

# AKADEMIA NAUK STOSOWANYCH W NOWYM SĄCZU

Wydział Nauk Inżynierskich  
Katedra Informatyki

## DOKUMENTACJA PROJEKTOWA

Bazy Danych

### **E-firma - RCP i system kart**

Autor:  
Agnieszka Ryś  
Rafał Madoń

Prowadzący:  
mgr inż. Nikodem Bulanda

Nowy Sącz 2024

# Spis treści

<b>1. Cel projektu</b>	<b>4</b>
<b>2. Wymagania funkcjonalne i нефункционалне</b>	<b>5</b>
<b>3. Dobór technologii</b>	<b>6</b>
3.1. Laravel [1] . . . . .	6
3.2. React [2] . . . . .	6
3.3. MySQL . . . . .	6
<b>4. Diagram przypadków użycia</b>	<b>7</b>
4.1. Scenariusze użytkownika . . . . .	9
4.1.1. Scenariusz dla Administratora: . . . . .	9
4.1.2. Scenariusz dla Pracownika: . . . . .	12
4.1.3. Scenariusz dla Gościa: . . . . .	15
<b>5. Triggery</b>	<b>16</b>
5.1. Trigger before_insert_strefy_dostepu . . . . .	17
5.1.1. Opis . . . . .	17
5.1.2. Działanie . . . . .	17
5.2. Trigger after_insert_strefy_dostepu . . . . .	17
5.2.1. Opis . . . . .	17
5.2.2. Działanie . . . . .	17
5.3. Trigger after_update_strefy_dostepu . . . . .	18
5.3.1. Opis . . . . .	18
5.3.2. Działanie . . . . .	18
5.4. Trigger before_delete_strefy_dostepu . . . . .	18
5.4.1. Opis . . . . .	18
5.4.2. Działanie . . . . .	18
5.5. Podsumowanie . . . . .	19
<b>6. Transakcje</b>	<b>20</b>
6.1. Dodawanie pracownika . . . . .	20

6.2. Aktualizacja pracownika . . . . .	20
6.3. Zmiana statusu konta pracownika . . . . .	21
6.4. Usuwanie pracownika . . . . .	23
<b>7. Funkcje</b>	<b>24</b>
7.1. Działanie . . . . .	24
7.2. Przykład użycia funkcji . . . . .	25
7.3. Wynik . . . . .	25
<b>8. Git</b>	<b>26</b>
8.1. Ilość commitów . . . . .	26
8.2. Ilość zapytań Raw Query . . . . .	26
8.3. Ilość zapytań ORM . . . . .	26
<b>9. Podsumowanie projektu</b>	<b>27</b>
9.1. Zrealizowane zagadnienia . . . . .	27
9.2. Niewykonane zagadnienia . . . . .	29
<b>Literatura</b>	<b>30</b>
<b>Spis rysunków</b>	<b>31</b>
<b>Spis listingów</b>	<b>32</b>

# 1. Cel projektu

## 1. Przed zalogowaniem

- Widoczność aktualności o firmie
- Widoczność podstawowych informacji o firmie
- Dostęp do kontaktu z firmą
- Możliwość zalogowania się

## 2. Interfejs użytkownika:

Zalogowany użytkownik powinien mieć dostępny interfejs, umożliwiający:

- Podgląd ilości dni wolnych i wykorzystanie ich
- Dostęp do ogłoszeń
- Możliwość używania karty, poprzez skanowanie kodu QR do otwarcia drzwi
- Podgląd podstawowych informacji o sobie

## 3. Interfejs Administracyjny:

Administrator powinien mieć dostępny interfejs administracyjny, umożliwiający:

- Dodawanie/edycja użytkownika
- Podgląd informacji o pracowniku
- Wstrzymywanie użytkownikowi dostęp do firmy
- Dodawanie/edycja szkoleń
- Dodawanie/edycja ogłoszeń do pracowników
- Dodawanie/edycja aktualności
- Dodanie/edycja kart, zmiana uprawnień dostępu
- Obliczanie wypłat pracowniczych

## 2. Wymagania funkcjonalne i нефункционалне

### 1. Wymagania funkcjonalne:

- Modyfikacja użytkowników przez admina: Administrator może dodawać, usuwać oraz zmieniać konta pracowników.
- Blokowanie dostępu do konta użytkownika: Administrator ma możliwość wstrzymywania dostępu do konta danemu użytkownikowi.
- Logowanie: Mechanizm uwierzytelniania, umożliwiający użytkownikom logowanie się do systemu.
- Zarządzanie czasem pracy poprzez system kart: Możliwość rejestracji godzin rozpoczęcia i zakończenia pracy dla każdego pracownika.
- Zarządzanie strefami dostępu: Możliwość definiowania stref dostępu w firmie oraz przypisywania pracowników do odpowiednich stref dostępu.
- Obliczanie wypłat pracowniczych: Administrator powinien mieć możliwość obliczania automatycznego wypłat pracowniczych z danego miesiąca.
- Dodawanie i edycja aktualności oraz ogłoszeń: Administrator ma możliwość dodawania oraz edycji aktualności i ogłoszeń, dla danej grupy w firmie.

