



**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

Słowa kluczowe

Thesis title

Workforce management system

Abstract

Key words

Spis treści

1 Wstęp	1
1.1 Wprowadzenie do problematyki	1
1.2 Cel pracy	2
1.3 Zakres pracy	3
1.4 Charakterystyka rozdziałów	3
2 Analiza tematu	5
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
3 Wymagania i narzędzia	9
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	13
4 Specyfikacja zewnętrzna	15
4.1 Użytkownicy i ich role	15
4.1.1 Użytkownik	15
4.1.2 Manager	15
4.1.3 Administrator	16
4.2 Wygląd interfejsu użytkownika	16
4.2.1 Kolorystyka	16
4.2.2 Struktura widoku	17
4.2.3 Strona logowania	17

4.2.4	Panel główny	18
4.2.5	Widok użytkowników	18
4.2.6	Widok timesheetu	20
4.2.7	Widok jednostek organizacyjnych	21
4.2.8	Widok harmonogramu	24
4.2.9	Widok dodawania karty	25
4.3	Obsługa czytnika kart	26
4.3.1	Uruchomienie	26
4.3.2	Odczyt karty	27
4.4	Dostępność	27
4.4.1	Dostępność cyfrowa	27
4.4.2	Dostępność językowa	27
5	Specyfikacja wewnętrzna	29
5.1	Architektura systemu	29
5.1.1	Aplikacja webowa	29
5.1.2	Serwer	29
5.1.3	Aplikacja mobilna	30
5.1.4	Układ mikroprocesorowy	30
5.2	Struktura bazy danych	30
5.2.1	Tabele użytkowników	31
5.2.2	Tabele jednostek organizacyjnych	31
5.2.3	Tabele harmonogramów	32
5.2.4	Tabele kart dostępowych	33
5.2.5	Tabele słownikowe	33
5.3	Modele i struktury danych	34
5.3.1	Klasy encji	34
5.3.2	Klasy pomocnicze	34
5.3.3	Klasy DTO	35
5.4	Algorytmy	35
5.4.1	Rejestracja i pierwsze logowanie użytkownika	35
5.4.2	Logowanie	36
5.4.3	Uwierzytelnianie użytkownika	37
5.4.4	Harmonogramowanie	37
5.4.5	Przypisanie karty dostępowej	39
5.4.6	Uruchomienie czytnika kart	40
5.5	Połączenie modułów czytnika kart	40

6 Weryfikacja i walidacja	43
6.1 Walidacja danych	43
6.1.1 Walidacja po stronie klienta	43
6.1.2 Walidacja po stronie serwera	44
6.2 Testy	44
7 Podsumowanie i wnioski	45
7.1 Wnioski	45
7.1.1 Problemy napotkane podczas pracy	46
7.1.2 Ocena dobranych technologii po zakończeniu pracy	46
7.2 Perspektywy rozwoju	46
7.3 Podsumowanie	47
Bibliografia	51
Spis skrótów i symboli	55
Źródła	57
Lista dodatkowych plików, uzupełniających tekst pracy	59
Spis rysunków	61
Spis tabel	63

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 często 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. Niestety, taka hierarchia uniemożliwia skuteczne zarządzanie, ponieważ występują w niej zbyt duże obszary odpowiedzialności i skorelowanie działów. [4] W takiej sytuacji konieczne jest wprowadzenie bardziej skomplikowanej struktury organizacyjnej, która pozwoli na lepsze zarządzanie firmą. Poniżej wymieniono najczęściej spotykane struktury organizacyjne. [36]

- 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, który jest odpowiedzialny za całość działu.

- Struktura sztabowo-liniowa - jest połączeniem struktury liniowej oraz funkcjonalnej. Zespół składa się z jednego kierownika wspieranego przez kilku specjalistów oraz 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 wytworzyć 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 skuteczne zarządzanie firmą.

Kolejną krytyczną kwestią w zarządzaniu firmą jest kontrola czasu pracy pracowników. Wpływa ona na efektywność pracy, a także na zadowolenie pracowników. W zależności od wielkości firmy oraz od 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, 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. [38]

1.2 Cel pracy

Celem pracy jest analiza dziedziny, stwierdzenie zasadności stworzenia systemu, jego projekt oraz implementacja. Docelowo system ma składać się 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 oraz strukturą firmy. Dodatkowo ma umożliwiać rejestrację czasu pracy pracowników oraz kontrolę dostępu do pomieszczeń. System ma być kompleksowym narzędziem do zarządzania personelem w firmie, które pozwoli na zautomatyzowanie wielu procesów, zwiększenie efektywności pracy oraz przejrzystość ról w firmie.

1.3 Zakres pracy

W ramach pracy zostanie przeprowadzona analiza dziedziny, na podstawie której zostanie stworzony projekt systemu. Następnie zostaną wyznaczone wymagania funkcjonalne i niefunkcjonalne, które musi spełniać system oraz wykonany zostanie jego projekt graficzny. Po zakończeniu fazy projektowania zostaną rozpoczęte prace nad implementacją kompletnego systemu. Zakończeniu prac będzie towarzyszyło przeprowadzenie testów oraz ewentualne poprawki.

1.4 Charakterystyka rozdziałów

Praca składa się z wielu rozdziałów, z których każdy ma określone zadanie. Poniżej przedstawiono krótką charakterystykę każdego z nich.

Rozdział 2 - Analiza tematu

W rozdziale zostanie przeprowadzony przegląd dostępnych rozwiązań oraz analiza dziedziny. Zostanie określona zasadność stworzenia systemu oraz przedstawione zostaną najważniejsze funkcjonalności, które powinien on posiadać.

Rozdział 3 - Wymagania i narzędzia

Rozdział zawiera w sobie wszelkie założenia, jakie musi spełniać system, aby był użyteczny. Zostaną w nim opisane wymagania funkcjonalne i niefunkcjonalne, a także narzędzia i technologie, które zostaną użyte do implementacji systemu.

Rozdział 4 - Specyfikacja zewnętrzna

Wszelkie informacje dotyczące wyglądu i działania systemu, które są widoczne dla użytkownika, znajdują się w tym rozdziale. Zostaną w nim opisane role użytkowników,

dostępne funkcjonalności oraz sposób ich użycia. Znajdzie się w nim również opis interfejsu graficznego oraz obsługi czytnika kart zbliżeniowych.

Rozdział 5 - Specyfikacja wewnętrzna

Rozdział zawiera w sobie informacje dotyczące implementacji systemu oraz jego architektury. Znajdą się w nim opisy najważniejszych klas, algorytmów, bazy danych i struktury połączeń układu czytnika kart zbliżeniowych. Wyszczególnione zostaną również najważniejsze algorytmy zapewniające poprawne działanie systemu.

Rozdział 6 - Weryfikacja i walidacja

W rozdziale zostaną opisane testy, które zostały przeprowadzone w celu sprawdzenia poprawności działania systemu oraz sposoby walidacji danych wejściowych.

Rozdział 7 - Podsumowanie i wnioski

W ostatnim rozdziale pracy zostaną przedstawione wnioski wynikające z przeprowadzonych prac, problemy jakie w ich trakcie wystąpiły oraz 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** - 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 pracy zapisywane są godziny pracy pracowników oraz zadania jakie 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. Podobnie jak w przypadku kart pracy, konieczne jest przechowywanie i archiwizacja dokumentów.
- **Harmonogram pracy** - pozwala przełożonym na ustalenie grafiku pracy pracowników. Jest on następnie przekazywany podwładnym, którzy muszą przestrzegać ustalonych godzin pracy. Ta metoda wymaga wykorzystania dodatkowych narzędzi, takich jak karty pracy. Największą wadą tej metody 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 podatny na błędy. Korzystanie z papierowych dokumentów jest obarczone bardzo dużym ryzykiem utraty danych, nieautoryzowanego dostępu oraz błędów ludzkich.

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*. 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**[24] - 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, czy Fujitsu.
- **SAP SuccessFactors HCM**[31] - 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, Nestle, Allianz.
- **MintHCM**[19] - oprogramowanie firmy eVolpe oparte o otwartoźródłowe systemy 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, czy Asseco.

Główym powodem, dla którego firmy decydują się wdrożyć systemy HCM jest ich pozytywny wpływ na efektywność pracy, a co za tym idzie - zwiększenie przychodów. Dobrze zaprojektowany system, który pozwala na załatwienie wielu spraw formalnych oraz administracyjnych w jednym miejscu ułatwia pracownikom codzienną pracę, pozwala na szybsze reagowanie na zmiany w organizacji i daje jasny wgląd do danych dotyczących ich wydajności. Dla kadry, system umożliwia monitorowanie działań i wyników pracowników, co może przełożyć się na premie i awanse.

Często w rozwiązaniach HCM brakuje funkcji związanej z przyznawaniem dostępów oraz kontrolą wejść i wyjść pracowników z firmy. W takich przypadkach konieczne jest zintegrowanie systemu HCM z systemem kontroli dostępu, co zwiększa koszty i skomplikowanie systemu. Dodatkowo, systemy HCM są zazwyczaj dostępne jedynie w formie

chmurowej, co może być 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 wejść i wyjść pracowników oraz gości. Ich głównym zadaniem jest zapewnienie bezpieczeństwa pracownikom oraz ochrona mienia firmy. Pozwalają one na identyfikację osób przemieszczających się po budynku oraz na kontrolę dostępu do poszczególnych pomieszczeń. Zdarza się, że systemy są zintegrowane z alarmami oraz monitoringiem. Zazwyczaj spotyka się je w dużych firmach, w których kontrola dostępu jest kluczowym elementem bezpieczeństwa. Takie systemy dostarczają m. in. firmy:

- **Satel** - polska firma zajmująca się produkcją systemów alarmowych, monitoringowych i kontroli dostępu, której rozwiązania opierają się o technologię RFID. Możliwe jest ich wdrożenie lokalne oraz rozproszone,
- **Avigilon** - firma zajmująca się produkcją systemów monitoringu i kontroli dostępu. Ich rozwiązania opierają się głównie na technologiach bezprzewodowych oraz pin-padach.

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 czasem pracy. W takich przypadkach konieczne jest zintegrowanie systemu kontroli dostępu z systemem HCM, co zwiększa koszty i skomplikowanie systemu.

