Dokumentacja projektu
wykonywanego w ramach zajęć
BAZY DANYCH I

System Zarządzania Pracownikami i ich Wynagrodzeniami

Wydział Fizyki i Informatyki Stosowanej
Mikhail Shupliakou
30.10.2025

Spis treści

1 Projekt koncepcji, założenia

1.1 Tematu projektu

System Zarządzania Wynagrodzeniami. Projekt dotyczy stworzenia systemu Zarządzania Wynagrodzeniami, którego celem jest automatyzacja procesu naliczania wynagrodzeń pracowników w organizacji. Aplikacja umożliwia gromadzenie, przetwarzanie i analizę danych dotyczących zatrudnionych osób, ich wynagrodzeń, przepracowanego czasu oraz rozliczeń miesięcznych. System ma usprawnić pracę działu kadr i księgowości poprzez eliminację ręcznych obliczeń, zmniejszenie ryzyka błędów oraz zapewnienie szybkiego dostępu do danych płacowych. Dodatkowo umożliwia pracownikom podgląd własnych informacji dotyczących wynagrodzeń, co zwiększa przejrzystość i komfort korzystania.

1.2 Analiza wymagań użytkownika

System powinien zapewniać funkcjonalności dostosowane do potrzeb trzech typów użytkowników: administratora, księgowego oraz pracownika.
Wymagania użytkowników:

- **Administrator:**
  - zarządzanie użytkownikami systemu oraz nadawanie ról,
  - dodawanie i modyfikacja danych pracowników i działów,
  - konfiguracja stawek wynagrodzeń i zasad rozliczeń.
- **Księgowy:**
  - rejestrowanie czasu pracy pracowników (urlopy, nadgodziny, nieobecności),
  - naliczanie wynagrodzeń za wybrany okres,
  - generowanie list płac oraz raportów rozliczeniowych,
  - obsługa korekt i historii wypłat.
- **Pracownik:**
  - podgląd własnych danych dotyczących wynagrodzenia i przepracowanych godzin,
  - dostęp do historii wypłat,
  - możliwość zgłaszania wniosków urlopowych lub korekt czasu pracy.

System powinien dodatkowo zapewniać ochronę danych oraz kontrolę dostępu poprzez nadawanie ról i autoryzację użytkowników.

1.3 Zaprojektowanie funkcji

Projektowana baza danych realizować będzie następujące podstawowe funkcje:

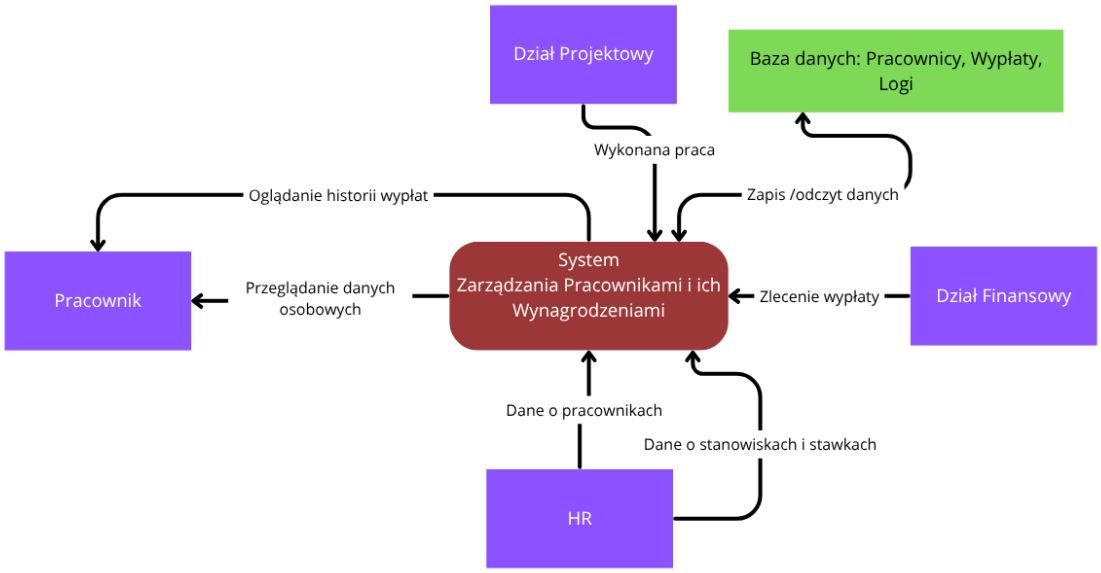
- **Ewidencja czasu pracy** – rejestrowanie obecności, nadgodzin, urlopów, nieobecności oraz automatyczne uwzględnianie ich w naliczaniu płac.
- **Generowanie list płac** – automatyczne obliczanie wynagrodzenia na podstawie danych pracownika oraz przepracowanego czasu.
- **Raportowanie** – tworzenie raportów płacowych, miesięcznych zestawień kosztów pracowniczych oraz historii wypłat.
- **Statystyki** – możliwość zobaczenia średniej liczby godzin pracy za jakis okres.

2 Projekt diagramów

2.1 Budowa i analiza diagramu przepływu danych

2.1.1 Diagram przepływu danych (DFD – poziom 0)

Diagram przepływu danych (DFD) na rysunku 1 przedstawia sposób, w jaki informacje przemieszczają się pomiędzy zewnętrznymi aktorami a systemem Zarządzania Pracownikami i ich Wynagrodzeniami. Jest to diagram kontekstowy (poziomu 0), który ukazuje ogólny przepływ danych bez wchodzenia w szczegóły implementacyjne. Na tym etapie system jest traktowany jako jedna, spójna jednostka przetwarzająca dane, a celem diagramu jest pokazanie głównych źródeł, odbiorców oraz typów przesyłanych informacji.



Rysunek 1: DFD Diagram poziomu 0(konceptualny) systemu

Opis elementów diagramu

- **System Zarządzania Pracownikami i ich Wynagrodzeniami**
  Centralny proces odpowiedzialny za gromadzenie, przetwarzanie i udostępnianie informacji o pracownikach, stanowiskach, wynagrodzeniach, historii wypłat oraz urlopach. System komunikuje się z czterema zewnętrznymi jednostkami oraz korzysta z jednej wspólnej bazy danych.
- **HR (Dział Kadr)**
  Wprowadza i aktualizuje dane o pracownikach, stanowiskach oraz stawkach wynagrodzeń. Otrzymuje dostęp do informacji o aktualnym stanie zatrudnienia.
- **Dział Finansowy**
  Przekazuje do systemu zlecenia wypłaty wynagrodzeń i pobiera raporty dotyczące historii wypłat, statusów i kosztów wynagrodzeń.
- **Dział Projektowy**
  Przekazuje dane o wykonanej pracy (np. liczbie godzin, zrealizowanych zadaniach), które wpływają na naliczanie wynagrodzeń projektowych.
- **Pracownik**
  Ma dostęp do swoich danych osobowych oraz historii wypłat zapisanych w systemie. Centralne repozytorium przechowujące dane o pracownikach, stanowiskach, wypłatach, urlopach i logach operacji. System zapisuje w niej zmiany oraz odczytuje dane niezbędne do raportowania i wyświetlania informacji użytkownikom.
- **Baza danych systemu**
  Centralne repozytorium przechowujące dane o pracownikach, stanowiskach, wypłatach, urlopach i logach operacji. System zapisuje w niej zmiany oraz odczytuje dane niezbędne do raportowania i wyświetlania informacji użytkownikom.

2.2 Zdefiniowanie encji oraz ich atrybutów

Na podstawie analizy wymagań oraz diagramu bazy danych, zdefiniowano następujące encje (tabele) oraz przypisano im odpowiednie atrybuty.

2.2.1 Zarządzanie strukturą i personelem

- **Pracownik** (`pracownik`) – centralna encja systemu reprezentująca zatrudnioną osobę.
  - `id_pracownik` (PK) – unikalny identyfikator pracownika.
  - `imie`, `nazwisko` – dane osobowe.
  - `email`, `telefon` – dane kontaktowe.
  - `id_stanowisko` (FK) – powiązanie ze stanowiskiem.

  - `id_rola` (FK) – rola w systemie (np. administrator, pracownik).
  - `id_dzial` (FK) – przypisanie do działu w firmie.
  - `wynagrodzenie_pln_g` – stawka godzinowa pracownika.
  - `haslo_hash` – zahaszowane hasło do logowania.
  - `data_zatrudnienia`, `data_zwolnienia` – okres zatrudnienia.
  - `aktywny` – flaga określająca, czy pracownik jest aktualnie zatrudniony.
  - `konto_bankowe` – numer konta do przelewów.
- **Dział (`dzial`)** – słownik działów w firmie.
  - `id_dzial` (PK) – identyfikator działu.
  - `nazwa` – nazwa działu.
  - `opis` – dodatkowy opis działu.
- **Stanowisko (`stanowisko`)** – słownik dostępnych stanowisk.
  - `id_stanowisko` (PK) – identyfikator stanowiska.
  - `nazwa` – nazwa stanowiska.
  - `opis` – opis obowiązków.
- **Rola (`rola`)** – słownik ról systemowych (uprawnień).
  - `id_rola` (PK) – identyfikator roli.
  - `nazwa` – nazwa roli (np. Admin, HR).

2.2.2 Zarządzanie czasem pracy i projektami

- **Rejestracja godzin pracy (`rejestracja_godzin_pracy`)** – ewidencja czasu pracy.
  - `id_rejestracji` (PK) – identyfikator wpisu.
  - `id_pracownik` (FK) – pracownik, którego dotyczy wpis.
  - `data` – dzień pracy.
  - `godzina_rozpoczecia`, `godzina_zakonczenia` – ramy czasowe pracy.
  - `id_typ_pracy` (FK) – rodzaj pracy (np. stacjonarna, zdalna).
  - `id_projekt` (FK) – projekt, nad którym pracowano.
  - `komentarz`, `zatwierdzenie` – dodatkowe informacje i status akceptacji.
- **Projekt (`projekt`)** – realizowane projekty.
  - `id_projekt` (PK) – identyfikator projektu.
  - `nazwa`, `opis` – szczegóły projektu.
  - `data_rozpoczecia`, `data_zakonczenia` – ramy czasowe projektu.

- **Pracownik w projekcie (`pracownik_projekt`)** – tabela łącząca (wiele do wielu), przypisująca pracowników do projektów.
  - `id_pracownik` (FK), `id_projekt` (FK) – klucze obce.
  - `rola_w_projekcie` – funkcja pełniona w danym projekcie.
  - `data_przypisania` – data dołączenia do zespołu.
- **Urlop (`urlop`)** – ewidencja nieobecności.
  - `id_urlop` (PK) – identyfikator wniosku.
  - `id_pracownik` (FK) – wnioskujący pracownik.
  - `poczatek`, `koniec` – zakres dat urlopu.
  - `id_typ_urlop` (FK), `id_status_urlopu` (FK) – rodzaj i status wniosku.
  - `data_zatwierdzenia`, `id_osoba_zatwierdzajaca` – dane autoryzacji.
- **Słowniki urlopowe** – encje pomocnicze:
  - `typ_urlop`: nazwa i opis rodzaju urlopu.
  - `status_urlopu`: nazwa i opis statusu (np. Oczekujący, Zatwierdzony).