### 2. Wymagania нефункционалне:

- Bezpieczeństwo: Zapewnienie bezpiecznej autoryzacji i uwierzytelniania, oraz ochrona danych pracowników przed nieautoryzowanym dostępem.
- Wydajność: System powinien być wystarczająco wydajny, aby obsłużyć dużą liczbę użytkowników jednocześnie oraz zapewnić szybki dostęp do danych.
- Dostępność: System powinien być dostępny 24/7, z minimalnymi przerwami na konserwację i aktualizacje.
- Skalowalność: Możliwość elastycznego skalowania systemu w miarę wzrostu liczby użytkowników i danych.
- Łatwość obsługi: Intuicyjny interfejs użytkownika, który będzie łatwy w obsłudze dla pracowników i administratorów.

## 3. Dobór technologii

### 3.1. Laravel [1]

Laravel jest popularnym frameworkiem PHP, który oferuje wiele funkcji ułatwiających tworzenie aplikacji internetowych, takich jak:

- **Szybkość i wydajność:** Laravel oferuje wydajne narzędzia do tworzenia aplikacji, co pozwala nam skupić się na kodowaniu, a nie na tworzeniu podstawowych funkcjonalności.
- **Obsługa baz danych:** Dzięki wbudowanym mechanizmom ORM (Object-Relational Mapping), takim jak Eloquent, Laravel ułatwia interakcję z bazą danych.

### 3.2. React [2]

React to darmowa i otwartoźródłowa biblioteka JavaScript służąca do budowania interfejsów użytkownika (UI). React wyróżnia się swoim komponentowym podejściem i deklaratywnym stylem programowania. Kluczowe cechy React:

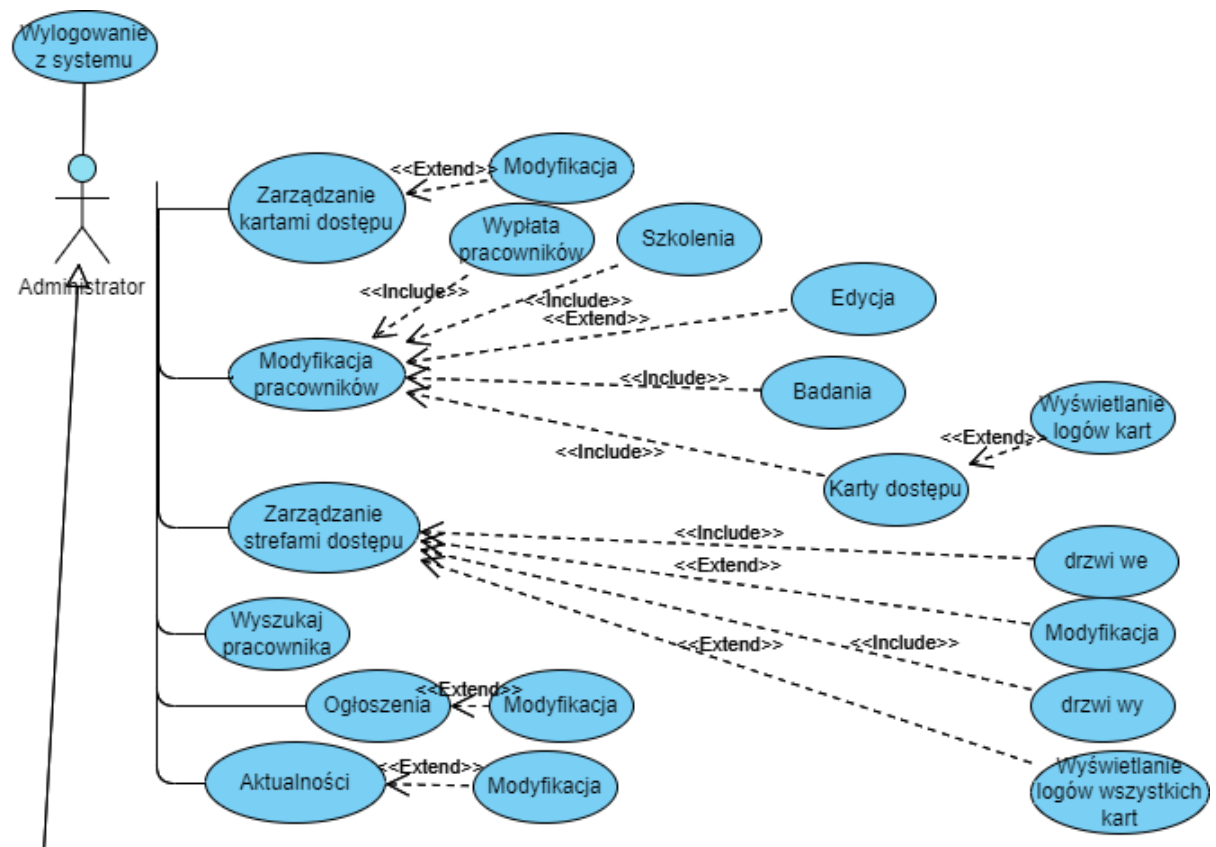
- **Komponentowy:** Interfejs aplikacji budowany jest z małych, ponownie używalnych komponentów
- **Deklaratywny:** Deklarujesz, jak powinien wyglądać interfejs w danym stanie, a React dba o aktualizację i renderowanie zmian.
- **JSX:** Używa składni JSX (JavaScript XML) do opisywania struktury komponentów, co ułatwia czytanie i utrzymanie kodu.