2.1.4 Podsumowanie

Analiza dziedziny pozwala na stwierdzenia, 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** - pozwala na kontrolę dostępu pracowników do budynku. Moduł powinien umożliwiać zarządzanie kartami dostępowymi, kontrolować wejścia i wyjścia pracowników, a także umożliwiać nadawanie uprawnień dostępu do poszczególnych pomieszczeń.

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

3.1.2 Wymagania niefunkcjonalne

Projekt systemu zakłada spełnienie następujących wymagań niefunkcjonalnych:

- system dostępny jest przez 24 godziny na dobę, 7 dni w tygodniu, z wyjątkiem przerw na konserwację,
- aplikacja działa płynnie i bez zacięć, nawet przy dużej ilości użytkowników,
- system jest bezpieczny i odporny na ataki z zewnątrz,
- aplikacja mobilna działa na systemach Android i iOS,
- aplikacja jest intuicyjna i łatwa w obsłudze,
- system jest łatwy w utrzymaniu i rozbudowie,
- system jest zgodny z obowiązującymi przepisami prawa.

3.2 Projekt systemu

3.2.1 Technologie

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

MySQL

MySQL to relacyjna baza danych rozwijana aktualnie przez firmę Oracle. Jej najważniejszymi cechami są: popularność, szybkość, niezawodność oraz łatwość w obsłudze. 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. [21]

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 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, takie jak: transakcje, bezpieczeństwo, obsługa danych, integracja z bazami danych, obsługa REST API, itp. [33]

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

Vue.js

Vue.js 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 któremu użytkownik widzi ją jako pojedynczą stronę internetową. Reakcje na interakcje użytkownika są szybkie i płynne, co zwiększa komfort korzystania z aplikacji przypominając niejako aplikacje desktopowe. [40]

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

Expo

Expo 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 EAS (ang. *Expo Application Service*). [5]

Expo zostało wybrane ze względu na język programowania, który jest znany autorowi pracy, a także na możliwość szybkiego utworzenia i wdrożenia aplikacji na platformy mobilne.

Raspberry Pi Pico W

Raspberry Pi Pico W jest najmniejszym modułem z rodziny Raspberry Pi, który może łączyć się z siecią. Posiada wbudowany kontroler Raspberry RP4020 oparty na architekturze ARM, co czyni go idealnym wyborem do zastosowań IoT. Układ wyposażony jest w 264 KB pamięci RAM, 2 MB pamięci flash oraz 26 pinów GPIO, 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. [26]

MicroPython

MicroPython jest implementacją języka Python przeznaczona dla mikrokontrolerów i systemów wbudowanych, zoptymalizowaną pod kątem niskiego zużycia zasobów - co czyni ją idealnym wyborem dla urządzeń o ograniczonej mocy obliczeniowej i pamięci. MicroPython oferuje większość funkcji standardowego Pythona, co pozwala na szybkie i efektywne tworzenie oprogramowania dla mikrokontrolerów. Dzięki temu istnieje możliwość korzystania z dobrze znanych narzędzi i bibliotek, co znaczaco przyspiesza proces tworzenia i testowania programu. [17]

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

Inne technologie

W projekcie wykorzystano również mnóstwo znaczących technologii, 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 [32]	Spring	Apache License 2.0
Spring Security	Framework do zarządzania bezpieczeństwem aplikacji [34]	Spring	Apache License 2.0
Springdoc OpenAPI	Biblioteka do generowania dokumentacji API oraz dostępu do Swagger-ui [35]	Springdoc	Apache License 2.0
MySQL Connector	Sterownik do łączenia się z bazą danych MySQL [22]	Oracle	GPL 2.0
Lombok	Biblioteka do generowania kodu Java [15]	Project Lombok	MIT
JJWT	Biblioteka do obsługi tokenów JWT [12]	jwtk	Apache License 2.0

Tabela 3.2: Biblioteki i frameworki wykorzystane w części frontend oraz aplikacji mobilnej

Technologia	Opis	Autor	Licencja
-------------	------	-------	----------

Axios	Biblioteka do wykonywania zapytań HTTP [1]	Axios	MIT
Heroicons	Zestaw ikon SVG [13]	Tailwind Labs	MIT
PrimeVue	Biblioteka komponentów do Vue.js [28]	PrimeTek	MIT
Luxon	Biblioteka do obsługi dat i czasu [20]	Moment.js	MIT
Valibot	Biblioteka do walidacji formularzy [9]	Fabian Hiller	MIT
Vue i18n	Biblioteka do obsługi wielojęzyczności [10]	Intlify	MIT
Vue Router	Biblioteka do zarządzania trasami w aplikacji Vue.js [39]	Vue	MIT
TailwindCSS	Biblioteka do stylowania aplikacji internetowych [14]	Tailwind Labs	MIT
Nativewind	Biblioteka umożliwiająca używanie TailwindCSS w aplikacjach opartych o React Native [23]	Nativewind	MIT
React Native Gesture Handler	Biblioteka do obsługi gestów w aplikacjach opartych o React Native [16]	Software Mansion	MIT
react native nfc manager	Biblioteka do obsługi NFC w aplikacjach opartych o React Native [30]	RevtelTech	MIT

Tabela 3.3: Biblioteki wykorzystane w programie układu mikroprocesorowego

Technologia	Opis	Autor	Licencja
mfrc522	Biblioteka do obsługi modułu RFID [25]	Daniel Perron	MIT
picozero	Biblioteka do obsługi modułów peryferyjnych z Raspberry Pi Pico [7]	Raspberry Pi Foundation	MIT
Requests	Biblioteka do wykonywania zapytań HTTP w Python [29]	Python Software Foundation	Apache License 2.0

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 [3]	Docker Inc.
Git	System kontroli wersji [8]	Linus Torvalds
Visual Studio Code	Edytor kodu wykorzystany przy tworzeniu części frontend oraz aplikacji mobilnej [18]	Microsoft
IntelliJ IDEA	Zintegrowane środowisko programistyczne do języków Java oraz Kotlin, wykorzystane przy tworzeniu części backend [11]	JetBrains
Figma	Narzędzie do projektowania interfejsów użytkownika [6]	Figma
Zeplin	Narzędzie do współpracy nad projektami interfejsów użytkownika, umożliwiające generowanie stylów CSS [41]	Zeplin
Thonny	Zintegrowane środowisko programistyczne dla języka Python na mikrokontrolery [37]	Thonny
PlantUML	Narzędzie do tworzenia diagramów UML [bib:plantuml]	PlantUML

Rozdział 4

Specyfikacja zewnętrzna

4.1 Użytkownicy i ich role

W systemie zostały zdefiniowane 3 role użytkowników:

- Użytkownik,
- Manager,
- Administrator.

Każda rola posiada inne uprawnienia i możliwości w systemie. Każda rola posiada przypisany do siebie kolor, który umożliwia łatwe rozróżnienie użytkowników w systemie. Widoczny jest on na pasku nawigacyjnym aplikacji mobilnej i na stronie internetowej, a także w tabelach użytkowników.

Dodatkowo, każde konto użytkownika może zostać zarchiwizowane, co oznacza, że użytkownik nie ma dostępu do systemu. Wiąże się to również z usunięciem danych, które są uregulowane przez 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, które posiada w firmie. Może przeglądać swoje dane, harmonogram pracy, oraz dodawać wpisy do timesheetu.

Kolorem przypisanym do użytkownika jest kolor pomarańczowy.

4.1.2 Manager

Manager posiada większe uprawnienia niż użytkownik. Oprócz możliwości przeglądania swoich danych, harmonogramu pracy, oraz dodawania wpisów do timesheetu, manager

może również zarządzać przypisanym do siebie zespołem, harmonogramem pracy zespołu oraz przeglądać i akceptować timesheety swoich pracowników. Manager nie ma możliwości zarządzania innymi managerami, administratorami oraz strukturą organizacyjną firmy.

Kolorem przypisanym do managera jest zielony.

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, edycję i usuwanie działów oraz stanowisk - oraz konfigurację systemu. Administrator ma dostęp do wszystkich danych w systemie.

Kolorem przypisanym do administratora jest niebieski.

4.2 Wygląd interfejsu użytkownika

Interfejs użytkownika został zaprojektowany w sposób przejrzysty i intuicyjny. Przy etapie projektowania kierowano się tym, aby użytkownik mógł szybko i sprawnie dowiedzieć się tego, co go interesuje. Zadbano, aby interfejs nie był zbyt przeładowany informacjami, a jednocześnie nie brakowało w nim żadnych ważnych elementów. Pierwszym widokiem jaki widzi użytkownik po zalogowaniu się do systemu jest panel główny.

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 odcięń do oznaczania elementów informacyjnych. Kolorami używanymi w systemie są:

- #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 4.1.