2.2.3 Finanse i historia

- **Historia wypłat (`historia_wyplat`)** – rejestr dokonanych przelewów.
  - `id_wyplata` (PK) – identyfikator transakcji.
  - `id_pracownik` (FK) – odbiorca wypłaty.
  - `data`, `wyplata` – data operacji i kwota.
  - `id_status_wyplaty` (FK), `id_typ_wyplaty` (FK) – status i rodzaj przelewu.
- **Historia zmian wynagrodzeń (`historia_zmian_wynagrodzen`)** – archiwum zmian stawek.
  - `id_zmiany_wynagrodzenia` (PK).
  - `stare_wynagr`, `nowe_wynagr` – wartości przed i po zmianie.
  - `data`, `opis` – data zmiany i powód.
- **Słowniki finansowe** – encje pomocnicze:
  - `typ_wyplaty`: rodzaj świadczenia (np. podstawowe, premia).
  - `status_wyplaty`: status przelewu (np. zlecony, wysłany).

2.2.4 Logi systemowe

- **Logi operacji (`logi_operacji`)** – audyt działań w systemie.
  - `id_log` (PK) – identyfikator zdarzenia.
  - `id_pracownik` (FK) – sprawca zdarzenia.
  - `nazwa_operacji`, `tabela` – co i gdzie zostało zmienione.
  - `stara_wartosc`, `nowa_wartosc` – szczegóły zmiany.
  - `data_operacji`, `adres_ip` – metadane techniczne.

2.3 Zaprojektowanie relacji pomiędzy encjami

System oparty jest na relacyjnym modelu danych, w którym spójność informacji zapewniona jest poprzez sieć powiązań między tabelami. Poniżej opisano kluczowe relacje zidentyfikowane w projekcie, z podziałem na ich charakter logiczny.

2.3.1 Relacje strukturalne (Słowniki i Pracownik)

Centralną encją systemu jest tabela `pracownik`, która jest stroną liczną (N) w relacjach ze słownikami systemowymi. Oznacza to, że jeden pracownik może mieć przypisane tylko jedno stanowisko, rolę czy dział, ale dany dział może zatrudniać wielu pracowników.

- **Pracownik – Dział** ($N:1$):
  Relacja realizowana przez klucz obcy `pracownik.id_dzial` odnoszący się do `dzial.id_dzial`. Pozwala na grupowanie pracowników w jednostki organizacyjne.
- **Pracownik – Stanowisko** ($N:1$):
  Relacja poprzez `pracownik.id_stanowisko` $\rightarrow$ `stanowisko.id_stanowisko`. Definiuje hierarchię i zakres obowiązków.
- **Pracownik – Rola** ($N:1$):
  Relacja poprzez `pracownik.id_rola` $\rightarrow$ `rola.id_rola`. Służy do zarządzania uprawnieniami w aplikacji (RBAC).

2.3.2 Relacje operacyjne (Czas pracy i Projekty)

Zarządzanie czasem pracy wymaga relacji, które pozwalają na wielokrotne rejestrowanie zdarzeń dla jednego pracownika.

- **Pracownik – Projekty** (Relacja wiele-do-wielu $N:M$):
  Bezpośrednia relacja między pracownikiem a projektem nie jest możliwa w modelu relacyjnym, dlatego zastosowano tabelę łączącą `pracownik_projekt`.
  - `pracownik_projekt.id_pracownik` $\rightarrow$ `pracownik.id_pracownik`

  - `pracownik_projekt.id_projekt` → `projekt.id_projekt`

  Umożliwia to przypisanie jednego pracownika do wielu projektów jednocześnie oraz pracę wielu osób nad jednym projektem.

- **Rejestracja godzin pracy**:
  Tabela `rejestracja_godzin_pracy` posiada relacje typu $N:1$ z tabelami:
  - `pracownik` (kto pracował),
  - `projekt` (nad czym pracował),
  - `typ_pracy` (w jakim trybie, np. zdalnie).
- **Urlopy**:
  Tabela `urlop` jest powiązana z pracownikiem (`id_pracownik`) oraz słownikami określającymi typ nieobecności (`typ_urlop`) i jej status akceptacji (`status_urlopu`).

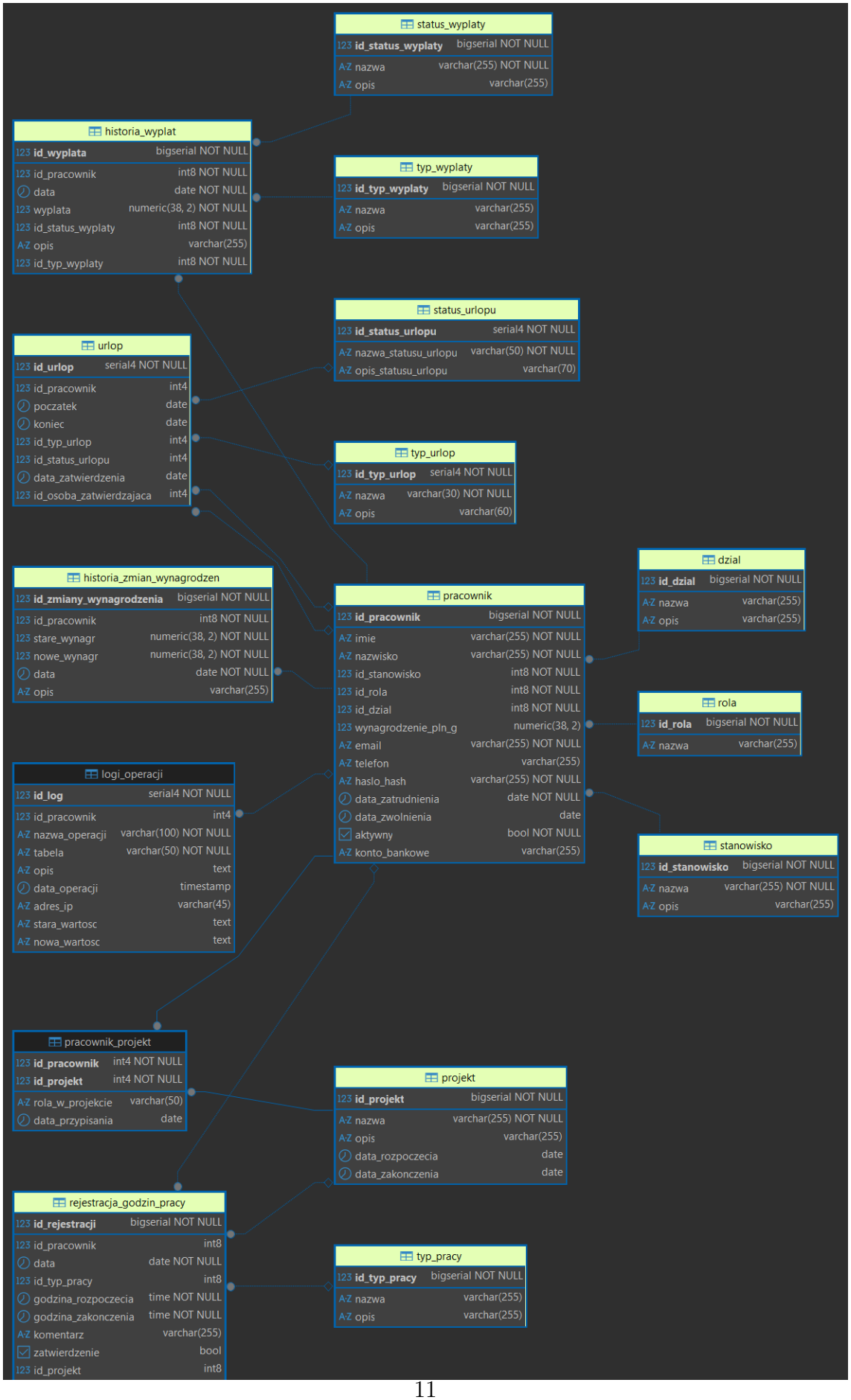
2.3.3 Relacje finansowe i historyczne

Ze względu na wymogi księgowe, system przechowuje historię operacji, co wymusza relacje jeden-do-wielu ($1:N$) od pracownika do tabel historycznych.

- **Historia wypłat** ($1:N$):
  Jeden pracownik posiada wiele rekordów w tabeli `historia_wyplat`. Każdy wpis jest dodatkowo opisany relacjami do słowników `status_wyplaty` i `typ_wyplaty`.
- **Historia zmian wynagrodzeń** ($1:N$):
  Tabela `historia_zmian_wynagrodzen` przechowuje audyt zmian stawek płacowych, łącząc się z tabelą pracownika kluczem `id_pracownik`.

2.3.4 Integralność danych

Wszystkie zdefiniowane relacje oparte są na kluczach obcych z wymuszeniem integralności referencyjnej. Usunięcie kluczowych rekordów słownikowych (np. typu umowy) jest zablokowane, jeśli istnieją powiązane z nimi rekordy operacyjne, co zabezpiecza system przed niespójnością danych.



Rysunek 2: ERD diagram bazy danych

3 Projekt logiczny

3.1 Projektowanie tabel, kluczy, indeksów

W oparciu o zdefiniowany model logiczny oraz diagram ERD, zaprojektowano fizyczną strukturę bazy danych. System został zaimplementowany przy użyciu relacyjnej bazy danych PostgreSQL, co zapewnia stabilność, zgodność ze standardami SQL oraz obsługę zaawansowanych typów danych.

3.2 Specyfikacja techniczna struktury

Struktura bazy danych opiera się na następujących założeniach technicznych:

- **Nazewnictwo:** Zastosowano konwencję *snake_ case* dla nazw tabel i kolumn, co jest standardem w środowisku PostgreSQL.
- **Klucze główne (Primary Keys):** Każda tabela posiada klucz główny (zazwyczaj o nazwie `id_nazwatabeli`), będący liczbą całkowitą, generowaną automatycznie (typy `SERIAL` lub `BIGSERIAL`). Gwarantuje to unikalność rekordów.
- **Klucze obce (Foreign Keys):** Relacje między tabelami są realizowane poprzez klucze obce, które wymuszają integralność referencyjną.
- **Typy danych:**
  - `VARCHAR(n)` – dla danych tekstowych o zmiennej długości (imiona, nazwy, opisy, e-maile).
  - `NUMERIC(38, 2)` – dla wartości finansowych (wynagrodzenia), aby uniknąć błędów zaokrągleń typowych dla typów zmiennoprzecinkowych.
  - `DATE` / `TIMESTAMP` – dla przechowywania dat i znaczników czasowych.
  - `BOOLEAN` – dla flag logicznych (np. czy pracownik jest aktywny).

3.3 Indeksy

W celu optymalizacji wydajności zapytań, baza danych wykorzystuje mechanizm indeksowania:

1. **Indeksy klastrowe (Primary Key Indexes):** Tworzone automatycznie dla każdego klucza głównego.
2. **Indeksy na kluczach obcych:** Zalecane dla kolumn wykorzystywanych w złączeniach (`JOIN`), takich jak `id_pracownik` w tabelach historycznych czy rejestracji czasu pracy.