### 3.3. MySQL

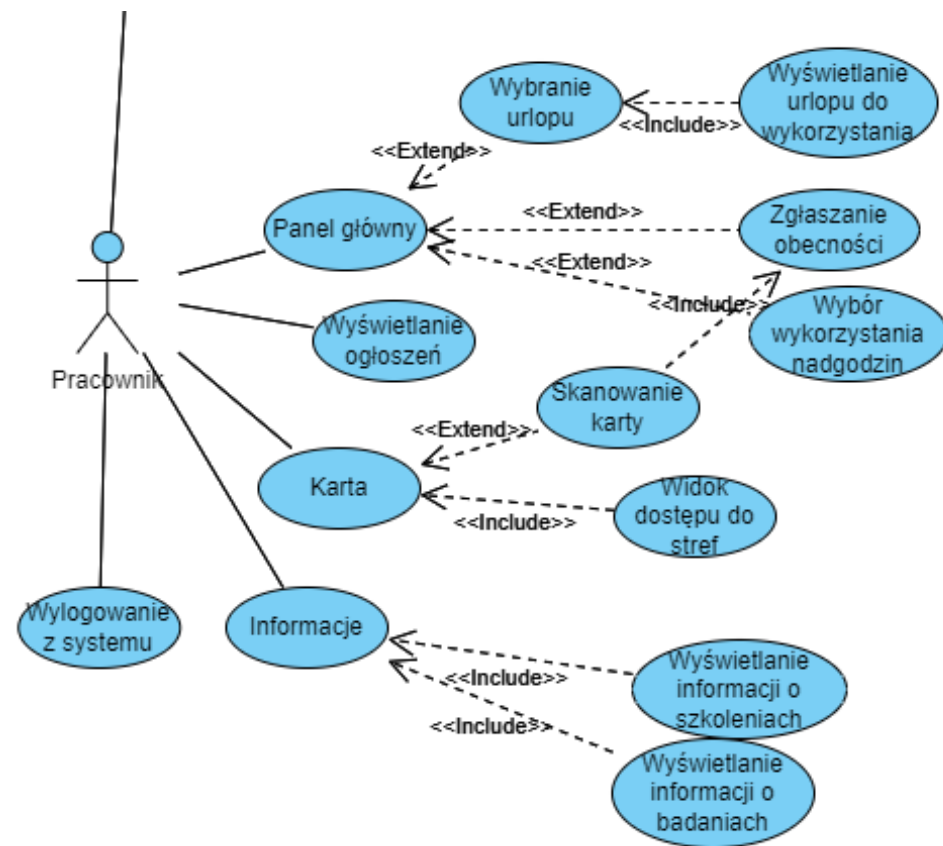
MySQL to otwartoźródłowy system zarządzania bazami danych (SGBD), oparty na relacyjnym modelu danych[3]. Jest szeroko stosowany w różnych środowiskach, dzięki:

- **Otwartość:** Otwartoźródłowy kod umożliwiający dostosowanie i rozszerzenie funkcjonalności.
- **Łatwość użycia:** Intuicyjny interfejs i prosta składnia języka SQL.