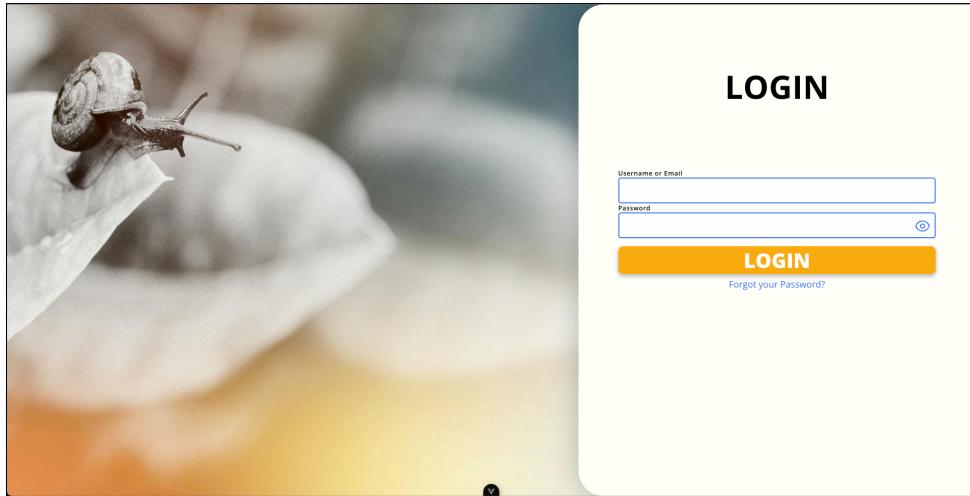


Rysunek 4.1: Makieta struktury widoku

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

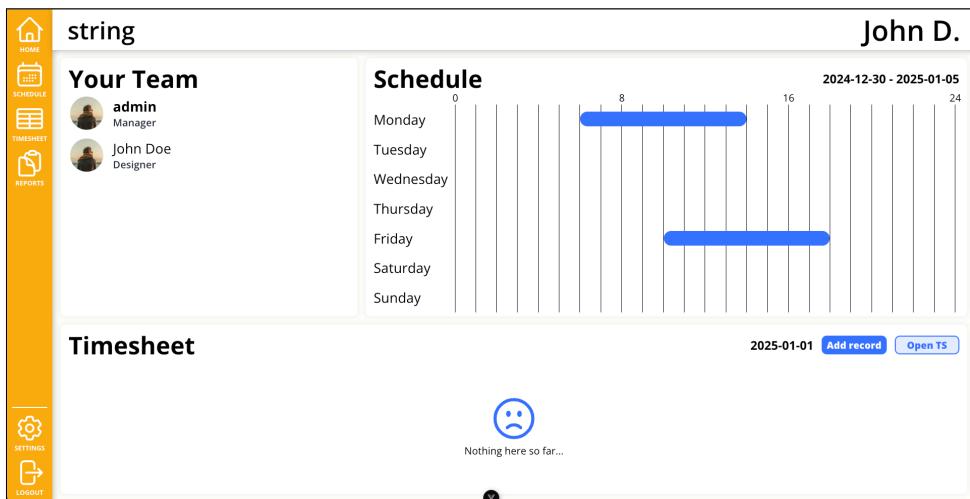
Logowanie do systemu odbywa się poprzez bardzo prosty formularz, w którym użytkownik musi podać swoją nazwę użytkownika oraz hasło. W przypadku, gdy użytkownik nie ma jeszcze ustawionego hasła pojawia się dodatkowe pole na ponowne wpisanie hasła, które - jeżeli jest zgodne z wcześniejszym wpisem - zostanie zapamiętane w systemie. Wszelkie błędy związane z logowaniem są wyświetlane w formie komunikatów pod polami formularza. Widok strony logowania został przedstawiony na rysunku 4.2.



Rysunek 4.2: 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 timesheetu. W zależności od roli użytkownika, poszczególne karty panelu mogą różnić się nagłówkami lub treścią. Przykładowy panel dla roli użytkownika został przedstawiony na rysunku 4.3.



Rysunek 4.3: 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 ostatnia 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 4.4. Kliknięcie w ikony akcji otwiera odpowiednie okna dialogowe, szuflady lub wykonuje ją bezpośrednio. Akcje krytyczne z punktu widzenia bezpieczeństwa, takie jak usunięcie konta, wymagają potwierdzenia w osobnym oknie dialogowym.

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.

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@nowak.pl	USER	
6	Jacek				USER	
7	new				USER	
11	newUser				NEW USER	

Rysunek 4.4: Widok użytkowników

(a) Szczegóły użytkownika

(b) Edycja użytkownika

Rysunek 4.5: Okna dialogowe widoku użytkowników

4.2.6 Widok timesheetu

Widok timesheetu 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. Widok timesheetu został przedstawiony na rysunku 4.6. Dodawanie wpisów odbywa się poprzez formularz otwierany w oknie dialogowym po kliknięciu odpowiedniego przycisku. Został on ukazany na rysunku 4.7.

Date	Unit	Duration	Description	Approved
2025-01-04	asd	1	asdafgg	Yes
2025-01-05	IPSUM	1	Task #11100	Yes
2025-01-05	IPSUM	1	Bugfixes	No

Rysunek 4.6: Widok timesheetu

The screenshot shows a modal dialog box titled "Add record". Inside, there are four input fields: "Unit*" (dropdown menu showing "Select a unit"), "Duration*" (input field containing "1"), "Date*" (input field containing "2025-01-06"), and "Description*" (text area with a character count of "0/255"). At the bottom right of the dialog are two buttons: "Save" (green) and "Cancel" (red).

Rysunek 4.7: Okno dialogowe dodawania wpisu do timesheetu

Administratorzy i managerowie mają możliwość przejścia do widoku timesheetów 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 timesheetów pojawiają się odpowiednie oznaczenia. Widok wpisów oczekujących na akceptację został przedstawiony na rysunku 4.8.

The screenshot shows a dashboard titled "AIPSUM1". On the left is a sidebar with icons for HOME, SCHEDULE, TIMESHEET (highlighted in green), REPORTS, TEAMS, SETTINGS, and LOGOUT. The main content area displays a table titled "Pending" with the following data:

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

To the right, a sidebar for "manager m." includes buttons for "Approve all" and "Show my", and a table of actions:

Actions
Approved
Rejected

Rysunek 4.8: Widok timesheetów oczekujących na akceptację

4.2.7 Widok jednostek organizacyjnych

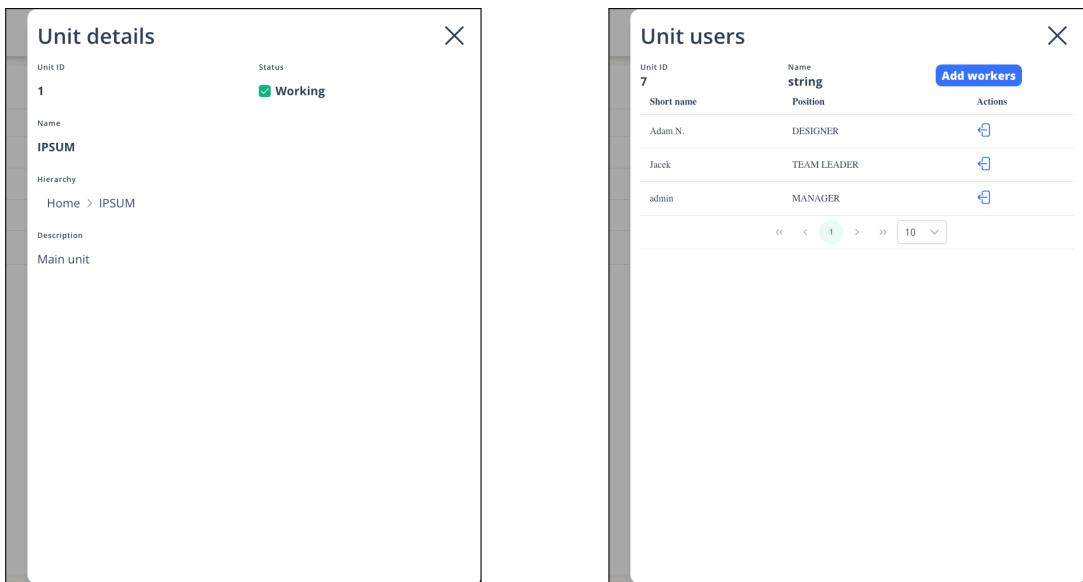
Widok został skonstruowany w taki sposób, aby nie ukazywać wszystkich informacji na raz. Hierarchiczna struktura jednostek organizacyjnych pozwala na wyświetlenie jedynie tych, które są bezpośrednio zależne od wybranej jednostki. Jednostki, które wyświetlają

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 4.9.

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

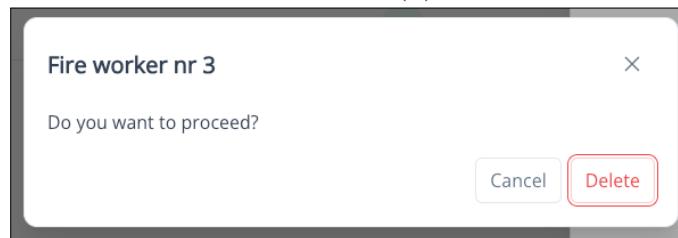
Rysunek 4.9: Widok jednostek organizacyjnych

Kolumna akcji może zawierać maksymalnie cztery ikony: otwarcie szczegółów, edycję, przejście do podjednostek oraz otwarcie widoku pracowników. 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 zamknięte, a tabela jest odświeżana. Otwierane okna są ukazane na rysunku 4.10.



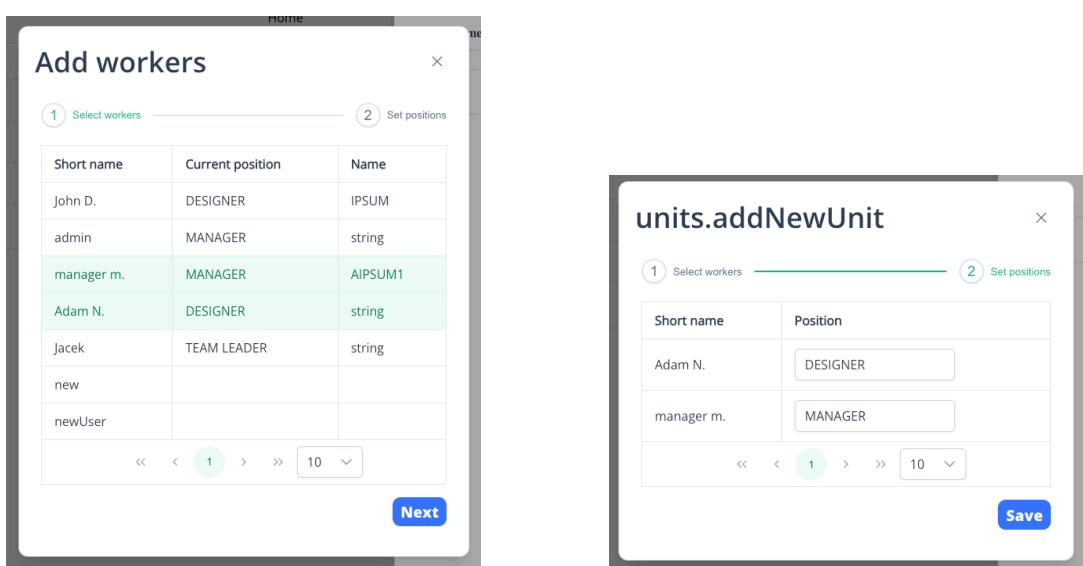
(a) Szczegóły jednostki organizacyjnej

(b) Pracownicy jednostki organizacyjnej



(c) Potwierdzenie usunięcia użytkownika

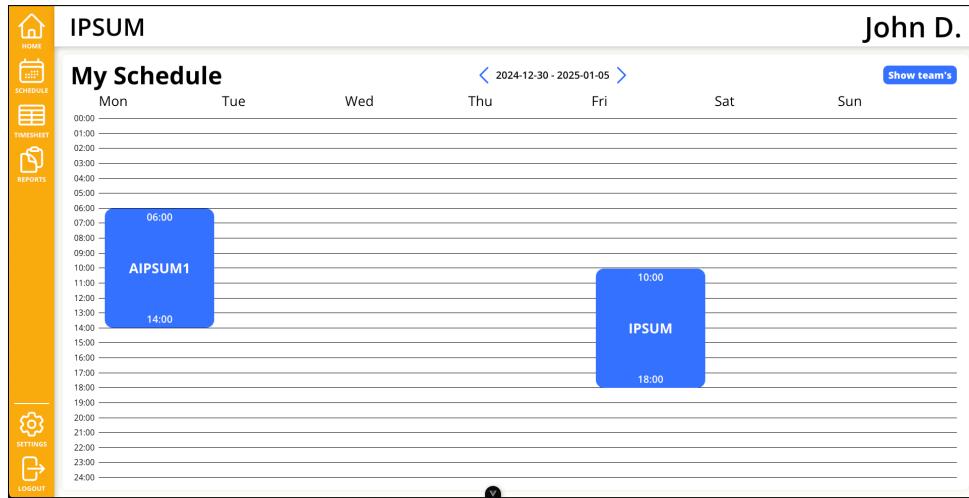
Rysunek 4.10: Okna dialogowe widoku jednostek organizacyjnych