3.4 Implementacja SQL (DDL)

Poniżej przedstawiono kod SQL tworzący strukturę tabel zgodnie z projektowanym diagramem. Skrypt uwzględnia kolejność tworzenia tabel (najpierw słowniki, potem tabele zależne) w celu uniknięcia błędów kluczy obcych.

```sql
-- 1. Tabela: Dzial
CREATE TABLE system_zarzadzania_wynagrodzeniami.dzial (
  id_dzial bigserial NOT NULL,
  nazwa varchar(255) NULL,
  opis varchar(255) NULL,
  CONSTRAINT dzial_pkey PRIMARY KEY (id_dzial)
);

-- 2. Tabela: Stanowisko
CREATE TABLE system_zarzadzania_wynagrodzeniami.stanowisko (
  id_stanowisko bigserial NOT NULL,
  nazwa varchar(255) NOT NULL,
  opis varchar(255) NULL,
  CONSTRAINT stanowisko_pkey PRIMARY KEY (id_stanowisko)
);

-- 3. Tabela: Rola
CREATE TABLE system_zarzadzania_wynagrodzeniami.rola (
  id_rola bigserial NOT NULL,
  nazwa varchar(255) NULL,
  CONSTRAINT rola_pkey PRIMARY KEY (id_rola)
);

-- 4. Tabela: Typ Pracy
CREATE TABLE system_zarzadzania_wynagrodzeniami.typ_pracy (
  id_typ_pracy bigserial NOT NULL,
  nazwa varchar(255) NULL,
  opis varchar(255) NULL,
  CONSTRAINT typ_pracy_pkey PRIMARY KEY (id_typ_pracy)
);

-- 5. Tabela: Typ Urlopu
CREATE TABLE system_zarzadzania_wynagrodzeniami.typ_urlop (
  id_typ_urlop serial4 NOT NULL,
  nazwa varchar(30) NOT NULL,
  opis varchar(60) NULL,
  CONSTRAINT typ_urlop_pkey PRIMARY KEY (id_typ_urlop)
);

-- 6. Tabela: Status Urlopu
CREATE TABLE system_zarzadzania_wynagrodzeniami.status_urlopu (
  id_status_urlopu serial4 NOT NULL,
  nazwa_statusu_urlopu varchar(50) NOT NULL,
  opis_statusu_urlopu varchar(70) NULL,
  CONSTRAINT status_urlopu_pkey PRIMARY KEY (id_status_urlopu)
);

-- 7. Tabela: Typ Wyplaty
CREATE TABLE system_zarzadzania_wynagrodzeniami.typ_wyplaty (
  id_typ_wyplaty bigserial NOT NULL,
  nazwa varchar(255) NULL,
  opis varchar(255) NULL,
  CONSTRAINT typ_wyplaty_pkey PRIMARY KEY (id_typ_wyplaty)
```

```
);

-- 8. Tabela: Status Wyplaty
CREATE TABLE system_zarzadzania_wynagrodzeniami.status_wyplaty (
  id_status_wyplaty bigserial NOT NULL,
  nazwa varchar(255) NOT NULL,
  opis varchar(255) NULL,
  CONSTRAINT status_wyplaty_pkey PRIMARY KEY (id_status_wyplaty)
);

-- 9. Tabela: Projekt
CREATE TABLE system_zarzadzania_wynagrodzeniami.projekt (
  id_projekt bigserial NOT NULL,
  nazwa varchar(255) NOT NULL,
  opis varchar(255) NULL,
  data_rozpoczecia date NULL,
  data_zakonczenia date NULL,
  CONSTRAINT projekt_pkey PRIMARY KEY (id_projekt)
);

-- 10. Tabela: Pracownik (Główna tabela)
CREATE TABLE system_zarzadzania_wynagrodzeniami.pracownik (
  id_pracownik bigserial NOT NULL,
  imie varchar(255) NOT NULL,
  nazwisko varchar(255) NOT NULL,
  id_stanowisko int8 NOT NULL,
  id_rola int8 NOT NULL,
  id_dzial int8 NOT NULL,
  wynagrodzenie_pln_g numeric(38, 2) NULL,
  email varchar(255) NOT NULL,
  telefon varchar(255) NULL,
  haslo_hash varchar(255) NOT NULL,
  data_zatrudnienia date NOT NULL,
  data_zwolnienia date NULL,
  aktywny bool NOT NULL,
  konto_bankowe varchar(255) NULL,
  CONSTRAINT pracownik_pkey PRIMARY KEY (id_pracownik)
);


-- system_zarzadzania_wynagrodzeniami.pracownik foreign keys

ALTER TABLE system_zarzadzania_wynagrodzeniami.pracownik ADD CONSTRAINT
↪  fk_id_dzial FOREIGN KEY (id_dzial) REFERENCES
↪  system_zarzadzania_wynagrodzeniami.dzial(id_dzial) ON DELETE RESTRICT;
ALTER TABLE system_zarzadzania_wynagrodzeniami.pracownik ADD CONSTRAINT
↪  fk_id_rola FOREIGN KEY (id_rola) REFERENCES
↪  system_zarzadzania_wynagrodzeniami.rola(id_rola) ON DELETE RESTRICT;
ALTER TABLE system_zarzadzania_wynagrodzeniami.pracownik ADD CONSTRAINT
↪  fk_id_stanowisko FOREIGN KEY (id_stanowisko) REFERENCES
↪  system_zarzadzania_wynagrodzeniami.stanowisko(id_stanowisko) ON DELETE
↪  RESTRICT;

-- 11. Tabela: Rejestracja Godzin Pracy
CREATE TABLE system_zarzadzania_wynagrodzeniami.rejestracja_godzin_pracy (
  id_rejestracji bigserial NOT NULL,
  id_pracownik int8 NULL,
  "data" date NOT NULL,
  id_typ_pracy int8 NULL,
  godzina_rozpoczecia time NOT NULL,
```

```
  godzina_zakonczenia time NOT NULL,
  komentarz varchar(255) NULL,
  zatwierdzenie bool NULL,
  id_projekt int8 NULL,
  CONSTRAINT rejestracja_godzin_pracy_pkey PRIMARY KEY (id_rejestracji)
);


-- system_zarzadzania_wynagrodzeniami.rejestracja_godzin_pracy foreign keys

ALTER TABLE system_zarzadzania_wynagrodzeniami.rejestracja_godzin_pracy ADD
↪  CONSTRAINT dk_id_typ_pracy FOREIGN KEY (id_typ_pracy) REFERENCES
↪  system_zarzadzania_wynagrodzeniami.typ_pracy(id_typ_pracy);
ALTER TABLE system_zarzadzania_wynagrodzeniami.rejestracja_godzin_pracy ADD
↪  CONSTRAINT fk_rejestracja_projekt FOREIGN KEY (id_projekt) REFERENCES
↪  system_zarzadzania_wynagrodzeniami.projekt(id_projekt);
ALTER TABLE system_zarzadzania_wynagrodzeniami.rejestracja_godzin_pracy ADD
↪  CONSTRAINT rejestracja_godzin_pracy_id_pracownik_fkey FOREIGN KEY
↪  (id_pracownik) REFERENCES
↪  system_zarzadzania_wynagrodzeniami.pracownik(id_pracownik) ON DELETE CASCADE;

-- 12. Tabela: Pracownik_Projekt (Tabela łącząca)
CREATE TABLE system_zarzadzania_wynagrodzeniami.pracownik_projekt (
  id_pracownik int4 NOT NULL,
  id_projekt int4 NOT NULL,
  rola_w_projekcie varchar(50) NULL,
  data_przypisania date DEFAULT CURRENT_DATE NULL,
  CONSTRAINT pracownik_projekt_pkey PRIMARY KEY (id_pracownik, id_projekt)
);


-- system_zarzadzania_wynagrodzeniami.pracownik_projekt foreign keys

ALTER TABLE system_zarzadzania_wynagrodzeniami.pracownik_projekt ADD CONSTRAINT
↪  pracownik_projekt_id_pracownik_fkey FOREIGN KEY (id_pracownik) REFERENCES
↪  system_zarzadzania_wynagrodzeniami.pracownik(id_pracownik) ON DELETE CASCADE;
ALTER TABLE system_zarzadzania_wynagrodzeniami.pracownik_projekt ADD CONSTRAINT
↪  pracownik_projekt_id_projekt_fkey FOREIGN KEY (id_projekt) REFERENCES
↪  system_zarzadzania_wynagrodzeniami.projekt(id_projekt) ON DELETE CASCADE;

-- 13. Tabela: Urlop
CREATE TABLE system_zarzadzania_wynagrodzeniami.urlop (
  id_urlop serial4 NOT NULL,
  id_pracownik int4 NULL,
  poczatek date NULL,
  koniec date NULL,
  id_typ_urlop int4 NULL,
  id_status_urlopu int4 NULL,
  data_zatwierdzenia date NULL,
  id_osoba_zatwierdzajaca int4 NULL,
  CONSTRAINT urlop_pkey PRIMARY KEY (id_urlop)
);


-- system_zarzadzania_wynagrodzeniami.urlop foreign keys

ALTER TABLE system_zarzadzania_wynagrodzeniami.urlop ADD CONSTRAINT
↪  fk_id_osoba_zatwierdzajaca FOREIGN KEY (id_osoba_zatwierdzajaca) REFERENCES
↪  system_zarzadzania_wynagrodzeniami.pracownik(id_pracownik) ON DELETE SET
↪  NULL;
```

```
ALTER TABLE system_zarzadzania_wynagrodzeniami.urlop ADD CONSTRAINT
↪  fk_id_status_urlopu FOREIGN KEY (id_status_urlopu) REFERENCES
↪  system_zarzadzania_wynagrodzeniami.status_urlopu(id_status_urlopu) ON DELETE
↪  SET NULL;
ALTER TABLE system_zarzadzania_wynagrodzeniami.urlop ADD CONSTRAINT fk_id_urlop
↪  FOREIGN KEY (id_typ_urlop) REFERENCES
↪  system_zarzadzania_wynagrodzeniami.typ_urlop(id_typ_urlop);
ALTER TABLE system_zarzadzania_wynagrodzeniami.urlop ADD CONSTRAINT
↪  urlop_id_pracownik_fkey FOREIGN KEY (id_pracownik) REFERENCES
↪  system_zarzadzania_wynagrodzeniami.pracownik(id_pracownik);

-- 14. Tabela: Historia Wyplat
CREATE TABLE system_zarzadzania_wynagrodzeniami.historia_wyplat (
  id_wyplata bigserial NOT NULL,
  id_pracownik int8 NOT NULL,
  "data" date NOT NULL,
  wyplata numeric(38, 2) NOT NULL,
  id_status_wyplaty int8 NOT NULL,
  opis varchar(255) NULL,
  id_typ_wyplaty int8 NOT NULL,
  CONSTRAINT historia_wyplat_pkey PRIMARY KEY (id_wyplata)
);


-- system_zarzadzania_wynagrodzeniami.historia_wyplat foreign keys

ALTER TABLE system_zarzadzania_wynagrodzeniami.historia_wyplat ADD CONSTRAINT
↪  fk_id_pracownik FOREIGN KEY (id_pracownik) REFERENCES
↪  system_zarzadzania_wynagrodzeniami.pracownik(id_pracownik) ON DELETE CASCADE;
ALTER TABLE system_zarzadzania_wynagrodzeniami.historia_wyplat ADD CONSTRAINT
↪  fk_id_status_wyplaty FOREIGN KEY (id_status_wyplaty) REFERENCES
↪  system_zarzadzania_wynagrodzeniami.status_wyplaty(id_status_wyplaty);
ALTER TABLE system_zarzadzania_wynagrodzeniami.historia_wyplat ADD CONSTRAINT
↪  fk_id_typ_wyplaty FOREIGN KEY (id_typ_wyplaty) REFERENCES
↪  system_zarzadzania_wynagrodzeniami.typ_wyplaty(id_typ_wyplaty);

-- 15. Tabela: Historia Zmian Wynagrodzen
CREATE TABLE system_zarzadzania_wynagrodzeniami.historia_zmian_wynagrodzen (
  id_zmiany_wynagrodzenia bigserial NOT NULL,
  id_pracownik int8 NOT NULL,
  stare_wynagr numeric(38, 2) NOT NULL,
  nowe_wynagr numeric(38, 2) NOT NULL,
  "data" date NOT NULL,
  opis varchar(255) NULL,
  CONSTRAINT historia_zmian_wynagrodzen_pkey PRIMARY KEY
  ↪  (id_zmiany_wynagrodzenia)
);


-- system_zarzadzania_wynagrodzeniami.historia_zmian_wynagrodzen foreign keys

ALTER TABLE system_zarzadzania_wynagrodzeniami.historia_zmian_wynagrodzen ADD
↪  CONSTRAINT fk_id_pracownik FOREIGN KEY (id_pracownik) REFERENCES
↪  system_zarzadzania_wynagrodzeniami.pracownik(id_pracownik) ON DELETE CASCADE;

-- 16. Tabela: Logi Operacji
CREATE TABLE system_zarzadzania_wynagrodzeniami.logi_operacji (
  id_log serial4 NOT NULL,
  id_pracownik int4 NULL,
  nazwa_operacji varchar(100) NOT NULL,
```

```
  tabela varchar(50) NOT NULL,
  opis text NULL,
  data_operacji timestamp DEFAULT CURRENT_TIMESTAMP NULL,
  adres_ip varchar(45) NULL,
  stara_wartosc text NULL,
  nowa_wartosc text NULL,
  CONSTRAINT logi_operacji_pkey PRIMARY KEY (id_log)
);


-- system_zarzadzania_wynagrodzeniami.logi_operacji foreign keys

ALTER TABLE system_zarzadzania_wynagrodzeniami.logi_operacji ADD CONSTRAINT
↪  logi_operacji_id_pracownik_fkey FOREIGN KEY (id_pracownik) REFERENCES
↪  system_zarzadzania_wynagrodzeniami.pracownik(id_pracownik) ON DELETE SET
↪  NULL;
```