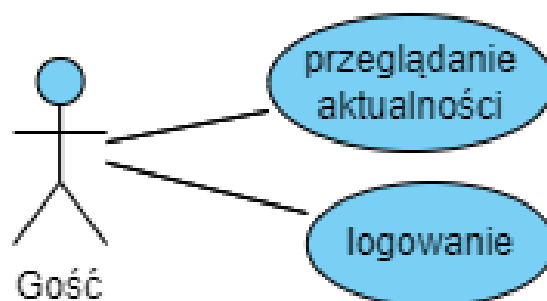
## 4. Diagram przypadków użycia



Rys. 4.1. Diagram przypadków użycia Administratora dla aplikacji E-firma



Rys. 4.2. Diagram przypadków użycia Pracownika dla aplikacji E-firma



Rys. 4.3. Diagram przypadków użycia Gościa dla aplikacji E-firma



## 4.1. Scenariusze użytkownika

### 4.1.1. Scenariusz dla Administratora:

Nr Scenariusza	A1
Aktor	Administrator
Warunki wstępne	Administrator jest zalogowany do aplikacji E-firma.
Scenariusz główny	<ol style="list-style-type: none"> <li>Administrator wybiera opcję "Zarządzanie kartami dostępu".</li> <li>Administrator wyświetla listę wszystkich kart dostępu.</li> <li>Administrator może: <ul style="list-style-type: none"> <li>Dodać nową kartę dostępu, podając imię i nazwisko pracownika, numer karty oraz strefy, do których ma ona dostęp.</li> <li>Edytować istniejącą kartę dostępu, zmieniając jej dane lub przypisując ją do innych stref.</li> <li>Usunąć kartę dostępu.</li> </ul> </li> <li>Administrator przegląda logi kart dostępu, aby zobaczyć, kto i kiedy używał kart dostępu do poszczególnych stref.</li> </ol>
Warunki końcowe	Administrator może skutecznie zarządzać kartami dostępu.
Rezultat	Administrator może dodawać, edytować i usuwać karty dostępu oraz przeglądać logi kart dostępu.

Nr Scenariusza	A2
Aktor	Administrator
Warunki wstępne	Administrator jest zalogowany do aplikacji E-firma.
Scenariusz główny	<ol style="list-style-type: none"> <li>1. Administrator wybiera opcję "Modyfikacja pracowników"</li> <li>2. Administrator wybiera Pracownika z listy</li> <li>3. Administrator może: <ul style="list-style-type: none"> <li>• Zmodyfikować podstawowe dane Pracownika</li> <li>• Wybrać z listy: wypłatę, szkolenie, badanie lub kartę dostępu, a następnie zmodyfikować dane</li> <li>• Zobaczyć logi karty przypisanej do pracownika</li> </ul> </li> </ol>
Warunki końcowe	Administrator może skutecznie modyfikować dowolnego pracownika
Rezultat	Administrator może zmieniać dane takie jak: wypłata pracowników, szkolenia, badania oraz karty dostępu

Nr Scenariusza	A3
Aktor	Administrator
Warunki wstępne	Administrator jest zalogowany do aplikacji E-firma.
Scenariusz główny	<ol style="list-style-type: none"> <li>1. Administrator wybiera opcję "Zarządzanie strefami dostępu"</li> <li>2. Administrator wybiera strefę dostępu z listy</li> <li>3. Administrator może: <ul style="list-style-type: none"> <li>• Zmodyfikować nazwę, wejście lub wyjście oraz wymagane uprawnienia</li> </ul> </li> </ol>
Warunki końcowe	Administrator może skutecznie modyfikować wybraną strefę dostępu
Rezultat	Administrator może zmieniać dane takie jak: nazwa przypisana do danego nr wejścia, rodzaj oraz wymagane uprawnienia.

Nr Scenariusza	A4
Aktor	Administrator
Warunki wstępne	Administrator jest zalogowany do aplikacji E-firma.
Scenariusz główny	<ol style="list-style-type: none"><li>1. Administrator wybiera opcję "Wyszukaj pracownika"</li><li>2. Administrator wybiera odpowiedniego pracownika</li><li>3. System wyświetla informacje o wybranym pracowniku</li></ol>
Scenariusz alternatywny	2a. Administrator wyszukał pracownika, który nie znajduje się w bazie. System wysyła informację o braku takiego pracownika w bazie.
Warunki końcowe	Administrator może skutecznie wyszukiwać pracowników
Rezultat	Administrator może wyszukiwać pracowników oraz wyświetlać jego dane.