(a) Wybór pracowników

(b) Ustawienie ról

Rysunek 4.11: Okna dialogowe dodawania użytkowników do jednostki organizacyjnej

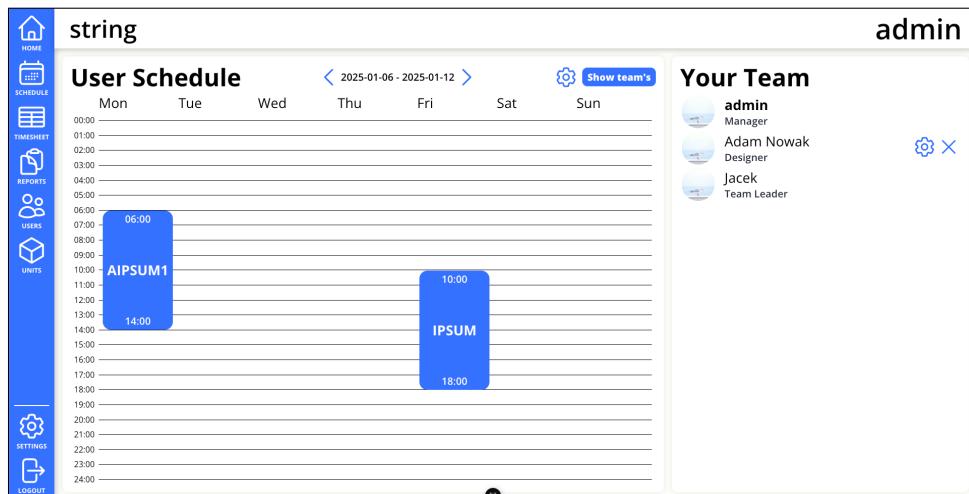


Rysunek 4.12: Widok harmonogramu

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 łatwe ich odczytanie. Dodatkowo, każdy z nich posiada informacje o godzinach rozpoczęcia i zakończenia pracy oraz nazwę zespołu. Widok harmonogramu użytkownika został przedstawiony na rysunku 4.12.

Administratorzy oraz managerowie mają możliwość podglądu harmonogramu użytkowników należących do ich zespołu. Odbywa się to poprzez wybór odpowiedniego użytkownika z dodatkowej karty wyświetlonej po prawej stronie widoku. Ikony kół zębatych umożliwiają edycję harmonogramu. Widok harmonogramu dla managera został przedstawiony na rysunku 4.13.



Rysunek 4.13: Widok harmonogramu dla managera

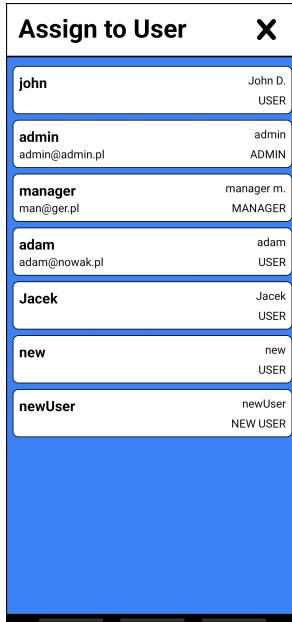
4.2.9 Widok dodawania karty

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 jej dotyczące. Widok jest różny w zależności od tego, czy karta jest już przypisana do użytkownika, czy nie. Jeżeli w urządzeniu jest wyłączona funkcja NFC, aplikacja wyświetla komunikat o konieczności jej włączenia ukazany na rysunku 4.16a. Widoki zostały przedstawione na rysunku 4.14.

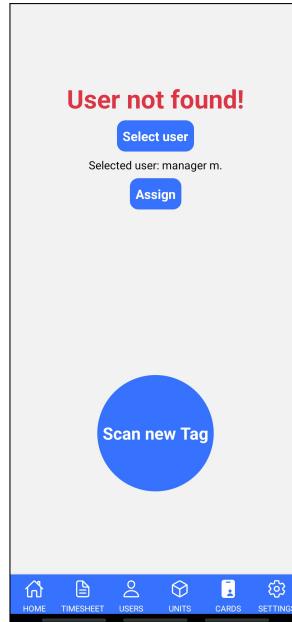


Rysunek 4.14: Warianty widoku dodawania kart

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 do użytkownika, a administrator otrzymuje informację zwrotną o powodzeniu operacji. Widoki przypisania karty zostały przedstawione na rysunkach 4.15, a komunikat o powodzeniu operacji na rysunku 4.16b.

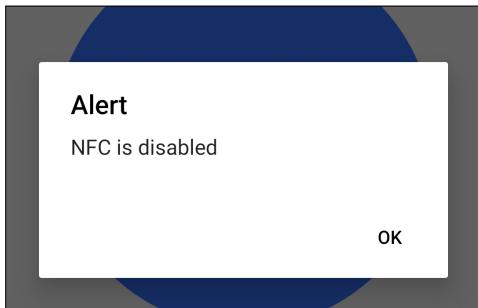


(a) Lista użytkowników

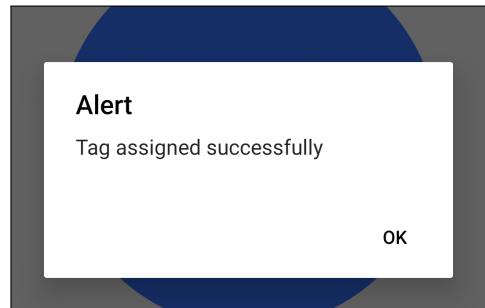


(b) Wybrany użytkownik

Rysunek 4.15: Widoki dodawania kart



(a) Komunikat o konieczności włączenia NFC



(b) Komunikat o powodzeniu operacji

Rysunek 4.16: Komunikaty widoku dodawania kart

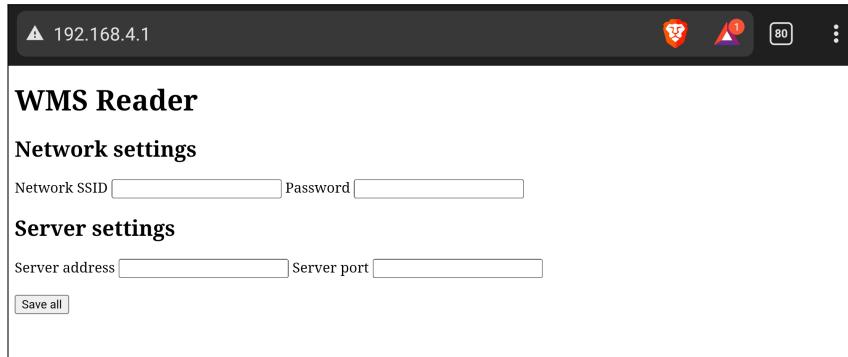
4.3 Obsługa czytnika kart

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 świeci się na żółto. Jeżeli czytnik napotka jakikolwiek problem podczas uruchamiania, dioda LED zacznie świecić na zielono, a mikrokontroler udostępni sieć bezprzewodową, do której można się połączyć i otworzyć stronę konfiguracyjną. Widok strony konfiguracyjnej został przedstawiony na rysunku 4.17.

Po poprawnym uruchomieniu, dioda LED zmienia kolor na niebieski.



Rysunek 4.17: Widok strony konfiguracyjnej czytnika.

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 powiodł - 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 wyłącznie przy użyciu klawiatury. Użycie biblioteki komponentów PrimeVue pozwoliło na zapewnienie dostępności dla osób o ograniczonych możliwościach ruchowych. Komponenty umieszczone na stronie internetowej są zgodne z wytycznymi WCAG 2.1. [2]

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 zawierającej 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, które komunikują się ze sobą w celu zapewnienia pełnej funkcjonalności. Poniżej zostały przedstawione opisy każdego z nich.

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 roli użytkownika, umożliwia wykonanie innych czynności. Aplikacja została zaprojektowana w taki sposób, aby była przejrzysta i intuicyjna w obsłudze. Dzięki temu, użytkownik może szybko i sprawnie wykonywać swoje codzienne obowiązki. Odrębne od siebie panele zostały umieszczone w kartach pojawiających się na widoku, dzięki czemu użytkownik wie dokładnie w którym miejscu aplikacji się znajduje i jakie ma możliwości. Frontend komunikuje się z serwerem aplikacyjnym wysyłając żądania HTTP i odbierając odpowiedzi w formacie JSON.