3.5 Słowniki danych

Słownik danych stanowi szczegółową specyfikację atrybutów wykorzystywanych w systemie. Definiuje on nazwy pól, ich fizyczne typy danych, wymagalność oraz ograniczenia biznesowe i integralnościowe.

3.6 Definicja dziedzin (Typy danych)

W projekcie przyjęto następujące standardy typów danych (dziedziny), które zapewniają spójność przechowywanych informacji:

- **Identyfikator (PK/FK):** `BIGSERIAL / BIGINT` – liczby całkowite służące do unikalnej identyfikacji rekordów i tworzenia relacji.
- **Tekst krótki:** `VARCHAR(255)` – nazwy własne, imiona, nazwiska, e-maile.
- **Tekst opisowy:** `TEXT` lub `VARCHAR(>255)` – komentarze, opisy, logi zmian.
- **Waluta:** `NUMERIC(38, 2)` – wartości pieniężne z dokładnością do 2 miejsc po przecinku (grosze), zabezpieczone przed błędami zaokrągleń.
- **Data i Czas:** `DATE` (data kalendarzowa), `TIME` (godzina), `TIMESTAMP` (znacznik czasu dla logów).
- **Flaga:** `BOOLEAN` – wartości logiczne prawda/fałsz (np. czy zatwierdzono).

3.7 Szczegółowy opis atrybutów

Poniższe tabele prezentują słownik danych dla kluczowych modułów systemu.

3.7.1 Moduł: Kadry (Tabela `pracownik`)

Tabela 1: Słownik danych dla tabeli Pracownik

Nazwa atrybutu	Typ danych	Wymagany	Opis i Ograniczenia
id_pracownik	BIGSERIAL	TAK	Klucz główny (PK). Automatycznie inkrementowany.
imie	VARCHAR(255)	TAK	Imię pracownika.
nazwisko	VARCHAR(255)	TAK	Nazwisko pracownika.
email	VARCHAR(255)	TAK	Adres e-mail. Musi zawierać znak '@'. Unikalny w systemie.
wynagrodzenie_pln_g	NUMERIC(38, 2)	NIE	Stawka godzinowa. Ograniczenie: wartość ≥ 0.
data_zatrudnienia	DATE	TAK	Data rozpoczęcia pracy.
aktywny	BOOLEAN	TAK	Flaga statusu. Default: `TRUE`.
haslo_hash	VARCHAR(255)	TAK	Skrót hasła (nie przechowywane jawnym tekstem).
id_stanowisko	BIGINT	TAK	Klucz obcy (FK) do tabeli `stanowisko`.

3.7.2 Moduł: Czas Pracy (Tabela `rejestracja_godzin_pracy`)

Tabela 2: Słownik danych dla tabeli Rejestracja Godzin Pracy

Nazwa atrybutu	Typ danych	Wymagany	Opis i Ograniczenia
id_rejestracji	BIGSERIAL	TAK	Klucz główny (PK).
data	DATE	TAK	Dzień wykonywania pracy. Data nie może być z przyszłości.
godzina_rozpoczecia	TIME	TAK	Czas startu.
godzina_zakonczenia	TIME	TAK	Czas końca. Ograniczenie: `zakonczenie > rozpoczecie`.
zatwierdzenie	BOOLEAN	NIE	Czy wpis został zaakceptowany przez przełożonego.
id_projekt	BIGINT	NIE	Klucz obcy (FK). Projekt, którego dotyczy wpis.

3.7.3 Moduł: Finanse (Tabela `historia_wyplat`)

Tabela 3: Słownik danych dla tabeli Historia Wypłat

Nazwa atrybutu	Typ danych	Wymagany	Opis i Ograniczenia
id_wyplata	BIGSERIAL	TAK	Klucz główny (PK).
data	DATE	TAK	Data wykonania przelewu.
wyplata	NUMERIC(38, 2)	TAK	Kwota netto przelewu. Ograniczenie: wartość > 0.
id_status_wyplaty	BIGINT	TAK	Klucz obcy (FK). Określa stan przelewu (np. Zlecony, Wysłany).
id_pracownik	BIGINT	TAK	Klucz obcy (FK). Odbiorca wynagrodzenia.

3.8 Ograniczenia integralnościowe

W systemie zdefiniowano następujące globalne reguły integralności:

- **Referencyjność:** Nie można usunąć rekordu słownikowego (np. stanowiska), jeśli jest on przypisany do co najmniej jednego pracownika (blokada `ON DELETE RESTRICT` lub kaskada w zależności od konfiguracji).
- **Logika dat:** Data zatrudnienia nie może być późniejsza niż data zwolnienia.
- **Unikalność:** Adresy e-mail pracowników nie mogą się powtarzać w obrębie systemu.

3.9 Analiza zależności funkcyjnych i normalizacja tabel (dekompozycja do 3NF ewentualnie BCNF)

Celem normalizacji jest organizacja struktury danych w sposób, który minimalizuje redundancję oraz eliminuje anomalie wstawiania, usuwania i aktualizacji. Poniżej przedstawiono analizę projektu pod kątem spełniania warunków poszczególnych postaci normalnych.

3.9.1 Pierwsza Postać Normalna (1NF)

Definicja: Tabela jest w 1NF, jeśli:

- Każda kolumna zawiera wartości atomowe (niepodzielne).
- Każdy wiersz jest unikalny (posiada klucz główny).
- W kolumnach nie występują grupy powtarzalne.

Analiza na przykładzie tabeli `pracownik`: W zaprojektowanym systemie atrybuty takie jak `imie`, `nazwisko`, `email` czy `telefon` przechowują pojedyncze wartości. Nie zastosowano pól przechowujących listy (np. "telefon1, telefon2"w jednej komórce). Każda tabela w schemacie posiada zdefiniowany klucz główny (*Primary Key*).
Wniosek: Wszystkie tabele w projekcie spełniają warunki 1NF.

3.9.2 Druga Postać Normalna (2NF)

Definicja: Tabela jest w 2NF, jeśli:

- Spełnia warunki 1NF.
- Wszystkie atrybuty niekluczowe są w pełni funkcyjnie zależne od całego klucza głównego (brak częściowych zależności w przypadku kluczy złożonych).

Analiza na przykładzie tabeli łączącej `pracownik_projekt`: Tabela ta posiada złożony klucz główny: (`id_pracownik`, `id_projekt`). Przeanalizujmy atrybuty niekluczowe:

- `rola_w_projekcie`: Zależy od konkretnego pracownika w konkretnym projekcie (nie od samego pracownika, ani nie od samego projektu).
- `data_przypisania`: Określa moment przypisania konkretnej pary (pracownik, projekt).

Zależność funkcyjna ma postać:

$$\{id\_pracownik, id\_projekt\} \rightarrow \{rola\_w\_projekcie, data\_przypisania\}$$

Żaden atrybut nie zależy tylko od części klucza.
Wniosek: Tabele posiadające klucze złożone spełniają warunki 2NF. Pozostałe tabele posiadają klucze proste (jednokolumnowe), więc automatycznie spełniają 2NF.

3.9.3 Trzecia Postać Normalna (3NF)

Definicja: Tabela jest w 3NF, jeśli:

- Spełnia warunki 2NF.
- Żaden atrybut niekluczowy nie jest zależny tranzytywnie od klucza głównego (nie zależy od innego atrybutu niekluczowego).

Proces dekompozycji (Normalizacja): W początkowej fazie projektowania, tabela `pracownik` mogłaby zawierać redundantne dane, np.:

$$id\_pracownik \rightarrow \{nazwisko, nazwa\_stanowiska, nazwa\_dzialu\}$$

Występowała tu zależność przechodnia:

$$id\_pracownik \rightarrow id\_stanowisko \rightarrow nazwa\_stanowiska$$

Aby spełnić 3NF, dokonano dekompozycji tabeli `pracownik` poprzez wydzielenie słowników.

Stan po normalizacji: Utworzono osobne tabele słownikowe: `stanowisko`, `dzial`, `rola`. W tabeli `pracownik` pozostawiono jedynie klucze obce:

- `id_stanowisko` zamiast `nazwa_stanowiska`
- `id_dzial` zamiast `nazwa_dzialu`

Dzięki temu zmiana nazwy stanowiska "Programista"na "Developer"odbywa się tylko w jednym miejscu (tabela `stanowisko`), eliminując anomalię aktualizacji.

Wniosek: Dzięki zastosowaniu tabel słownikowych i kluczy obcych, usunięto zależności przechodnie. Baza danych znajduje się w Trzeciej Postaci Normalnej (3NF).

3.9.4 Postać Normalna Boyce'a-Codda (BCNF)

Postać BCNF jest silniejszą wersją 3NF. Wymaga, aby każdy wyznacznik zależności funkcyjnej był kluczem kandydującym. W analizowanym schemacie:

- Klucze główne są proste i sztuczne (`BIGSERIAL`).
- Nie występują nietrywialne zależności funkcyjne, w których atrybut niekluczowy determinowałby część klucza głównego.

Przyjęta struktura, w której każda encja (Pracownik, Projekt, Urlop) posiada własną tabelę i jednoznaczny identyfikator, a relacje realizowane są przez klucze obce, gwarantuje spełnienie postulatów BCNF.

