- wprowadzenie zabezpieczeń przed atakami typu *Cross-Site Scripting* (XSS).

# Rozdział 7

## Podsumowanie i wnioski

### 7.1 Wnioski

Celem pracy było zaprojektowanie i implementacja systemu do zarządzania personelem w firmie. Umożliwia on zarządzanie pracownikami, ich zadaniami, czasem pracy i strukturą firmy w sposób zautomatyzowany. Wymierne korzyści z zastosowania systemu to zwiększenie efektywności pracy, zwiększenie kontroli nad zadaniami i pracownikami oraz zwiększenie przejrzystości ról w firmie. Przeprowadzona analiza wymagań pozwoliła na stworzenie systemu, który spełnia oczekiwania użytkowników, co potwierdziły przeprowadzone testy.

Wynikiem pracy jest system, który spełnia założenia projektowe i udostępnia użytkownikom sprecyzowane w wymaganiach funkcjonalności. Jest on bezpieczny, łatwy w obsłudze i automatyzuje szereg procesów, które wcześniej były wykonywane ręcznie. Aplikacja mobilna pozwala na dostęp do systemu z każdego miejsca i w każdym czasie, a dzięki funkcji dodawania kart zbliżeniowych niweluje konieczność ręcznego wpisywania danych lub użycia czytników podłączanych do komputera. Użycie mikrokontrolerów pozwala na automatyzację zapisu czasu pracy pracowników, co znacznie ułatwia jego śledzenie. Dzięki ich zastosowaniu możliwa jest również obsługa kontroli dostępu do pomieszczeń.

System został zrealizowany w wersji podstawowej, jednak posiada wiele możliwości rozwoju. Udało się zrealizować funkcjonalności dotyczące: rejestracji czasu pracy, zarządzania pracownikami, harmonogramami pracy, zadaniami, czasem pracy i strukturą firmy. Nie udało się zrealizować funkcjonalności dotyczących raportów, wniosków oraz automatycznego śledzenia czasu pracy pracowników. Wszystkie one mogą jednak zostać dodane w przyszłości, co znacznie zwiększyłoby użyteczność systemu. Implementacja kontroli dostępu do pomieszczeń została zrealizowana w ograniczonym zakresie, jednak możliwe jest jej rozbudowanie.

### 7.1.1 Problemy napotkane podczas pracy

Podczas pracy napotkano kilka problemów, które znaczco ją utrudniły i wydłużyły. Największym problemem było to, że przy projektowaniu systemu nie wzięto pod uwagę stopnia jego złożoności. Poskutkowało to koniecznością zrezygnowania z części funkcjonalności lub ograniczeniem ich zakresu na rzecz dotrzymania terminu.

W założeniach aplikacji mobilnej, miała ona umożliwiać autoryzację użytkownika poprzez przyłożenie urządzenia do czytnika kart zbliżeniowych. Niestety, przez decyzje podjęte na etapie projektowania komunikacji między układem mikrokontrolera a serwerem, nie udało się zrealizować tej funkcjonalności. Autoryzacja opiera się o numery seryjne kart, co znacznie ułatwia proces ich odczytania i rejestracji. Urządzenie mobilne musiało by być w stanie emulować te numery, jednakże nie jest to możliwe bez modyfikacji systemu operacyjnego.

Framework **Expo** użyty do stworzenia aplikacji mobilnej udostępniał funkcję odczytywania zmiennych środowiskowych. Niestety, po pewnym czasie funkcja przestała odświeżać wartości, co wiązało się z koniecznością częstej, wielokrotnej komplikacji aplikacji. Problem ten nie został rozwiązany, co znacznie wydłużyło pracę nad aplikacją mobilną.