5.1.2 Serwer

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

5.1.3 Aplikacja mobilna

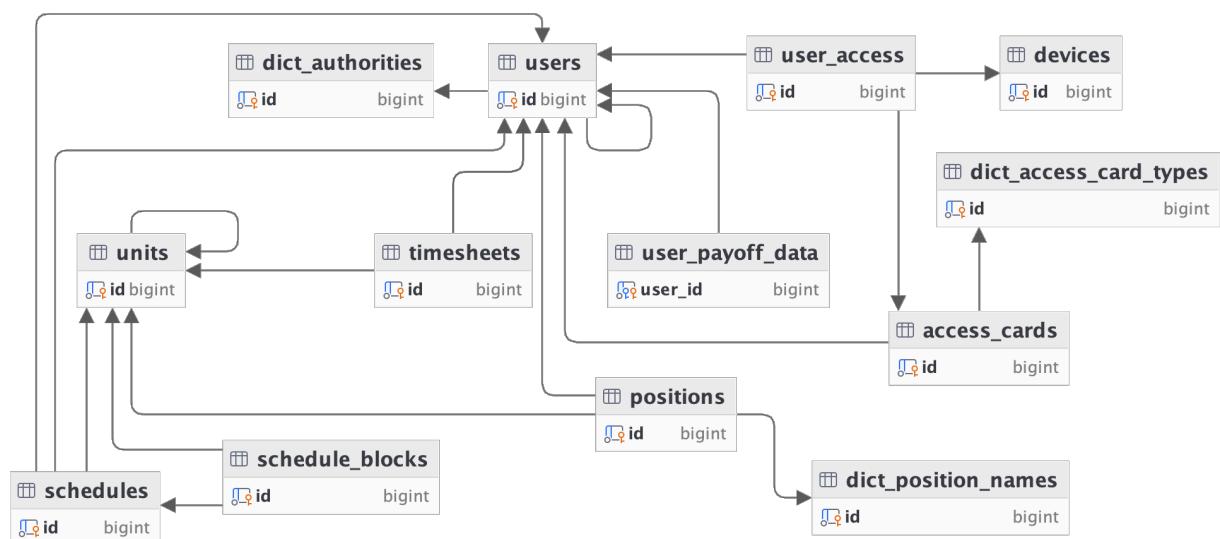
Aplikacja mobilna umożliwia użytkownikom przeglądanie swoich danych bez konieczności korzystania z przeglądarki internetowej. Jej głównym zadaniem jest jednak możliwość dodawania kart dostępowych dla poszczególnych użytkowników. Odbywa się to poprzez przyłożenie tagu NFC (ang. *Near Field Communication*) 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 do systemu. Układ mikroprocesorowy jest zasilany z wbudowanego portu Micro USB, co umożliwia bardzo proste podłączenie go do źródła zasilania. Dokumentacja techniczna mikrokontrolera [27] precyzuje również podłączenie innego źródła zasilania bez użycia wbudowanego portu.

5.2 Struktura bazy danych

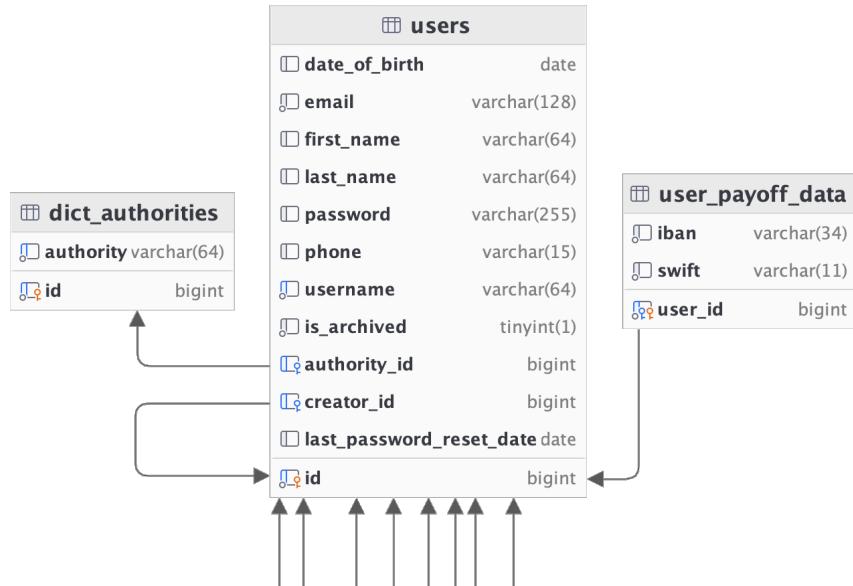
Baza danych obsługująca system składa się z kilkunastu tabel, które przechowują informacje o wszelkich danych w systemie. Na rysunku 5.1 został przedstawiony jej uproszczony schemat, a w kolejnych podrozdziałach zostaną opisane najważniejsze tabele oraz ich relacje.



Rysunek 5.1: Uproszczony schemat bazy danych

5.2.1 Tabele użytkowników

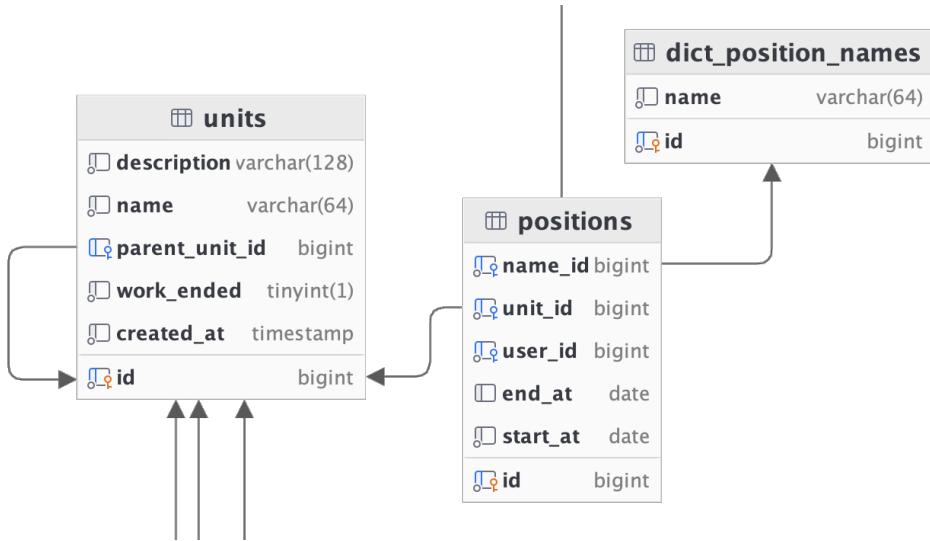
Główna tabelą przechowującą informacje o użytkownikach jest tabela **USERS**. Zawiera ona dane personalne, takie jak imię, nazwisko, adres e-mail i numer telefonu. Dodatkowo do tabeli wpisane są dane do logowania: nazwa użytkownika i hasło; oraz informacja o użytkowniku, który utworzył dany wpis. Każdy użytkownik ma przypisana jedną z ról z tabeli **DICT_AUTHORITIES**, która określa jego uprawnienia w systemie - w relacji wiele do jednego. 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 5.2.



Rysunek 5.2: Schemat tabel użytkowników

5.2.2 Tabele jednostek organizacyjnych

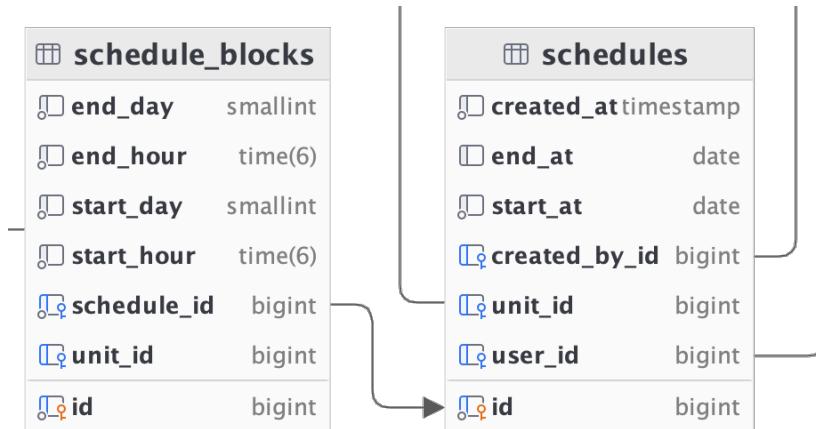
Wszystkie jednostki organizacyjne są przechowywane w tabeli **UNITS** zawierającej dane o nazwie jednostki, jej opisie, jednostce nadzędnej oraz dacie utworzenia. Pole **work-ended** określa, czy jednostka jest aktywna. Użytkownicy są przypisani do jednostki organizacyjnej poprzez tabelę **positions** 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 5.3.



Rysunek 5.3: Schemat tabel jednostek organizacyjnych

5.2.3 Tabele harmonogramów

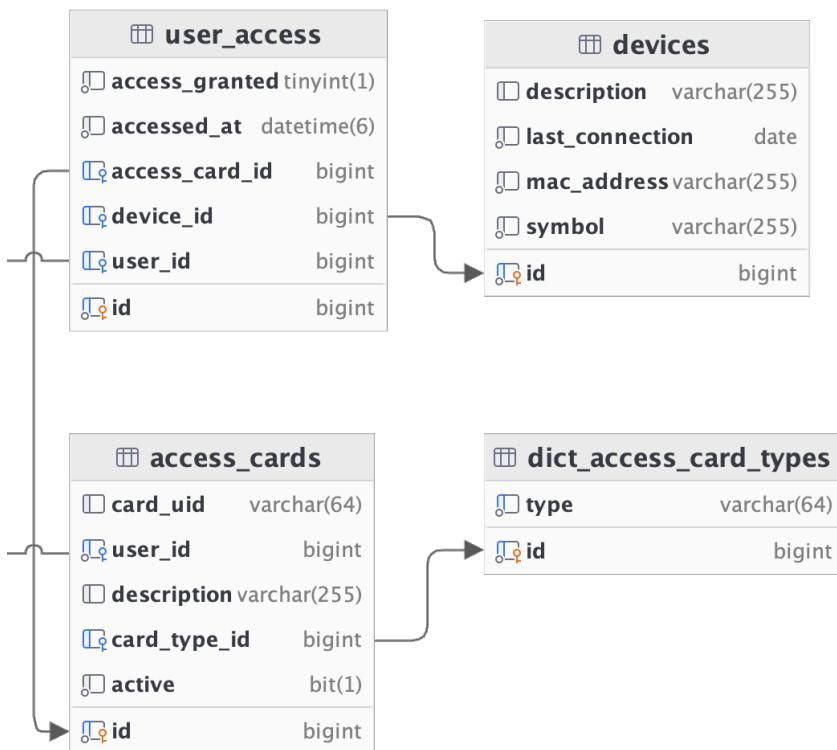
Na każdy z harmonogramów składa się pojedynczy rekord w tabeli **SCHEDULES** oraz pewna liczba rekordów w tabeli **SCHEDULE_BLOCKS**. Pierwsza z nich zawiera informacje o dacie rozpoczęcia i zakończenia harmonogramu, jego utworzenia oraz odwołanie do jednostki organizacyjnej lub użytkownika, którego dotyczy. Tabela **SCHEDULE_BLOCKS** odpowiada za przechowywanie pojedynczych bloków czasowych w harmonogramie opisując ich dzień i godzinę rozpoczęcia oraz zakończenia oraz jednostkę, której dotyczą. Tabele są powiązane relacją jeden do wielu - jeden harmonogram może zawierać wiele bloków czasowych. Przedstawienie tabel harmonogramów widoczne jest na rysunku 5.4.