3.9.5 Podsumowanie normalizacji

Zaprojektowana baza danych została znormalizowana do poziomu 3NF/BCNF. Proces ten zapewnił:

1. **Eliminację redundancji** – nazwy działów, stanowisk czy typów urlopów przechowywane są jednokrotnie.
2. **Spójność danych** – modyfikacja danych słownikowych jest natychmiast widoczna dla wszystkich powiązanych rekordów.
3. **Optymalizację struktury** – mniejszy rozmiar tabel operacyjnych (przechowywanie liczb `INT` zamiast ciągów `VARCHAR`).

3.10 Zaprojektowanie operacji na danych

W celu realizacji wymagań funkcjonalnych systemu, takich jak raportowanie, automatyzacja naliczania wynagrodzeń oraz audyt bezpieczeństwa, zaprojektowano zestaw obiektów bazodanowych: widoków (Views), procedur składowanych (Stored Procedures) oraz wyzwalaczy (Triggers). Poniżej przedstawiono ich specyfikację oraz implementację w języku SQL.

3.11 Widoki (Views) - Warstwa prezentacji danych

Widoki zostały zaprojektowane w celu uproszczenia dostępu do złożonych danych dla aplikacji klienckiej oraz zapewnienia bezpieczeństwa poprzez ukrycie bezpośrednich struktur tabel.

3.11.1 Statystyki pracownika

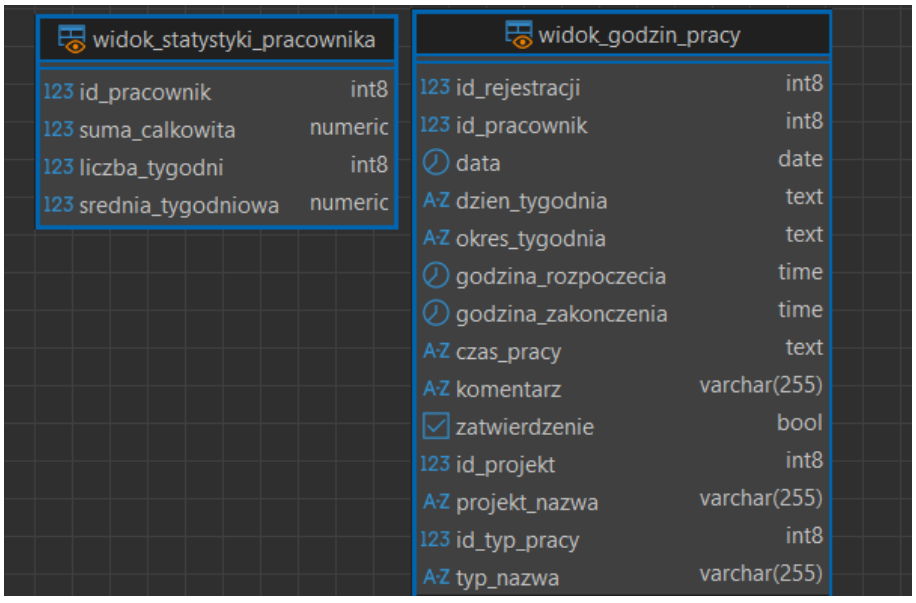
Widok `widok_statystyki_pracownika` służy do szybkiego generowania raportów wydajności. Agreguje on dane z ewidencji czasu pracy, obliczając sumaryczną liczbę przepracowanych godzin oraz średnią tygodniową.
Realizowana funkcja: Raportowanie i podgląd wyników pracownika.

```
CREATE OR REPLACE VIEW widok_statystyki_pracownika AS
SELECT
    id_pracownik,
    round(sum(EXTRACT(epoch FROM godzina_zakonczenia - godzina_rozpoczecia) /
    ↪  3600::numeric), 2)
        AS suma_calkowita,
    count(DISTINCT date_trunc('week'::text, data::timestamp with time zone))
        AS liczba_tygodni,
    CASE
        WHEN count(DISTINCT date_trunc('week'::text, data::timestamp with time
        ↪  zone)) > 0
        THEN round(sum(EXTRACT(epoch FROM godzina_zakonczenia -
        ↪  godzina_rozpoczecia) / 3600::numeric)
             / count(DISTINCT date_trunc('week'::text, data::timestamp with time
             ↪  zone))::numeric, 2)
        ELSE 0::numeric
    END AS srednia_tygodniowa
FROM rejestracja_godzin_pracy
GROUP BY id_pracownik;
```

3.11.2 Szczegółowa ewidencja czasu pracy

Widok `widok_godzin_pracy` prezentuje czytelne dla człowieka dane dotyczące czasu pracy. Zastępuje identyfikatory liczbowe (ID) nazwami projektów i typów pracy (poprzez złączenia `JOIN`) oraz formatuje daty do postaci "Dzień Tygodnia"i zakresów tygodniowych.
Realizowana funkcja: Przejrzysta historia pracy dla pracownika i managera.

```
CREATE OR REPLACE VIEW widok_godzin_pracy AS
SELECT
    r.id_rejestracji,
    r.id_pracownik,
    r.data,
    TRIM(BOTH FROM to_char(r.data::timestamp with time zone, 'Day'::text)) AS
    ↪  dzien_tygodnia,
    (to_char(date_trunc('week'::text, r.data::timestamp with time zone),
    ↪  'DD.MM.YYYY'::text)
     || ' - '::text) || to_char(date_trunc('week'::text, r.data::timestamp with
     ↪  time zone)
     + '6 days'::interval, 'DD.MM.YYYY'::text) AS okres_tygodnia,
    r.godzina_rozpoczecia,
    r.godzina_zakonczenia,
    to_char(r.godzina_zakonczenia - r.godzina_rozpoczecia, 'HH24:MI'::text) AS
    ↪  czas_pracy,
    r.komentarz,
    r.zatwierdzenie,
    p.id_projekt,
    p.nazwa AS projekt_nazwa,
    t.id_typ_pracy,
    t.nazwa AS typ_nazwa
FROM rejestracja_godzin_pracy r
    LEFT JOIN projekt p ON r.id_projekt = p.id_projekt
    JOIN typ_pracy t ON r.id_typ_pracy = t.id_typ_pracy;
```



Rysunek 3: Reprezentacja graficzna użytych widoków w narzędziu bazodanowym

3.12 Procedury składowane - Automatyzacja procesów

Do realizacji złożonej logiki biznesowej po stronie serwera bazy danych wykorzystano procedury składowane.

3.12.1 Generowanie listy płac

Procedura `generuj_wyplaty_za_miesiac` automatyzuje proces księgowy. Pobiera zatwierdzone godziny pracy z zadanego okresu, mnoży je przez stawkę godzinową pracownika i tworzy wpisy w tabeli `historia_wyplat`.
Kluczowe mechanizmy:

- Filtrowanie tylko zatwierdzonych godzin (`zatwierdzenie = true`).
- Obliczanie kwot z precyzją numeryczną.
- Zabezpieczenie przed duplikacją wypłat (klauzula `NOT EXISTS`).

```
CREATE OR REPLACE PROCEDURE generuj_wyplaty_za_miesiac(
    IN p_data_poczatkowa date,
    IN p_data_koncowa date)
LANGUAGE plpgsql
AS $procedure$
DECLARE
    v_opis TEXT;
    v_id_statusu INT;
    v_id_typu INT;
BEGIN
    -- Pobranie domyślnych ID statusów
    SELECT id_status_wyplaty INTO v_id_statusu FROM status_wyplaty WHERE nazwa =
    ↪  'Oczekuje' LIMIT 1;
    IF v_id_statusu IS NULL THEN v_id_statusu := 2; END IF;

    SELECT id_typ_wyplaty INTO v_id_typu FROM typ_wyplaty WHERE nazwa = 'Godziny
    ↪  Pracy' LIMIT 1;
    IF v_id_typu IS NULL THEN v_id_typu := 1; END IF;

    v_opis := 'Automatyczna wypłata: ' || TO_CHAR(p_data_poczatkowa, 'YYYY-MM');

    -- Wstawienie obliczonych wypłat
    INSERT INTO historia_wyplat (
        id_pracownik, data, wyplata, id_status_wyplaty, opis, id_typ_wyplaty
    )
    SELECT
        p.id_pracownik,
        NOW(),
        CAST(SUM(EXTRACT(EPOCH FROM (r.godzina_zakonczenia -
        ↪  r.godzina_rozpoczecia)) / 3600.0)
             * p.wynagrodzenie_pln_g AS NUMERIC(10,2)),
        v_id_statusu,
        v_opis,
        v_id_typu
    FROM rejestracja_godzin_pracy r
    JOIN pracownik p ON r.id_pracownik = p.id_pracownik
    WHERE r.zatwierdzenie = true
      AND r.data >= p_data_poczatkowa
      AND r.data <= p_data_koncowa
```

```
      AND NOT EXISTS (
          SELECT 1 FROM historia_wyplat hw
          WHERE hw.id_pracownik = p.id_pracownik AND hw.opis = v_opis
      )
    GROUP BY p.id_pracownik, p.wynagrodzenie_pln_g
    HAVING SUM(EXTRACT(EPOCH FROM (r.godzina_zakonczenia -
    ↪  r.godzina_rozpoczecia))) > 0;

    RAISE NOTICE 'Wyplaty wygenerowane dla okresu % - %', p_data_poczatkowa,
    ↪  p_data_koncowa;
END;
$procedure$;
```

3.13 Wyzwalacze (Triggers) - Audyt zmian

W celu zapewnienia integralności i śledzenia zmian w systemie, zaimplementowano mechanizm logowania operacji (Insert/Update/Delete).

3.13.1 Logowanie zmian danych pracownika

Funkcja `log_zmian_pracownicy` jest wywoływana automatycznie przy każdej modyfikacji danych. Zapisuje ona starą i nową wersję rekordu w formacie JSONB, co pozwala na pełną historię zmian (kto, kiedy, co i z jakiego IP zmienił).

```
CREATE OR REPLACE FUNCTION log_zmian_pracownicy()
RETURNS trigger
LANGUAGE plpgsql
AS $function$
DECLARE
    operacja TEXT;
BEGIN
    IF (TG_OP = 'INSERT') THEN
        operacja := 'INSERT';
        INSERT INTO logi_operacji (
            id_pracownik, nazwa_operacji, tabela, opis,
            data_operacji, adres_ip, nowa_wartosc
        ) VALUES (
            NULLIF(current_setting('app.current_user_id', TRUE), '')::INT,
            operacja, TG_TABLE_NAME, 'Dodano nowy rekord.', NOW(),
            current_setting('app.client_ip', TRUE), to_jsonb(NEW)::TEXT
        );
        RETURN NEW;

    ELSIF (TG_OP = 'UPDATE') THEN
        operacja := 'UPDATE';
        IF NEW IS DISTINCT FROM OLD THEN
            INSERT INTO logi_operacji (
                id_pracownik, nazwa_operacji, tabela, opis,
                data_operacji, adres_ip, stara_wartosc, nowa_wartosc
            ) VALUES (
                NULLIF(current_setting('app.current_user_id', TRUE), '')::INT,
                operacja, TG_TABLE_NAME, 'Zaktualizowano rekord.', NOW(),
                current_setting('app.client_ip', TRUE),
                to_jsonb(OLD)::TEXT, to_jsonb(NEW)::TEXT
            );
```

```
        END IF;
        RETURN NEW;

    ELSIF (TG_OP = 'DELETE') THEN
        operacja := 'DELETE';
        INSERT INTO logi_operacji (
            id_pracownik, nazwa_operacji, tabela, opis,
            data_operacji, adres_ip, stara_wartosc
        ) VALUES (
            NULLIF(current_setting('app.current_user_id', TRUE), '')::INT,
            operacja, TG_TABLE_NAME, 'Usunięto rekord.', NOW(),
            current_setting('app.client_ip', TRUE), to_jsonb(OLD)::TEXT
        );
        RETURN OLD;
    END IF;
END;
$function$;
```