**4.1.2. Scenariusz dla Pracownika:**

Nr Scenariusza	P1
Aktor	Pracownik
Warunki wstępne	Pracownik uruchamia zakładkę "panel główny"
Scenariusz główny	<ol style="list-style-type: none"><li>1. System wyświetla ile pracownik ma dni urlopu do wykorzystania.</li><li>2. Pracownik wybiera dni w których chce wykorzystać urlop.</li><li>3. Pracownik wciska guzik "Zatwierdź".</li></ol>
Scenariusz alternatywny	3a. W przypadku wzięcia urlopu ponad ilość dostępnych dni wyskakuje stosowny komunikat o braku takiej ilości dni urlopu.
Warunki końcowe	Pracownik wykorzystuje swój urlop.
Rezultat	Pracownik wykorzystuje swoje dni urlopu.

Nr Scenariusza	P2
Aktor	Pracownik
Warunki wstępne	Pracownik skanuje kod QR w celu wejścia do budynku oraz posiada uruchomioną usługę GPS
Scenariusz główny	<ol style="list-style-type: none"> <li>1. Pracownik klika zielony przycisk "zgłoś obecność" rozpoczynając pracę.</li> <li>2. System za pomocą usługi GPS weryfikuje obecność użytkownika</li> <li>3. System rejestruje czas rozpoczęcia pracy</li> <li>4. Pracownik klika przycisk "koniec pracy", wychodząc z pracy.</li> <li>5. System ponownie weryfikuje obecność użytkownika za pomocą GPS.</li> <li>6. System rejestruje czas zakończenia pracy.</li> </ol>
Scenariusz alternatywny	<p>1a. W przypadku nie zgłoszenia obecności system wysyła ostrzeżenie oraz informuje pracownika o konsekwencjach.</p> <p>2a. W przypadku nie wykrycia obecności pracownika w pobliżu firmy wysyłane jest ostrzeżenie i nie nalicza się czas pracy.</p> <p>4a. W przypadku, gdy pracownik nie zaznaczy zakończenia pracy, godziny pracy nie zostaną naliczone oraz system wyśle ostrzeżenie do pracownika.</p>
Warunki końcowe	Pracownik rejestruje czas pracy.
Rezultat	Skuteczne zarejestrowanie czasu pracy.

Nr Scenariusza	P3
Aktor	Pracownik
Warunki wstępne	Pracownik uruchamia zakładkę "panel główny"
Scenariusz główny	<ol style="list-style-type: none"> <li>1. System wyświetla którą formę wykorzystania nadgodzin mamy wybraną (poprzez podświetlenie odpowiedniego przycisku)</li> <li>2. Pracownik wybiera opcję którą formę chce mieć naliczaną.</li> </ol>
Warunki końcowe	Pracownikowi przyznawana jest forma naliczania nadgodzin w taki sposób jaki wybrał.
Rezultat	Pracownikowi przyznawana jest forma naliczania nadgodzin w taki sposób jaki wybrał.

Nr Scenariusza	P4
Aktor	Pracownik
Warunki wstępne	Pracownik uruchamia zakładkę "Informacje"
Scenariusz główny	<ol style="list-style-type: none"><li>1. System wyświetla informacje o badaniach.</li><li>2. System wyświetla informacje o szkoleniach, datę, oraz pdf możliwego do pobierania ze szczegółowym opisem szkolenia.</li></ol>
Warunki końcowe	Pracownik wyświetla swoje podstawowe informacje
Rezultat	Pracownik wyświetla swoje podstawowe informacje

**4.1.3. Scenariusz dla Gościa:**

Nr Scenariusza	G1
Aktor	Gość
Warunki wstępne	Wejście na stronę www. Niezalogowany użytkownik
Scenariusz główny	<ol style="list-style-type: none"><li>1. Gość wybiera opcję "Zaloguj się".</li><li>2. Gość wpisuje swój login i hasło.</li><li>3. Gość wciska przycisk "Zatwierdź".</li></ol>
Scenariusz alternatywny	3a. Gość wpisał taki login i hasło które nie istnieje w bazie, dostaje komunikat "Błędny login lub hasło", musi wpisać ponownie login i hasło.
Warunki końcowe	Gość skutecznie się zalogował i jest pracownikiem.
Rezultat	Skuteczne zalogowanie jako pracownik wyświetla poszczególny panel główny pracownika.