Rysunek 5.4: Schemat tabel harmonogramów

5.2.4 Tabele kart dostępowych

Karty dostępowe użytkowników są przechowywane w tabeli ACCESS_CARDS zawierającej informacje o numerze seryjnym, jej właściwym opisie, typie karty oraz statusie. Dzięki ostatniej z właściwości możliwe jest przypisanie dwóm użytkownikom jednej karty 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 autoryzacji. Wizualizacja tabel kart dostępowych znajduje się na rysunku 5.5.



Rysunek 5.5: 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. Każda z nich poprzedzona jest prefiksem 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*),

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, do której jest mapowana. Każde pole klasy, które odpowiada kolumnie w tabeli oznaczane jest odpowiednią adnotacją. Do generowania metod `get` i `set` używane są adnotacje `@Getter` i `@Setter` z biblioteki `lombok`. W klasach znajdują się również publiczne metody, dzięki którym możliwe było uproszczenie logiki biznesowej.

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.

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 subpakietы odpowiadające kontrolerom, które wymagają przesłania pełnego obiektu. Są to:

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

5.4 Algorytmy

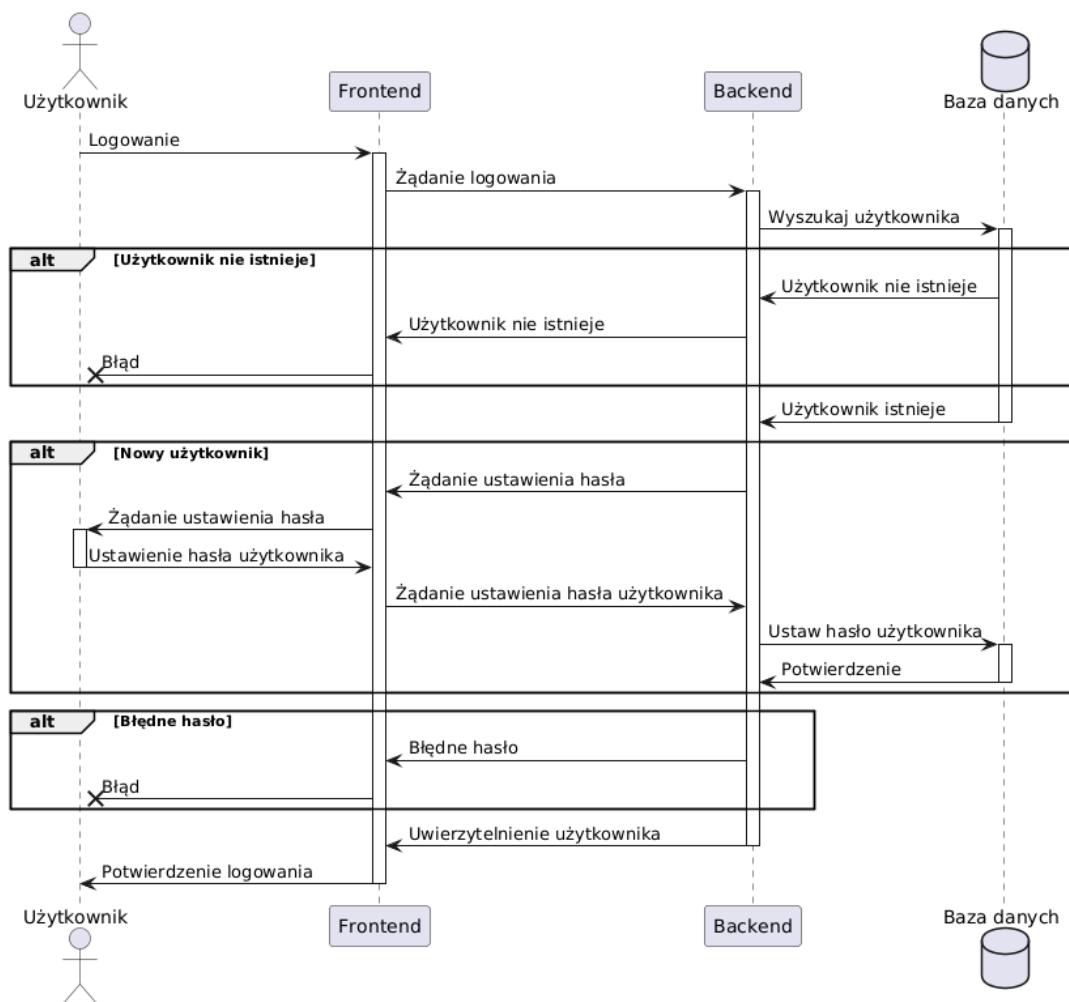
5.4.1 Rejestracja i pierwsze logowanie użytkownika

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, sprawdzi on czy użytkownik o podanej nazwie istnieje w bazie danych i czy ma przypisane hasło. Jeżeli nie, zwróci kod 206

- **Partial Content**, a aplikacja udostępnia 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 5.6, a szczegóły tego procesu zostały opisane w rozdziale 5.4.2

5.4.2 Logowanie

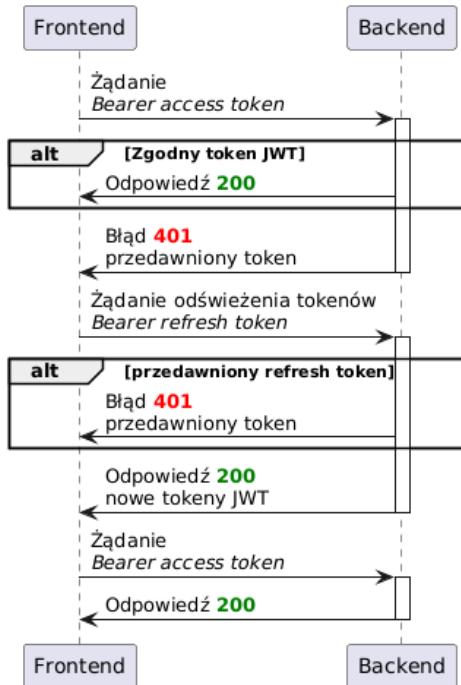
Logowanie do systemu odbywa się poprzez przesłanie żądania HTTP z danymi logowania do serwera aplikacyjnego. Po jego otrzymaniu serwer zwraca się do bazy danych w celu znalezienia użytkownika o podanej nazwie. Jeżeli użytkownik nie istnieje, serwer zwraca kod błędu 401. W przeciwnym wypadku sprawdza, czy zahashowane hasło użytkownika jest zgodne z zapisanym w bazie. Jeżeli hasła się zgadzają, serwer generuje dwa tokeny JWT (ang. *JSON Web Token*) 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 5.6.



Rysunek 5.6: Diagram czynności 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`. Przykładowy nagłówek został przedstawiony na rysunku. Token ten jest ważny przez określony czas, po którym traci ważność - serwer zwraca wtedy kod 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 5.7.



Rysunek 5.7: Diagram sekwencji procesu uwierzytelniania użytkownika

5.4.4 Harmonogramowanie

Harmonogramowanie jest częścią systemu, która pochłania najwięcej zasobów serwera - 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. 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.