4 Projekt funkcjonalny

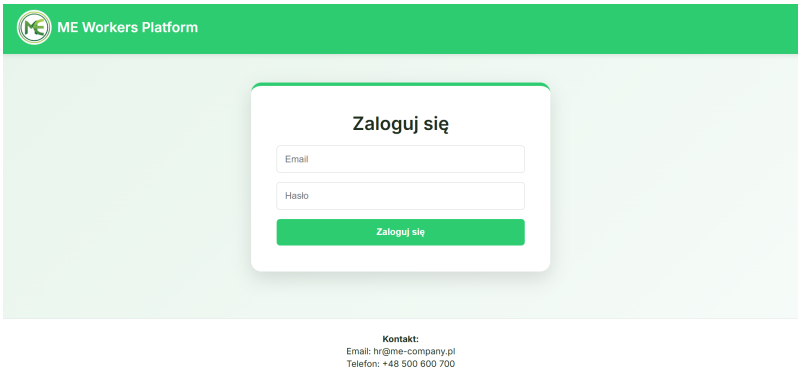
Projekt funkcjonalny definiuje warstwę prezentacji systemu oraz interakcję użytkownika z aplikacją. System został zrealizowany w architekturze klient-serwer, wykorzystując język Java (Spring Boot) po stronie serwera oraz HTML/CSS/JavaScript dla warstwy prezentacji. Kluczowym elementem interfejsu jest dynamiczne zarządzanie danymi tabelarycznymi.

4.1 Zdefiniowanie panelu sterowania aplikacji

Panel sterowania stanowi centralny punkt dostępu do funkcji systemu. Ze względu na wymogi bezpieczeństwa, dostęp do panelu poprzedzony jest procesem autoryzacji.

4.1.1 System logowania i role

W systemie wyróżniono trzy kluczowe role użytkowników, determinujące zakres dostępnych funkcjonalności: **ADMIN**, **USER** (Pracownik) oraz **ACCOUNTANT** (Księgowy). Rejestracja użytkowników odbywa się wyłącznie przez Administratora, co eliminuje ryzyko nieautoryzowanego dostępu z zewnątrz.

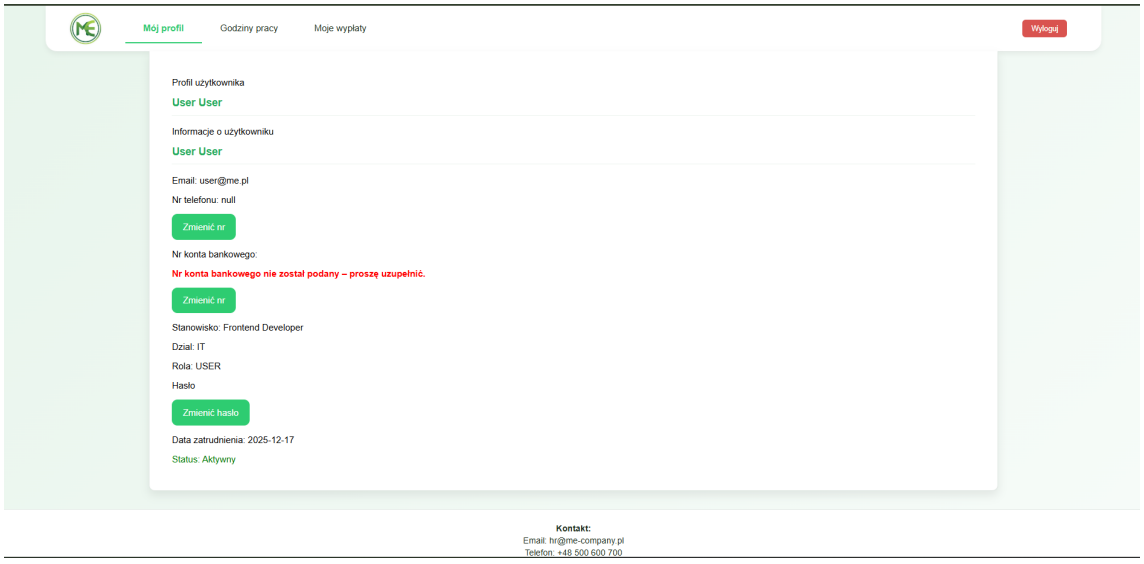


Rysunek 4: Ekran logowania do systemu

4.1.2 Struktura paneli sterowania

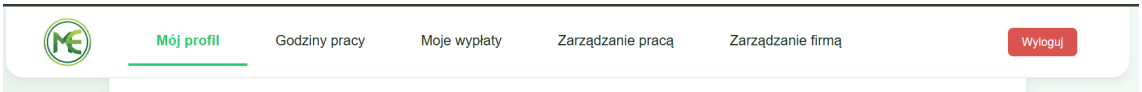
Po poprawnym zalogowaniu, użytkownik jest przekierowywany do dedykowanego panelu sterowania (Dashboard). Interfejs oparty jest na systemie zakładek, co ułatwia nawigację między modułami.

1. Panel Użytkownika (Pracownika) – umożliwia podgląd danych osobowych, edycję ewidencji czasu pracy oraz weryfikację historii wypłat.



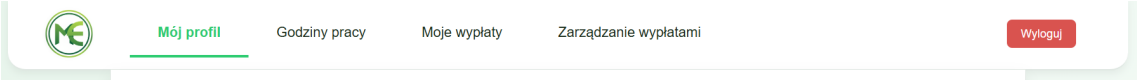
Rysunek 5: Panel sterowania - widok Użytkownika

2. Panel Administratora – posiada pełen zakres uprawnień. Oprócz funkcjonalności standardowego użytkownika, administrator ma dostęp do modułów zarządzania strukturą firmy (działy, stanowiska) oraz kontami pracowników.



Rysunek 6: Panel sterowania - widok Administratora

3. Panel Księgowego – rozszerza uprawnienia podstawowe o moduł finansowy, umożliwiający generowanie list płac, zatwierdzanie przelewów oraz analizę kosztów pracowniczych.

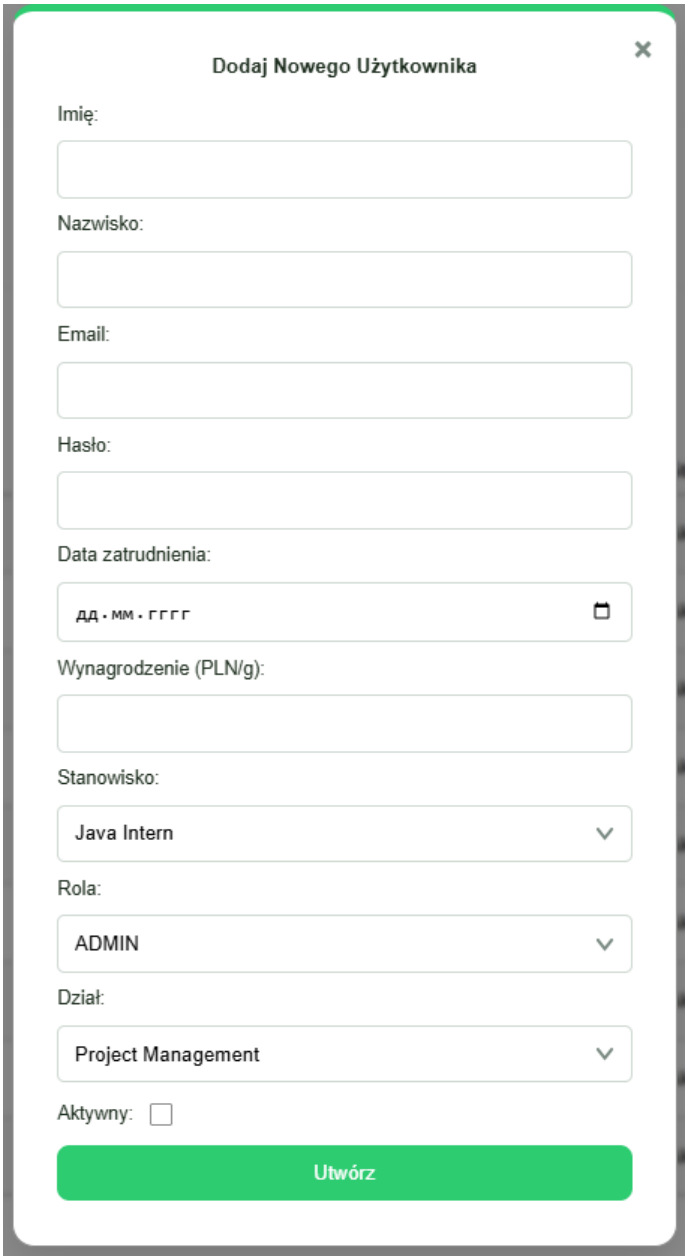


Rysunek 7: Panel sterowania - widok Księgowego

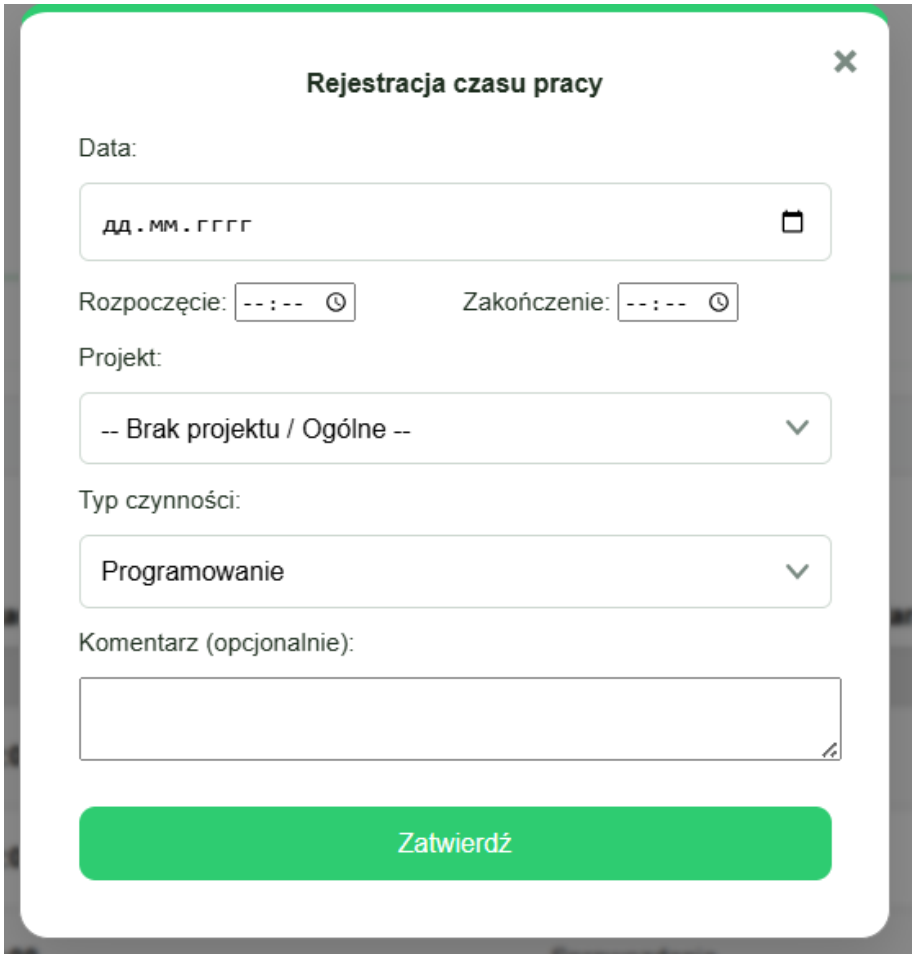
4.2 Interfejsy do prezentacji, edycji i obsługi danych

Interfejs użytkownika został zaprojektowany z naciskiem na ergonomię wprowadzania danych. Do obsługi operacji CRUD (Create, Read, Update, Delete) wykorzystano okna modalne, które pozwalają na edycję danych bez konieczności przeładowywania kontekstu strony.