### 7.1.2 Ocena dobranych technologii po zakończeniu pracy

Technologie wybrane do realizacji systemu okazały się być idealne, ponieważ w sposób znaczący przyspieszyły pracę nad projektem. Mikrokontroler, na którym zainstalowano system **MicroPython** okazał się być bardzo prosty w obsłudze i umożliwił szybkie zaimplementowanie funkcjonalności związanych z odczytem kart zbliżeniowych. **Spring Framework** nie sprawił, że system stał się zbyt skomplikowany, a wręcz przeciwnie - pozwolił na jasne zdefiniowanie struktury aplikacji i łatwe zarządzanie nią. Framework **Vue.js** zrobił na autorze bardzo dobre wrażenie swoją prostotą i intuicyjnością. W porównaniu z innymi frameworkami, takimi jak **Next.js**, okazał się być znacznie przyjazniejszy, jawnie oddzielając warstwę prezentacji, logiki i danych. **Expo** okazało się najbardziej uciążliwą technologią, jednakże poza wymienionym nie sprawiło innych problemów. Użycie innej technologii, np. **Flutter** mogłoby przyspieszyć pracę nad aplikacją mobilną, jednakże konieczność nauki nowego frameworka mogłaby wydłużyć czas potrzebny na jej stworzenie.

## 7.2 Perspektywy rozwoju

System posiada wiele możliwości rozwoju, które znacznie zwiększyłyby jego użytkowość. Pierwszym krokiem, jaki należy podjąć jest zwiększenie ilości czytników kart zbliżeniowych - obecnie w systemie działa tylko jeden, ale możliwe jest łatwe podłączenie większej ich ilości. Następnie należałoby zająć się brakującymi częściami: raportami, wnioskami i automatycznym śledzeniem czasu pracy pracowników. Raporty powinny być ge-

nerowane w formie plików PDF (ang. *Portable Document Format*), które można było wydrukować lub przesłać mailem. Do wniosków - oprócz możliwości ich składania i akceptacji - dobrze było dodać możliwość wymieniania się wiadomościami w formie czatu. Automatyczne śledzenie czasu pracy może być zrealizowane algorytmicznie, wywołując odpowiednie funkcje w określonych momentach. Możliwe, że dobrym wyborem było zastosowanie algorytmów uczenia maszynowego, które nauczyłyby się rozpoznawać co pracownik robi w danym momencie. Kontrola dostępu do pomieszczeń również wymaga rozbudowy - aktualnie użytkownicy mogą autoryzować się przy każdym czytniku bez ustalania dostępu do pomieszczeń. W przyszłości warto byłoby dodać możliwość ustalania uprawnień dla poszczególnych użytkowników. W dalszej perspektywie można rozszerzać system o kolejne funkcjonalności znane z systemów HCM.

### 7.3 Podsumowanie

Samodzielne stworzenie systemu, który spełnia podane wymagania, jest bardzo czasochłonne. Wymaga od programisty bardzo dużej wiedzy, umiejętności i zaangażowania. Duże zespoły są w stanie tworzyć systemy o wiele bardziej rozbudowane i złożone w krótkim czasie - co widać po rozwiązaniach dostępnych na rynku. Każdy z członków zespołu wnosi do projektu swoje doświadczenie, wiedzę i specjalizację, dzięki czemu praca nad projektem jest bardziej efektywna i przyjemna.

Wykonanie projektu pozwoliło na zrozumienie procesów zachodzących w firmach i ukażało, jakie korzyści przynosi zastosowanie systemu informatycznego. Praca nad systemem pozwoliła na zdobycie ogromnych zasobów wiedzy - tak technicznej, jak i biznesowej. Projekt okazał się być dużym wyzwaniem, ale również przyniósł wiele satysfakcji. Udało się stworzyć system, który spełnia założenia projektowe i jest gotowy do dalszego rozwoju. Praca nad projektem pozwoliła na zdobycie cennego doświadczenia i wiedzy, które przyniosą wymierne korzyści w przyszłości. System posiada wiele możliwości rozwoju, które mogą znacznie zwiększyć jego użyteczność i funkcjonalność. Warto kontynuować prace nad nim, aby w pełni wykorzystać jego potencjał.