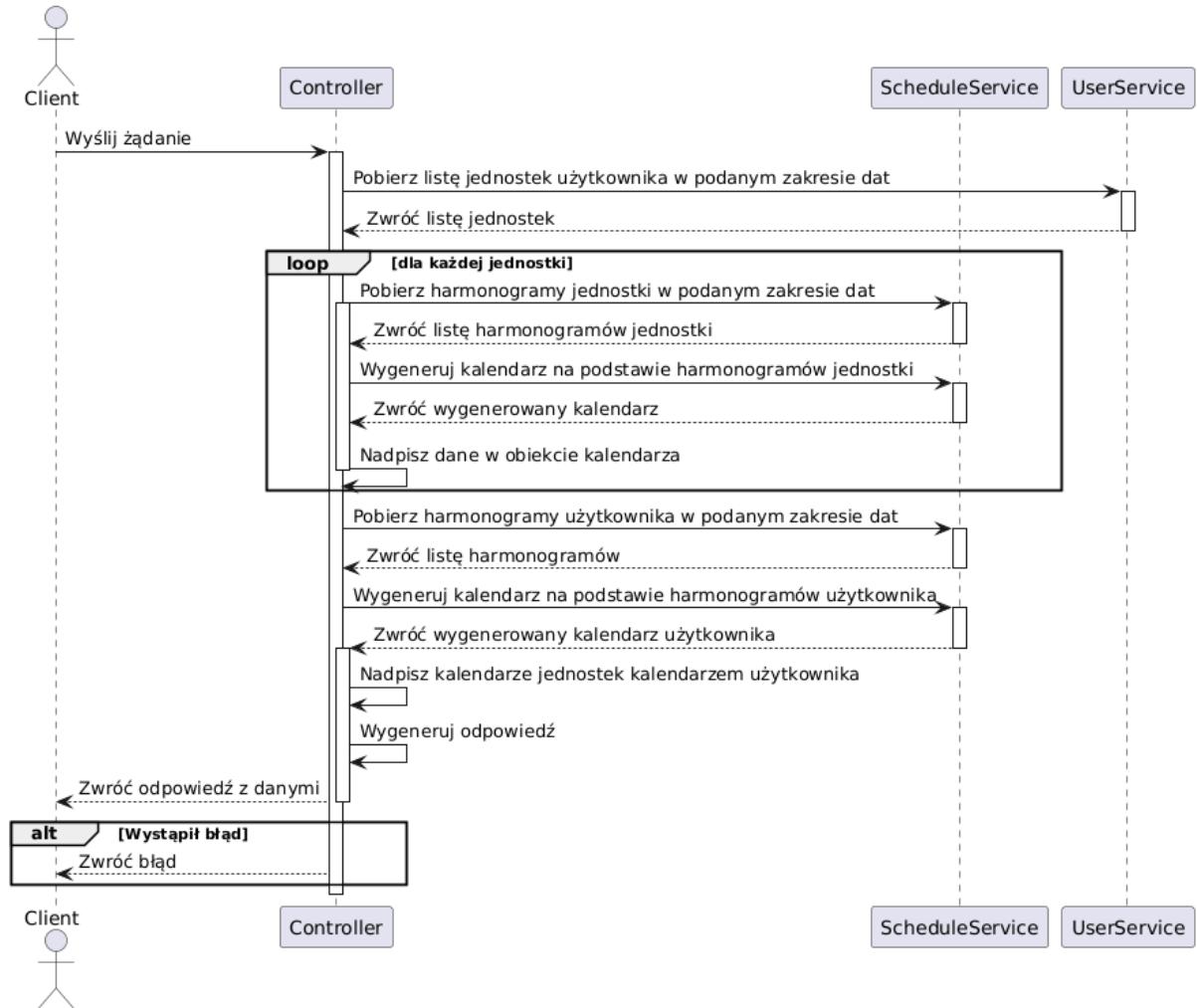
5.4.4.2 Odczyt harmonogramu

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 - pierwszy z nich uwzględnia wyłącznie harmonogramy jednostki, natomiast drugi 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.

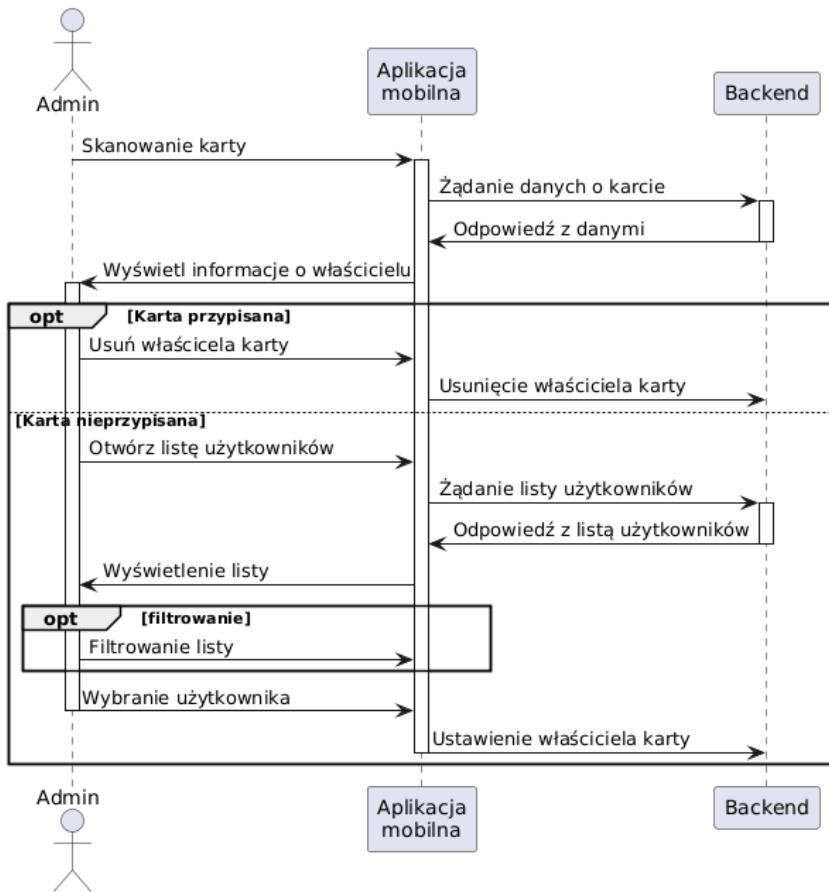
Odczytanie harmonogramu użytkownika jest dużo bardziej złożone niż jednostki organizacyjnej. Wymaga ono odczytu wszystkich jednostek, do których należał użytkownik w danym okresie czasu, następnie dla każdej z nich odczytania harmonogramu i połączenia ich w jeden widok przechowywany w klasie pomocniczej `Calendar`. Jeżeli użytkownik posiada również własny harmonogram, serwer generuje jego widok i nadpisuje dane przechowywane w klasie. Diagram sekwencji tego procesu został przedstawiony na rysunku 5.8.



Rysunek 5.8: 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 jej właścielowi. Jeżeli karta jest już przypisana do użytkownika, aplikacja umożliwi jej usunięcie, a w przeciwnym wypadku udostępni menu wyboru użytkownika, do którego ma zostać przypisana. Po zatwierdzeniu wyboru, aplikacja wyśle do serwera żądanie przypisania karty do użytkownika. Diagram sekwencji przypisania karty dostępowej został przedstawiony na rysunku 5.9.



Rysunek 5.9: 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 4.17. Po zatwierdzeniu, 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 5.10.

Do elementów pasywnych układu należą:

- Rezystory ograniczające prąd,
- Dioda LED RGB ze wspólną katodą, sygnalizująca stan czytnika.

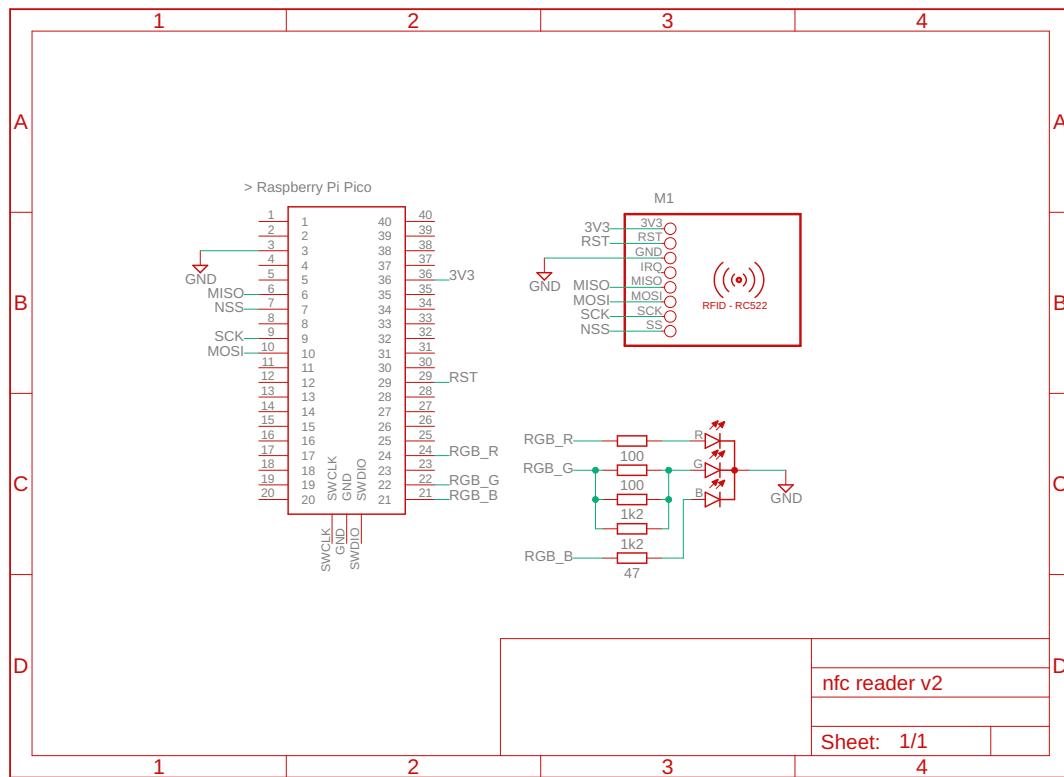
Zgodnie z opisem diody do każdej anody diody podłączony jest rezistor ograniczający prąd. Zgodnie z zaleceniami producenta wartości te powinny wynosić:

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

z dokładnością do 5%. Niemożność uzyskania rezystora dla kanału zielonego spowodowała, że połączono równolegle dostępne rezystory o wartościach 100Ω , $1,2k\Omega$ 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 5.10: Schemat połączeń czytnika kart NFC

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 aplikacji. Szczególnie ważne jest to w przypadku aplikacji, które wymagają od użytkownika podania pewnych danych.

6.1.1 Walidacja po stronie klienta

Walidacja po stronie klienta jest realizowana przy pomocy biblioteki `Valibot`. Opiera się ona na tworzeniu obiektów 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 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 rozwiążanie udostępniane przez **Spring Framework**. Opiera się ono na zbiorze adnotacji walidujących, które można umieszczać nad polami klas. Można wykorzystać gotowe adnotacje, lub wykorzystać 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

System został przetestowany aby upewnić się, że działa on poprawnie. Podczas prac zrezygnowano z testów jednostkowych, ponieważ ich wyniki nie byłyby miarodajne. Możliwe byłoby przetestowanie jedynie warstwy serwisowej która nie realizuje pełnej logiki biznesowej - część z niej jest realizowana poprzez repozytoria, które nie są testowane. W związku z tym zdecydowano się na przeprowadzenie jednie testów akceptacyjnych w symulowanym środowisku produkcyjnym. Dzięki nim stwierdzono, że aplikacja działa poprawnie i spełnia założenia.

Rozdział 7

Podsumowanie i wnioski

7.1 Wnioski

Celem pracy było zaprojektowanie i implementacja kompleksowego 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 kompletnego, który spełnia oczekiwania użytkowników.

Wynikiem pracy jest system, który spełnia założenia projektowe i udostępnia użytkownikom wszystkie funkcjonalności w sposób przejrzysty i intuicyjny. System jest 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 procesu rejestracji czasu pracy pracowników, co znacznie ułatwia jego śledzenie. Dzięki ich zastosowaniu możliwe jest również zautomatyzowanie procesu kontroli dostępu do pomieszczeń.