## 5. Triggery

```
1 DELIMITER //
2 CREATE TRIGGER before_insert_strefy_dostepu
3 BEFORE INSERT ON strefy_dostepu
4 FOR EACH ROW
5 BEGIN
6     IF NEW.nazwa_strefy IS NULL OR NEW.nazwa_strefy = '' THEN
7         SIGNAL SQLSTATE '45000'
8         SET MESSAGE_TEXT = 'Nazwa strefy nie moze byc pusta';
9     END IF;
10 END;
11 //
12 DELIMITER ;
13
14
15
16 DELIMITER //
17 CREATE TRIGGER after_insert_strefy_dostepu
18 AFTER INSERT ON strefy_dostepu
19 FOR EACH ROW
20 BEGIN
21     INSERT INTO log_operacji (akcja, data, szczegoly)
22     VALUES ('INSERT', NOW(), CONCAT('Dodano nowa strefe: ', NEW.
        nazwa_strefy));
23 END;
24 //
25 DELIMITER ;
26
27
28 DELIMITER //
29 CREATE TRIGGER after_update_strefy_dostepu
30 AFTER UPDATE ON strefy_dostepu
31 FOR EACH ROW
32 BEGIN
33     INSERT INTO log_operacji (akcja, data, szczegoly)
34     VALUES ('UPDATE', NOW(), CONCAT('Zaktualizowano strefe ID: ', NEW.
        Strefy_Dostepu_id));
35 END;
36 //
```



```
37 DELIMITER ;
38
39 DELIMITER //
40 CREATE TRIGGER before_delete_strefy_dostepu
41 BEFORE DELETE ON strefy_dostepu
42 FOR EACH ROW
43 BEGIN
44     INSERT INTO log_operacji (akcja, data, szczegoly)
45     VALUES ('DELETE', NOW(), CONCAT('Usunieto strefe ID: ', OLD.
46         Strefy_Dostepu_id));
47 END;
48 //
49 DELIMITER ;
```

## 5.1. Trigger before\_insert\_strefy\_dostepu

### 5.1.1. Opis

Trigger `before_insert_strefy_dostepu` sprawdza, czy kolumna `nazwa_strefy` nie jest pusta lub NULL przed wstawieniem nowego wiersza do tabeli `strefy_dostepu`.

### 5.1.2. Działanie

- Trigger uruchamia się przed każdą operacją `INSERT` na tabeli `strefy_dostepu`.
- Sprawdza, czy wartość `nazwa_strefy` w nowym wierszu jest pusta lub NULL.
- Jeśli warunek jest spełniony, wywołuje błąd z komunikatem 'Nazwa strefy nie może być pusta'.

## 5.2. Trigger after\_insert\_strefy\_dostepu

### 5.2.1. Opis

Trigger `after_insert_strefy_dostepu` rejestruje operację wstawienia nowego wiersza do tabeli `strefy_dostepu` poprzez dodanie wpisu do tabeli `log_operacji`.

### 5.2.2. Działanie

- Trigger uruchamia się po każdej operacji `INSERT` na tabeli `strefy_dostepu`.

- Dodaje nowy wiersz do tabeli `log_operacji`, rejestrując:
  - Typ akcji (`'INSERT'`).
  - Aktualny czas operacji (`NOW()`).
  - Szczegóły operacji (połączenie tekstu `'Dodano nowa strefe: '` z wartością `nazwa_strefy`).

### 5.3. Trigger after\_update\_strefy\_dostepu

#### 5.3.1. Opis

Trigger `after_update_strefy_dostepu` rejestruje operację aktualizacji wiersza w tabeli `strefy_dostepu` poprzez dodanie wpisu do tabeli `log_operacji`.

#### 5.3.2. Działanie

- Trigger uruchamia się po każdej operacji `UPDATE` na tabeli `strefy_dostepu`.
- Dodaje nowy wiersz do tabeli `log_operacji`, rejestrując:
  - Typ akcji (`'UPDATE'`).
  - Aktualny czas operacji (`NOW()`).
  - Szczegóły operacji (połączenie tekstu `'Zaktualizowano strefe ID: '` z wartością `Strefy_Dostepu_id` z nowego wiersza).

### 5.4. Trigger before\_delete\_strefy\_dostepu

#### 5.4.1. Opis

Trigger `before_delete_strefy_dostepu` rejestruje operację usunięcia wiersza z tabeli `strefy_dostepu` poprzez dodanie wpisu do tabeli `log_operacji`.