# Bibliografia

- [1] Avigilon. *Avigilon - Security Solutions*. URL: <https://www.avigilon.com/> (term. wiz. 05.11.2024).
- [2] Axios. *Axios*. URL: <https://axios-http.com/> (term. wiz. 04.10.2024).
- [3] Tom Collings. *Controller, Service and Repository*. 2021. URL: <https://tom-collings.medium.com/controller-service-repository-16e29a4684e5> (term. wiz. 25.09.2024).
- [4] Ministerstwo Cyfryzacji. *WCAG 2.1 w skrócie*. URL: <https://www.gov.pl/web/dostepnosc-cyfrowa/wcag-21-w-skrocie> (term. wiz. 07.10.2024).
- [5] Docker. *Docker*. URL: <https://www.docker.com/> (term. wiz. 25.09.2024).
- [6] Stefan Trzcieliński Edmund Pawłowski. *Zarządzanie przedsiębiorstwem, Funkcje i struktury*. Poznań: Wydawnictwo Politechniki Poznańskiej, 2011. ISBN: 978-83-7143-968-1.
- [7] Expo. *Expo*. URL: <https://expo.dev/> (term. wiz. 25.10.2024).
- [8] Expo. *Expo Application Services (EAS)*. URL: <https://expo.dev/eas> (term. wiz. 25.10.2024).
- [9] Figma. *Figma*. URL: <https://www.figma.com/> (term. wiz. 23.09.2024).
- [10] Raspberry Pi Foundation. *PicoZero*. URL: <https://github.com/RaspberryPiFoundation/picozero> (term. wiz. 08.11.2024).
- [11] Git. *Git - Distributed Version Control System*. URL: <https://git-scm.com/> (term. wiz. 25.09.2024).
- [12] Fabian Hiller. *Valibot*. URL: <https://valibot.dev/> (term. wiz. 04.11.2024).
- [13] Intlify. *Vue I18n*. URL: <https://vue-i18n.intlify.dev/> (term. wiz. 04.10.2024).
- [14] JetBrains. *IntelliJ IDEA*. URL: <https://www.jetbrains.com/idea/> (term. wiz. 23.09.2024).
- [15] jwtk. *Java JWT: JSON Web Token for Java and Android*. URL: <https://github.com/jwtk/jjwt> (term. wiz. 09.09.2024).

- [16] Kadromierz. *Potwierdzenie obecności w pracy – jakie są sposoby?* 2023. URL: [https://kadromierz.pl/blog/potwierdzenie-obecnosci-w-pracy-jakie-sa-sposoby/#Potwierdzenie\\_obecnosci\\_wpracy\\_cowarto\\_wiedziec](https://kadromierz.pl/blog/potwierdzenie-obecnosci-w-pracy-jakie-sa-sposoby/#Potwierdzenie_obecnosci_wpracy_cowarto_wiedziec) (term. wiz. 05.11.2024).
- [17] Tailwind Labs. *Heroicons*. URL: <https://heroicons.com/> (term. wiz. 04.10.2024).
- [18] Tailwind Labs. *Tailwind CSS*. URL: <https://tailwindcss.com/> (term. wiz. 26.11.2024).
- [19] Project Lombok. *Project Lombok*. URL: <https://projectlombok.org/> (term. wiz. 25.09.2024).
- [20] Nedap Security Management. *Na czym polega kontrola dostępu i dlaczego jest tak ważna?* URL: <https://www.nedapsecurity.com/pl/insight/na-czym-polega-kontrola-dostepu-i-dlaczego-jest-tak-wazna/> (term. wiz. 05.11.2024).
- [21] Software Mansion. *React Native Gesture Handler*. URL: <https://docs.swmansion.com/react-native-gesture-handler/> (term. wiz. 25.10.2024).
- [22] MicroPython. *MicroPython*. URL: <https://micropython.org/> (term. wiz. 10.11.2024).
- [23] Microsoft. *Visual Studio Code*. URL: <https://code.visualstudio.com/> (term. wiz. 04.10.2024).
- [24] MintHCM. *MintHCM*. URL: <https://minthcm.org/> (term. wiz. 30.09.2024).
- [25] Joanna M. Moczydłowska. „Efektywność zarządzania kapitałem ludzkim jako element efektywności organizacyjnej”. W: *Efektywność organizacji* (2013), s. 183–192.
- [26] Moment.js. *Luxon*. URL: <https://moment.github.io/luxon/#/> (term. wiz. 03.12.2024).
- [27] MySQL. *MySQL*. URL: <https://www.mysql.com/> (term. wiz. 25.09.2024).
- [28] MySQL. *MySQL Connector*. URL: <https://www.mysql.com/products/connector/> (term. wiz. 25.09.2024).
- [29] Nativewind. *Nativewind*. URL: <https://www.nativewind.dev/> (term. wiz. 25.11.2024).
- [30] Oracle. *Oracle Human Capital Management (HCM)*. URL: <https://www.oracle.com/human-capital-management/> (term. wiz. 30.09.2024).
- [31] Oracle. *What is Human Capital Management (HCM)?* URL: <https://www.oracle.com/pl/human-capital-management/what-is-hcm/> (term. wiz. 05.11.2024).
- [32] Daniel Perron. *MicroPython MFRC522*. URL: <https://github.com/danjperron/micropython-mfrc522/tree/master?tab=readme-ov-file> (term. wiz. 11.11.2024).
- [33] Raspberry Pi. *Pico-series Microcontrollers*. URL: <https://www.raspberrypi.com/documentation/microcontrollers/pico-series.html> (term. wiz. 08.11.2024).