System został zrealizowany w wersji podstawowej, jednak posiada wiele możliwości rozwoju. Udało się zrealizować funkcjonalności dotyczące: zarządzania pracownikami, harmonogramami pracy, zadaniami, czasem pracy, strukturą firmy i rejestracji czasu pracy. Nie udało się zrealizować funkcjonalności dotyczących raportów, wniosków oraz automatycznego śledzenia czasu pracy pracowników. Wszystkie te funkcjonalności mogą 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 ją znaczowo utrudniły i wydłużyły. Pierwszym z nich była konieczność przeprowadzenia kompleksowej analizy wymagań i zrozumienia procesów zachodzących w firmie, co wymagało poświęcenia dużej ilości czasu na dogłębne zaznajomienie się z nimi. Przy projektowaniu systemu nie wzięto pod uwagę stopnia jego złożoności, co poskutkowało koniecznością zrezygnowania z części funkcjonalności lub ograniczeniem ich zakresu.

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 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 ponownej 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 bardo 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 przyjaźniejszy, jawnie oddzielając warstwę prezentacji, logiki i danych. `Expo` okazało się najbardziej uciążliwą technologią, jednakże poza problemem z zmiennymi środowiskowymi nie sprawiło innych problemów. Użycie innej technologii, np. `Flutter` mogłoby przyspieszyć pracę nad aplikacją mobilną, jednakże konieczność nauki nowego frameworka mogłaby znacznie 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 podłączenie większej

ich ilości. Następnie należałoby zająć się brakującymi częściami: raportami, wnioskami i automatycznym śledzeniem czasu pracy pracowników. Raporty powinny być generowane w formie plików PDF, które można by było wydrukować lub przesłać mailem. Do wniosków - oprócz możliwości ich składania i akceptacji - dobrze byłoby dodać możliwość wymieniania się wiadomościami, podobnie do czatu. Automatyczne śledzenie czasu pracy może być zrealizowane algorytmicznie, wywołując odpowiednie funkcje w określonych momentach. Możliwe, że dobrym wyborem było by 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. 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

Wykonany system można zaliczyć do dużych projektów informatycznych, które wymagają zaangażowania wielu osób. Samodzielne stworzenie takiej aplikacji jest możliwe, ale bardzo trudne. Wymaga od programisty bardzo dużej wiedzy, umiejętności i zaangażowania. Duże zespoły programistów są w stanie tworzyć systemy o wiele bardziej rozbudowane i złożone w krótszym 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 przyszłych projektach. System posiada wiele możliwości rozwoju, które mogą znacznie zwiększyć jego użyteczność i funkcjonalność. W przyszłości warto kontynuować prace nad nim, aby w pełni wykorzystać jego potencjał.

Bibliografia

- [1] Axios. *Axios*. URL: <https://axios-http.com/> (term. wiz. 04.10.2024).
- [2] Ministerstwo Cyfryzacji. *WCAG 2.1 w skrócie*. URL: <https://www.gov.pl/web/dostepnosc-cyfrowa/wcag-21-w-skrocie> (term. wiz. 07.10.2024).
- [3] Docker. *Docker*. URL: <https://www.docker.com/> (term. wiz. 25.09.2024).
- [4] Stefan Trzcieliński Edmund Pawłowski. *Zarządzanie przedsiębiorstwem, Funkcje i struktury*. Poznań: Wydawnictwo Politechniki Poznańskiej, 2011. ISBN: 978-83-7143-968-1.
- [5] Expo. *Expo*. URL: <https://expo.dev/> (term. wiz. 25.10.2024).
- [6] Figma. *Figma*. URL: <https://www.figma.com/> (term. wiz. 23.09.2024).
- [7] Raspberry Pi Foundation. *PicoZero*. URL: <https://github.com/RaspberryPiFoundation/picozero> (term. wiz. 08.11.2024).
- [8] Git. *Git - Distributed Version Control System*. URL: <https://git-scm.com/> (term. wiz. 25.09.2024).
- [9] Fabian Hiller. *Valibot*. URL: <https://valibot.dev/> (term. wiz. 04.11.2024).
- [10] Intlify. *Vue I18n*. URL: <https://vue-i18n.intlify.dev/> (term. wiz. 04.10.2024).
- [11] JetBrains. *IntelliJ IDEA*. URL: <https://www.jetbrains.com/idea/> (term. wiz. 23.09.2024).
- [12] jwtk. *Java JWT: JSON Web Token for Java and Android*. URL: <https://github.com/jwtk/jjwt> (term. wiz. 09.09.2024).
- [13] Tailwind Labs. *Heroicons*. URL: <https://heroicons.com/> (term. wiz. 04.10.2024).
- [14] Tailwind Labs. *Tailwind CSS*. URL: <https://tailwindcss.com/> (term. wiz. 26.11.2024).
- [15] Project Lombok. *Project Lombok*. URL: <https://projectlombok.org/> (term. wiz. 25.09.2024).
- [16] Software Mansion. *React Native Gesture Handler*. URL: <https://docs.swmansion.com/react-native-gesture-handler/> (term. wiz. 25.10.2024).
- [17] MicroPython. *MicroPython*. URL: <https://micropython.org/> (term. wiz. 10.11.2024).

- [18] Microsoft. *Visual Studio Code*. URL: <https://code.visualstudio.com/> (term. wiz. 04. 10. 2024).
- [19] MintHCM. *MintHCM*. URL: <https://minthcm.org/> (term. wiz. 30.09.2024).
- [20] Moment.js. *Luxon*. URL: <https://moment.github.io/luxon/#/> (term. wiz. 03.12.2024).
- [21] MySQL. *MySQL*. URL: <https://www.mysql.com/> (term. wiz. 25.09.2024).
- [22] MySQL. *MySQL Connector*. URL: <https://www.mysql.com/products/connector/> (term. wiz. 25.09.2024).
- [23] Nativewind. *Nativewind*. URL: <https://www.nativewind.dev/> (term. wiz. 25.11.2024).
- [24] Oracle. *Oracle Human Capital Management (HCM)*. URL: <https://www.oracle.com/human-capital-management/> (term. wiz. 30.09.2024).
- [25] Daniel Perron. *MicroPython MFRC522*. URL: <https://github.com/danjperron/micropython-mfrc522/tree/master?tab=readme-ov-file> (term. wiz. 11.11.2024).
- [26] Raspberry Pi. *Pico-series Microcontrollers*. URL: <https://www.raspberrypi.com/documentation/microcontrollers/pico-series.html> (term. wiz. 08.11.2024).
- [27] Raspberry Pi. *Pico W Datasheet*. 2023. URL: <https://datasheets.raspberrypi.com/picow/pico-w-datasheet.pdf> (term. wiz. 08.11.2024).
- [28] PrimeTek. *PrimeVue*. URL: <https://primevue.org/> (term. wiz. 04.10.2024).
- [29] Kenneth Reitz. *Requests: HTTP for Humans*. URL: <https://requests.readthedocs.io/en/latest/> (term. wiz. 12.11.2024).
- [30] RevTelTech. *React Native NFC Manager*. URL: <https://github.com/revtel/react-native-nfc-manager> (term. wiz. 25.10.2024).
- [31] SAP. *Human capital management (HCM)*. URL: <https://www.sap.com/products/hcm.html> (term. wiz. 30.09.2024).
- [32] Spring. *Spring Boot*. URL: <https://spring.io/projects/spring-boot> (term. wiz. 09.09.2024).
- [33] Spring. *Spring Framework*. URL: <https://spring.io/> (term. wiz. 25.09.2024).
- [34] Spring. *Spring Security*. URL: <https://spring.io/projects/spring-security> (term. wiz. 09.09.2024).
- [35] SpringDoc. *SpringDoc*. URL: <https://springdoc.org/> (term. wiz. 09.09.2024).
- [36] Struktura organizacyjna firmy – jak może wyglądać? Rodzaje i przykłady. URL: <https://tomhrm.com/struktura-organizacyjna-firmy-rodzaje-przyklady/> (term. wiz. 26.11.2024).

- [37] Thonny. *Thonny, Python IDE for beginners*. URL: <https://thonny.org/> (term. wiz. 03.12.2024).
- [38] 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).
- [39] Vue. *Vue Router*. URL: <https://router.vuejs.org/> (term. wiz. 04.10.2024).
- [40] Vue.js. *Vue.js*. URL: <https://vuejs.org/> (term. wiz. 04.10.2024).
- [41] 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*)

HR Zarządzanie Zasobami Ludzkimi (ang. *Human Resources*)

CRM Zarządzanie Relacjami z Klientami (ang. *Customer Relationship Management*)

SPA Aplikacja jednostronicowa (ang. *Single Page Application*)

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

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

MSC Model-Serwis-Kontroler (ang. *Model-Service-Controller*)

EAS Usługa aplikacji Expo (ang. *Expo Application Services*)

SPI Szeregowy interfejs urządzeń peryferyjnych (ang. *Serial Peripheral Interface*)

Źródła

Lista dodatkowych plików, uzupełniających tekst pracy

Spis rysunków

4.1	Makieta struktury widoku	17
4.2	Widok strony logowania	18
4.3	Panel użytkownika	18
4.4	Widok użytkowników	19
4.5	Okna dialogowe widoku użytkowników	20
4.6	Widok timesheetu	20
4.7	Okno dialogowe dodawania wpisu do timesheetu	21
4.8	Widok timesheetów oczekujących na akceptację	21
4.9	Widok jednostek organizacyjnych	22
4.10	Okna dialogowe widoku jednostek organizacyjnych	23
4.11	Okna dialogowe dodawania użytkowników do jednostki organizacyjnej	23
4.12	Widok harmonogramu	24
4.13	Widok harmonogramu dla managera	24
4.14	Warianty widoku dodawania kart	25
4.15	Widoki dodawania kart	26
4.16	Komunikaty widoku dodawania kart	26
4.17	Widok strony konfiguracyjnej czytnika.	27
5.1	Uproszczony schemat bazy danych	30
5.2	Schemat tabel użytkowników	31
5.3	Schemat tabel jednostek organizacyjnych	32
5.4	Schemat tabel harmonogramów	32
5.5	Schemat tabel kart dostępowych	33
5.6	Diagram czynności logowania	36
5.7	Diagram sekwencji procesu uwierzytelniania użytkownika	37
5.8	Diagram sekwencji odczytu harmonogramu użytkownika	39
5.9	Diagram sekwencji odczytu i przypisania karty dostępowej	40
5.10	Schemat połączeń czytnika kart NFC	42

Spis tabel

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