Formularze wprowadzania danych: Każdy formularz wyposażony jest w walidację danych wejściowych oraz słowniki rozwijane (combobox), pobierające aktualne dane z bazy (np. lista dostępnych stanowisk).



Rysunek 8: Formularz dodawania nowego użytkownika (Panel Administratora)



Rysunek 9: Formularz rejestracji czasu pracy (Panel Użytkownika)

4.3 Wizualizacja danych (Raporty)

Warstwa prezentacji danych opiera się na zaawansowanych mechanizmach tabelarycznych, realizowanych po stronie klienta (Client-Side) przy użyciu biblioteki **DataTables** oraz frameworka **jQuery**. Rozwiązanie to zapewnia płynność działania oraz redukuje obciążenie serwera.

4.3.1 Technologie wizualizacji

W projekcie zintegrowano następujące moduły wizualizacyjne:

- **DataTables Core** – silnik renderujący interaktywne siatki danych z obsługą sortowania i stronicowania.
- **DataTables Buttons & JSZip** – moduły umożliwiające eksport raportów do formatu Excel (`.xlsx`) oraz generowanie widoków do druku.
- **RowGroup** – rozszerzenie pozwalające na wizualne grupowanie wierszy (np. sumowanie godzin w obrębie tygodnia).

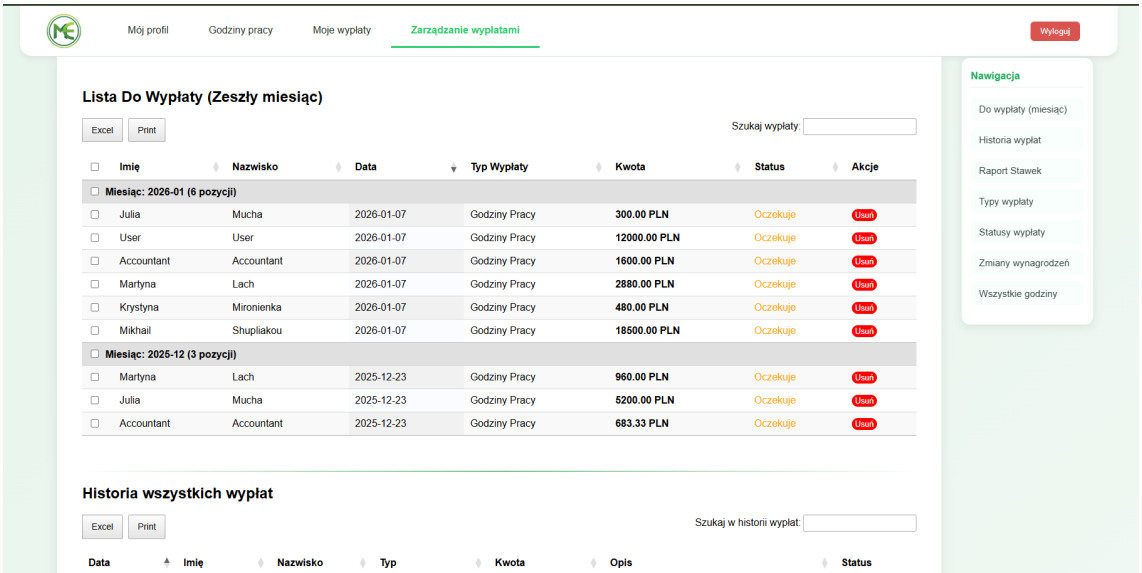
4.3.2 Niestandardowa nawigacja (Group Pagination)

Ze względu na specyfikę danych kadrowych (cykle tygodniowe/miesięczne), standardowa paginacja została zastąpiona **autorskim algorytmem nawigacji opartej na grupach**. Mechanizm ten:

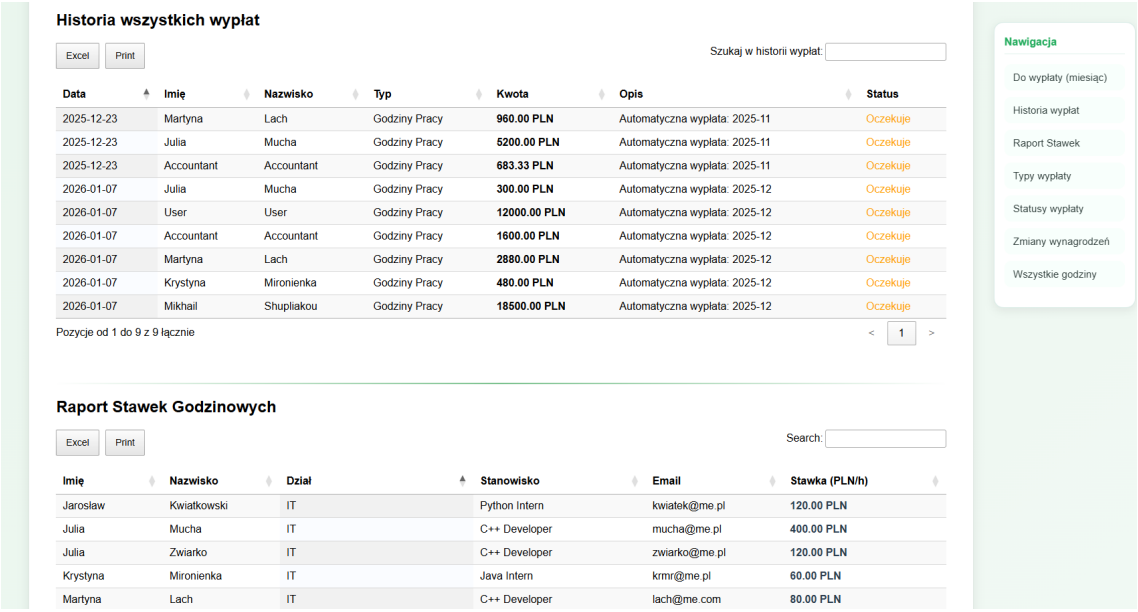
1. Analizuje dane i tworzy listę unikalnych okresów (tygodni/miesięcy).
2. Nadpisuje filtr wyszukiwania (`$.fn.dataTable.ext.search`), wyświetlając rekordy tylko dla wybranego okresu.
3. Udostępnia dedykowane przyciski nawigacyjne (*„Starsze”*, *„Nowsze”*), co znacznie ułatwia analizę kart pracy.

4.3.3 Przykłady generowanych raportów

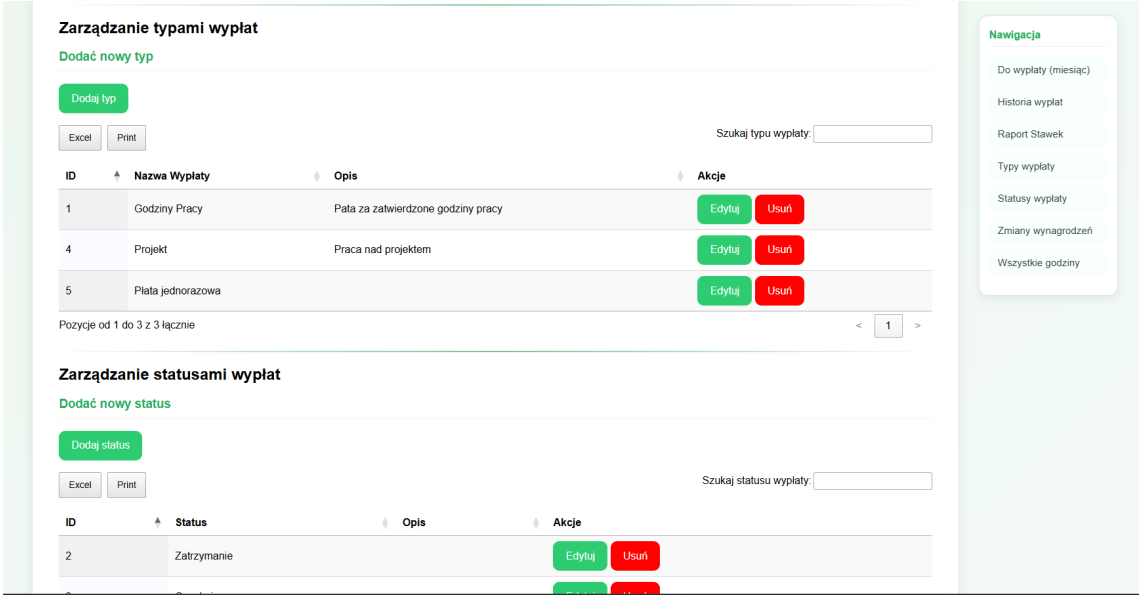
System umożliwia generowanie szeregu raportów operacyjnych i finansowych. Poniżej przedstawiono kluczowe wizualizacje dostępne dla Księgowego i Administratora.



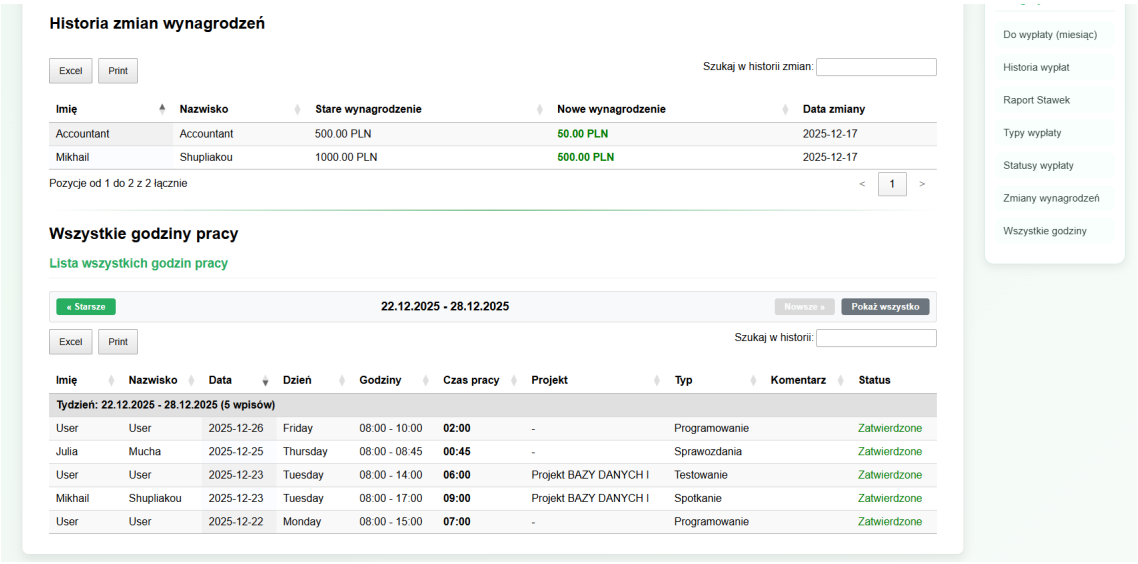
Rysunek 10: Raport: Lista płac oczekujących na realizację



Rysunek 11: Raport: Historia wypłat i stawek godzinowych pracowników



Rysunek 12: Słowniki systemowe: Typy i statusy wypłat



Rysunek 13: Raport łączony: Historia zmian wynagrodzeń i ewidencja godzin

5 Wprowadzanie danych

System realizuje proces wprowadzania i przetwarzania danych w modelu hybrydowym, łącząc interaktywne formularze użytkownika z automatycznymi procesami bazy danych. Przyjęte rozwiązania gwarantują integralność danych oraz minimalizują ryzyko błędów ludzkich.