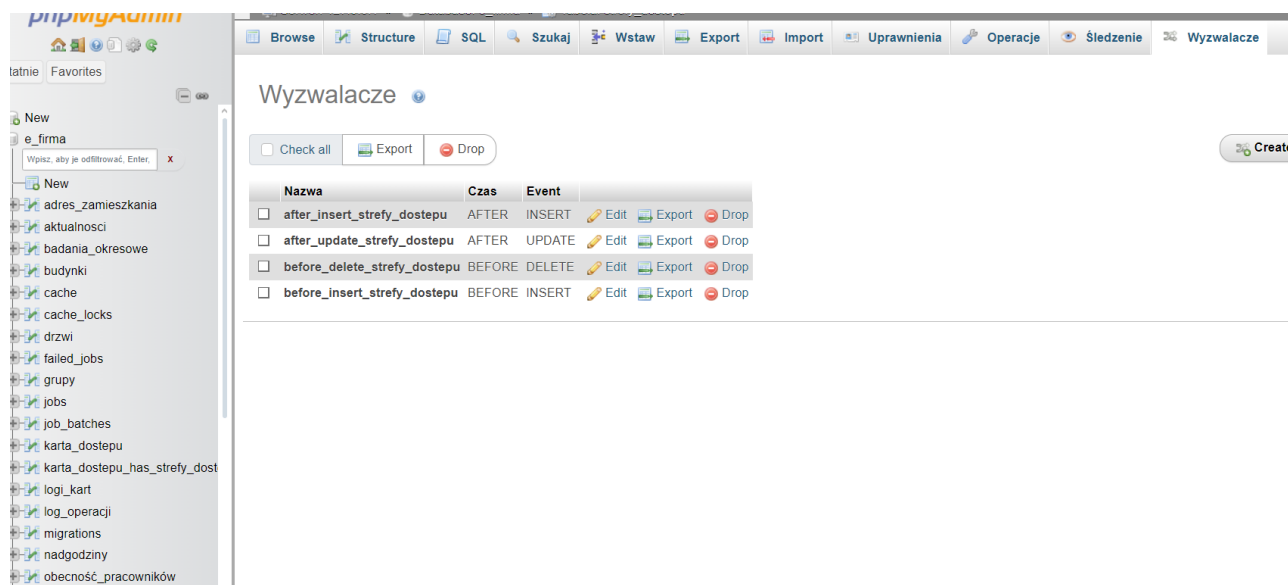
#### 5.4.2. Działanie

- Trigger uruchamia się przed każdą operacją `DELETE` na tabeli `strefy_dostepu`.
- Dodaje nowy wiersz do tabeli `log_operacji`, rejestrując:
  - Typ akcji (`'DELETE'`).
  - Aktualny czas operacji (`NOW()`).
  - Szczegóły operacji (połączenie tekstu `'Usunieto strefe ID: '` z wartością `Strefy_Dostepu_id` ze starego wiersza).

## 5.5. Podsumowanie

Cztery zdefiniowane triggery zapewniają integralność danych oraz umożliwiają śledzenie operacji wstawiania, aktualizacji i usuwania w tabeli `strefy_dostepu`.

Trigger `before_insert_strefy_dostepu` zapewnia, że kolumna `nazwa_strefy` nie jest pusta, podczas gdy pozostałe triggery (`after_insert_strefy_dostepu`, `after_update_strefy_dostepu` oraz `before_delete_strefy_dostepu`) rejestrują szczegóły każdej operacji odpowiednio do tabeli `log_operacji`.



Rys. 5.1. Działające triggery

## 6. Transakcje

Opisujemy transakcje zawarte w kontrolerze `PracownicyController`, który zarządza operacjami na modelu `Pracownicy`.

### 6.1. Dodawanie pracownika

```
1 public function store(Request $request)
2 {
3     DB::beginTransaction();
4     try {
5         $validatedData = $request->validate([
6             'imie' => 'required',
7             'nazwisko' => 'required',
8             'stanowisko_id' => 'required|exists:stanowisko,id',
9             'pensja' => 'required|numeric',
10        ]);
11        $pracownik = Pracownicy::create($validatedData);
12        DB::commit();
13        return response()->json(['message' => 'Pracownik dodany', '
14        pracownik' => $pracownik], 201);
15    } catch (\Exception $e) {
16        DB::rollback();
17        return response()->json(['message' => 'Wystapil blad podczas
18        dodawania pracownika'], 500);
19    }
20 }
```

W tej transakcji:

- Rozpoczynamy transakcję za pomocą `DB::beginTransaction()`.
- Walidujemy dane wejściowe.
- Tworzymy nowy rekord w tabeli `Pracownicy`.
- Jeśli wszystko przebiega pomyślnie, zatwierdzamy transakcję za pomocą `DB::commit()`.
- W przypadku błędu, wycofujemy transakcję za pomocą `DB::rollback()`.

### 6.2. Aktualizacja pracownika

```
1 public function update(Request $request, $id)
2 {
3     DB::beginTransaction();
4     try {
5         $pracownik = Pracownicy::find($id);
6         if (!$pracownik) {
7             return response()->json(['message' => 'Pracownik nie
znaleziony'], 404);
8         }
9         $validatedData = $request->validate([
10             'imie' => 'required',
11             'nazwisko' => 'required',
12             'stanowisko_id' => 'required|exists:stanowisko,id',
13             'pensja' => 'required|numeric',
14         ]);
15         $pracownik->update($validatedData);
16         DB::commit();
17         return response()->json(['message' => 'Pracownik
zaktualizowany', 'pracownik' => $pracownik]);
18     } catch (\Exception $e) {
19         DB::rollback();
20         return response()->json(['message' => 'Wystapil blad podczas
aktualizacji pracownika'], 500);
21     }
22 }
```

W tej transakcji:

- Rozpoczynamy transakcję.
- Znajdujemy pracownika o podanym id.
- Walidujemy dane wejściowe.
- Aktualizujemy rekord pracownika.
- Zatwierdzamy lub wycofujemy transakcję w zależności od wyniku operacji.