- [34] Raspberry Pi. *Pico W Datasheet*. 2023. URL: <https://datasheets.raspberrypi.com/picow/pico-w-datasheet.pdf> (term. wiz. 08.11.2024).
- [35] PlantUML. *PlantUML*. URL: <https://plantuml.com/> (term. wiz. 20.12.2024).
- [36] PrimeTek. *PrimeVue*. URL: <https://primevue.org/> (term. wiz. 04.10.2024).
- [37] Kenneth Reitz. *Requests: HTTP for Humans*. URL: <https://requests.readthedocs.io/en/latest/> (term. wiz. 12.11.2024).
- [38] RevTelTech. *React Native NFC Manager*. URL: <https://github.com/revtel/react-native-nfc-manager> (term. wiz. 25.10.2024).
- [39] SAP. *Human capital management (HCM)*. URL: <https://www.sap.com/products/hcm.html> (term. wiz. 30.09.2024).
- [40] Satel. *Satel - Security Systems*. URL: <https://www.satel.pl/> (term. wiz. 05.11.2024).
- [41] Luceos Smart. *Jak prawidłowo monitorować czas pracy pracowników?* 2023. URL: <https://www.luceossmart.com/post/jak-prawid%C5%82owo-monitorowa%C4%87-czas-pracy-pracownik%C3%B3w#viewer-dz3yt5254> (term. wiz. 05.11.2024).
- [42] Spring. *Spring Boot*. URL: <https://spring.io/projects/spring-boot> (term. wiz. 09.09.2024).
- [43] Spring. *Spring Framework*. URL: <https://spring.io/> (term. wiz. 25.09.2024).
- [44] Spring. *Spring Security*. URL: <https://spring.io/projects/spring-security> (term. wiz. 09.09.2024).
- [45] SpringDoc. *SpringDoc*. URL: <https://springdoc.org/> (term. wiz. 09.09.2024).
- [46] Struktura organizacyjna firmy – jak może wyglądać? Rodzaje i przykłady. URL: <https://tomhrm.com/struktura-organizacyjna-firmy-rodzaje-przyklady/> (term. wiz. 26.11.2024).
- [47] Thonny. *Thonny, Python IDE for beginners*. URL: <https://thonny.org/> (term. wiz. 03.12.2024).
- [48] Timeular. *Excel Time Tracking: How to Track Your Time in Excel*. 2023. URL: <https://timeular.com/blog/excel-time-tracking/> (term. wiz. 05.11.2024).
- [49] Ustawa z dnia 26 czerwca 1974 r. Kodeks pracy. 1974. URL: <https://isap.sejm.gov.pl/isap.nsf/download.xsp/WDU19740240141/U/D19740141Lj.pdf> (term. wiz. 27.11.2024).
- [50] Vue. *Vue Router*. URL: <https://router.vuejs.org/> (term. wiz. 04.10.2024).
- [51] Vue.js. *Vue.js*. URL: <https://vuejs.org/> (term. wiz. 04.10.2024).
- [52] Zeplin. *Zeplin*. URL: <https://zeplin.io/> (term. wiz. 23.09.2024).