5.1 Wprowadzanie ręczne (Interfejs użytkownika)

Podstawowym sposobem zasilania systemu danymi jest wprowadzanie ręczne za pośrednictwem aplikacji webowej. Proces ten odbywa się z wykorzystaniem formularzy i okien modalnych.

- **Formularze webowe:** Interfejsy zbudowane w technologii HTML/Thymeleaf umożliwiają wprowadzanie danych do słowników (np. Dodaj Dział), danych kadrowych (np. Rejestracja Pracownika) oraz operacyjnych (np. Ewidencja Czasu Pracy).
- **Walidacja danych:** Wprowadzanie danych jest nadzorowane przez dwuetapowy mechanizm walidacji:
  - *Client-side (JavaScript):* Sprawdzenie formatów dat, wymagalności pól oraz poprawności logicznej (np. godzina zakończenia pracy nie może być wcześniejsza niż rozpoczęcia) przed wysłaniem formularza.
  - *Server-side (Java/Spring):* Ostateczna weryfikacja typów danych i reguł biznesowych przed zapisem do bazy PostgreSQL.

- **Listy wyboru (Słowniki):** W celu uniknięcia błędów literowych, pola relacyjne (np. wybór stanowiska, projektu) są realizowane poprzez elementy typu `select`, pobierające aktualne wartości ze słowników bazodanowych.

5.2 Wprowadzanie automatyczne (Logika systemowa)

Część danych w systemie jest generowana automatycznie, bez bezpośredniego udziału użytkownika, co zapewnia spójność obliczeń i bezpieczeństwo audytowe.

- **Automatyzacja płac (Procedury składowane):** Tabela `historia_wyplat` jest zasilana automatycznie poprzez wywołanie procedury SQL `generuj_wyplaty_za_miesiac`. System samodzielnie pobiera zatwierdzone godziny, mnoży je przez stawki pracownika i tworzy rekordy wypłat.
- **Logi systemowe (Wyzwalacze):** Tabela `logi_operacji` jest wypełniana automatycznie przez wyzwalacze (*triggers*) bazy danych przy każdej operacji `INSERT`, `UPDATE` lub `DELETE`. Zapisywane są tam metadane takie jak: data operacji, adres IP oraz stara i nowa wartość rekordu.
- **Pola wyliczane:** Niektóre dane, takie jak "Czas trwania pracy"lub "Suma tygodniowa", nie są wprowadzane ręcznie, lecz wyliczane dynamicznie przez widoki SQL (`VIEWS`) na podstawie surowych danych wejściowych.

5.3 Import i Eksport danych

System wspiera wymianę danych z otoczeniem zewnętrznym, co ułatwia pracę działu księgowości i kadr.

- **Eksport danych:** Dzięki wykorzystaniu biblioteki *DataTables Buttons*, każdy widok tabelaryczny (Listy płac, Ewidencja godzin, Raporty) posiada wbudowaną funkcję eksportu danych do formatów:
  - **Excel (.xlsx):** Umożliwia dalszą obróbkę danych w arkuszach kalkulacyjnych.
  - **PDF / Druk:** Umożliwia generowanie sformatowanych dokumentów (np. pasków wynagrodzeń) gotowych do podpisu.
- **Import danych:** Wprowadzanie danych odbywa się poprzez interfejs aplikacji, który pełni rolę importera danych wejściowych do struktur relacyjnych bazy danych, mapując obiekty formularza na rekordy tabel.

6 Dokumentacja użytkownika: krótka instrukcja obsługi

Niniejsza instrukcja opisuje podstawowe czynności niezbędne do prawidłowej obsługi Systemu Zarządzania Wynagrodzeniami. Aplikacja dostępna jest z poziomu przeglądarki internetowej.

6.1 Uruchomienie i logowanie do systemu

1. **Dostęp:** Otwórz przeglądarkę internetową (zalecane: Google Chrome, Firefox) i wprowadź adres serwera aplikacji (np. `http://localhost:8082`).
2. **Logowanie:** Na ekranie startowym pojawi się formularz logowania.
   - Wprowadź swój login (adres e-mail) oraz hasło.
   - Kliknij przycisk **Zaloguj się**.
3. **Uwagi:** W przypadku zapomnienia hasła lub braku konta, należy skontaktować się z Administratorem systemu (brak możliwości samodzielnej rejestracji ze względów bezpieczeństwa).

6.2 Instrukcja dla Pracownika (Rola: USER)

Pracownik ma dostęp do panelu "Mój Profil", gdzie może zarządzać czasem pracy i przeglądać zarobki.

6.2.1 Rejestracja czasu pracy

1. Przejdź do zakładki **Ewidencja Czasu**.
2. Kliknij przycisk **Dodaj wpis**.
3. W oknie formularza wybierz:
   - Datę pracy,
   - Godzinę rozpoczęcia i zakończenia,
   - Projekt, nad którym pracowano.
4. Kliknij **Zapisz**. System automatycznie przeliczy czas pracy.

6.2.2 Sprawdzanie wypłat

1. Przejdź do zakładki **Moje Wypłaty**.
2. Tabela wyświetla historię przelewów. Skorzystaj z filtrów nad tabelą (przycisk *„Starsze”/„Nowsze”*), aby przeglądać dane z poprzednich miesięcy.
3. Aby wydrukować pasek wynagrodzeń, kliknij przycisk **Print** lub **Excel** nad tabelą.

6.3 Instrukcja dla Księgowego (Rola: ACCOUNTANT)

Księgowy odpowiada za weryfikację godzin i generowanie listy płac.

6.3.1 Generowanie listy płac

1. Przejdź do panelu **Zarządzanie wypłatami**.
2. Wybierz zakres dat (miesiąc rozliczeniowy).
3. System uruchomi procedurę obliczającą należności na podstawie zatwierdzonych godzin.
4. Nowe pozycje pojawią się w tabeli ze statusem *„Oczekuje”*.

6.3.2 Zatwierdzanie wypłat

1. W tabeli **Lista Płac** zaznacz checkboxami wypłaty, które chcesz zrealizować (możesz zaznaczyć całą grupę klikając w nagłówek miesiąca).
2. Kliknij przycisk **Zatwierdź zaznaczone**.
3. Status wypłat zmieni się na *„Zatwierdzone”*, a pracownicy zobaczą je w swoich panelach.

6.4 Instrukcja dla Administratora (Rola: ADMIN)

Administrator zarządza strukturą firmy i dostępami.

6.4.1 Dodawanie nowego pracownika

1. Przejdź do zakładki **Pracownicy**.
2. Kliknij przycisk **Dodaj Pracownika**.
3. Wypełnij formularz:
   - Dane osobowe (Imię, Nazwisko, Email),
   - Dane zatrudnienia (Dział, Stanowisko, Stawka godzinowa),
   - Rola w systemie (User, Accountant lub Admin).
4. Kliknij **Utwórz**. Pracownik może teraz zalogować się używając podanego adresu e-mail.

6.4.2 Zarządzanie słownikami

1. Aby dodać nowe stanowisko lub dział, przejdź do zakładki **Słowniki**.
2. Wybierz odpowiednią tabelę (np. Działy) i użyj opcji edycji, aby dostosować strukturę organizacyjną firmy.

7 Opracowanie dokumentacji technicznej

Dokumentacja techniczna kodu została opracowana zgodnie ze standardami języka Java, wykorzystując narzędzie **Javadoc**. Rozwiązanie to pozwala na automatyczne generowanie spójnej dokumentacji w formacie HTML bezpośrednio z komentarzy zawartych w kodzie źródłowym.

7.1 Standard dokumentowania kodu

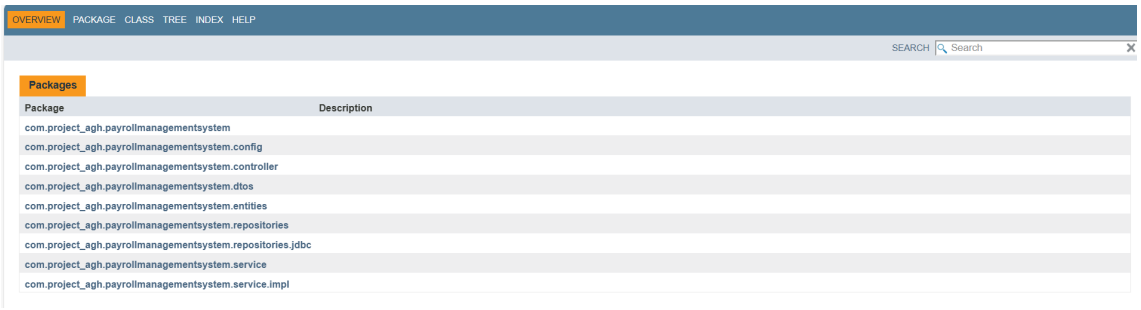
Wszystkie kluczowe klasy, interfejsy oraz metody (w szczególności w warstwie serwisowej oraz kontrolerach) zostały opatrzone komentarzami blokowymi. Komentarze te zawierają opis odpowiedzialności danej klasy oraz specyfikację metod.

Wykorzystano standardowe znaczniki Javadoc:

- `@param` – opis parametrów wejściowych metody,
- `@return` – opis zwracanej wartości,
- `@throws` – opis wyjątków, które mogą zostać rzucone przez metodę,
- `@see` – odnośniki do powiązanych elementów systemu.

7.2 Wygenerowana dokumentacja (HTML)

Dzięki zastosowaniu narzędzia Javadoc, wygenerowano zestaw plików HTML, które umożliwiają nawigację po strukturze projektu. Dokumentacja ta jest niezbędna dla nowych programistów, którzy w przyszłości będą rozwijać system, zapewniając im szybki wgląd w API klas i zależności między nimi.



Rysunek 14: Dokumentacja techniczna

7.3 Dokumentacja API (Swagger)

Dodatkowo, ze względu na wykorzystanie frameworka Spring Boot, struktura końcówek REST (endpoints) jest możliwa do podglądu przy użyciu standardu OpenAPI. Definicje kontrolerów mapują się na czytelną dokumentację interfejsów, co ułatwia integrację frontend'u z backend'em.

8 Źródła

1. Neon DB - Baza danych użyta w projekcie znajdująca się w chmurze
2. Gotowy projekt na GitHub
3. DataTables. jQuery