## **Dodatki**



# Spis skrótów i symboli

HCM System Zarządzania Kapitałem Ludzkim (ang. *Human Capital Management*).

RFID Identyfikacja za pomocą fal radiowych (ang. *Radio-Frequency Identification*).

NFC Komunikacja bliskiego zasięgu (ang. *Near Field Communication*).

LED Dioda elektroluminescencyjna (ang. *Light Emitting Diode*).

HTTP Protokół komunikacji w sieci WWW (ang. *Hypertext Transfer Protocol*).

JSON Notacja obiektów JavaScript (ang. *JavaScript Object Notation*).

MAC Adres sprzętowy karty sieciowej (ang. *Media Access Control address*).

JPA Standard mapowania obiektowo-relacyjnego (ang. *Java Persistence API*).

DTO Obiekt transferu danych (ang. *Data Transfer Object*).

JWT Token JSON (ang. *JSON Web Token*).



# **Lista dodatkowych plików, uzupełniających tekst pracy**

W systemie do pracy dołączono dodatkowe pliki:

- źródła programu w archiwum `WMS_sources.zip`, zawierające:
  - WMS\_Backend — kod serwera,
  - WMS\_Frontend — kod klienta,
  - WMS\_EMBEDDED — kod mikrokontrolera,
  - WMS\_DB — skrypty tworzące bazę danych,
- film pokazujący działanie systemu.



# Spis rysunków

4.1	Makieta głównego widoku dla managera . . . . .	18
4.2	Zmiany w interfejsie użytkownika . . . . .	19
4.3	Makieta struktury widoku . . . . .	20
4.4	Widok strony logowania . . . . .	21
4.5	Panel użytkownika . . . . .	21
4.6	Widok użytkowników . . . . .	22
4.7	Okna dialogowe widoku użytkowników . . . . .	22
4.8	Widok karty pracy . . . . .	23
4.9	Okno dialogowe dodawania wpisu do karty pracy . . . . .	23
4.10	Widok wpisów oczekujących na akceptację . . . . .	24
4.11	Widok jednostek organizacyjnych . . . . .	24
4.12	Okna dialogowe widoku jednostek organizacyjnych . . . . .	25
4.13	Okna dialogowe dodawania użytkowników do jednostki organizacyjnej . . . . .	25
4.14	Widok harmonogramu . . . . .	26
4.15	Widok harmonogramu dla managera . . . . .	26
4.16	Warianty widoku dodawania kart . . . . .	27
4.17	Widoki dodawania kart . . . . .	28
4.18	Komunikaty widoku dodawania kart . . . . .	28
4.19	Widok strony konfiguracyjnej czytnika. . . . .	29
5.1	Diagram architektury systemu . . . . .	31
5.2	Uproszczony schemat bazy danych . . . . .	33
5.3	Schemat tabel użytkowników . . . . .	33
5.4	Schemat tabel jednostek organizacyjnych . . . . .	34
5.5	Schemat tabel harmonogramów . . . . .	35
5.6	Schemat tabel kart dostępowych . . . . .	35
5.7	Diagram klas modeli . . . . .	37
5.8	Diagram sekwencji logowania . . . . .	40
5.9	Diagram sekwencji procesu uwierzytelniania użytkownika . . . . .	41
5.10	Diagram sekwencji odczytu harmonogramu użytkownika . . . . .	43
5.11	Diagram sekwencji odczytu i przypisania karty dostępowej . . . . .	44

5.12 Schemat połączeń czytnika kart NFC . . . . .	46
5.13 Prototyp czytnika kart na płytce stykowej . . . . .	46

# Spis tabel

3.1	Biblioteki i frameworki wykorzystane w części backend . . . . .	12
3.2	Biblioteki wykorzystane w programie układu mikroprocesorowego . . . . .	13
3.3	Biblioteki i frameworki wykorzystane w części frontend oraz aplikacji mobilnej . . . . .	14
3.4	Narzędzia wykorzystane podczas tworzenia systemu . . . . .	15