### 6.3. Zmiana statusu konta pracownika

```
1 public function change_status(Request $request, $id)
2 {
3     DB::beginTransaction();
4     try {
5         $pracownik = Pracownicy::find($id);
6         if (!$pracownik) {
7             return response()->json(['message' => 'Pracownik nie
znaleziony'], 404);
8         }
9         $pracownik->konto_aktywne = !$pracownik->konto_aktywne;
10        if (!$pracownik->konto_aktywne) {
11            $karty = $pracownik->kartyDostepu;
12            foreach ($karty as $karta) {
13                $karta->karta_aktywna = 0;
14                $karta->save();
15            }
16        }
17        $pracownik->save();
18        DB::commit();
19        return response()->json(['message' => 'Status konta pracownika
zmieniony, zablokowano karty', 'pracownik' => $pracownik]);
20    } catch (\Exception $e) {
21        DB::rollback();
22        return response()->json(['message' => 'Wystapil blad podczas
zmiany statusu konta pracownika'], 500);
23    }
24 }
```

W tej transakcji:

- Rozpoczynamy transakcję.
- Znajdujemy pracownika o podanym id.
- Zmieniamy status konta pracownika.
- Jeśli konto zostaje dezaktywowane, blokujemy wszystkie karty dostępu pracownika.
- Zapisujemy zmiany i zatwierdzamy transakcję.

## 6.4. Usuwanie pracownika

```
1 public function destroy($id)
2 {
3     DB::beginTransaction();
4     try {
5         $pracownik = Pracownicy::find($id);
6         if (!$pracownik) {
7             return response()->json(['message' => 'Pracownik nie
znaleziony'], 404);
8         }
9         $pracownik->delete();
10        DB::commit();
11        return response()->json(['message' => 'Pracownik usuniety']);
12    } catch (\Exception $e) {
13        DB::rollback();
14        return response()->json(['message' => 'Wystapil blad podczas
usuwania pracownika'], 500);
15    }
16 }
```

W tej transakcji:

- Rozpoczynamy transakcję.
- Znajdujemy pracownika o podanym id.
- Usuwamy rekord pracownika.
- Zatwierdzamy lub wycofujemy transakcję w zależności od wyniku operacji.

## 7. Funkcje

```
1 DELIMITER //
```

```
2
```

```
3 CREATE FUNCTION czas_przebywania_pracownika(pracownik_id INT, data
```

```
4     DATE) RETURNS TIME
```

```
5 BEGIN
```

```
6     DECLARE total_time TIME;
```

```
7     SELECT SEC_TO_TIME(SUM(TIMESTAMPDIFF(SECOND, Wejscie, Wyjscie)))
```

```
8     INTO total_time
```

```
9     FROM Obecnosc_pracownikow
```

```
10    WHERE Pracownicy_id = pracownik_id
```

```
11    AND Data = data;
```

```
12    RETURN total_time;
```

```
13 END //
```

```
14
```

```
15 DELIMITER ;
```

```
16
```

```
17
```

```
18 SELECT czas_przebywania_pracownika(7, '2024-05-27') AS
```

```
    czas_przebywania;
```

### 7.1. Działanie

- Funkcja przyjmuje dwa argumenty: `pracownik_id` (identyfikator pracownika) oraz `data` (data, dla której obliczany jest czas przebywania).
- Funkcja deklaruje zmienną `total_time`, która przechowuje całkowity czas przebywania pracownika w firmie.
- Funkcja oblicza całkowity czas przebywania pracownika w firmie w określonym dniu poprzez sumowanie różnic czasowych (w sekundach) między czasem wejścia (`Wejscie`) a czasem wyjścia (`Wyjscie`) i konwersję wyniku na format `TIME`.
- Funkcja zwraca obliczony czas przebywania pracownika.



## 7.2. Przykład użycia funkcji

Aby obliczyć czas przebywania pracownika o identyfikatorze 7 w firmie w dniu 2024-05-27, można użyć następującego zapytania:

```
1 SELECT czas_przebywania_pracownika(7, '2024-05-27') AS  
   czas_przebywania;
```

Listing 1. Przykładowe użycie funkcji `czas_przebywania_pracownika`

## 7.3. Wynik

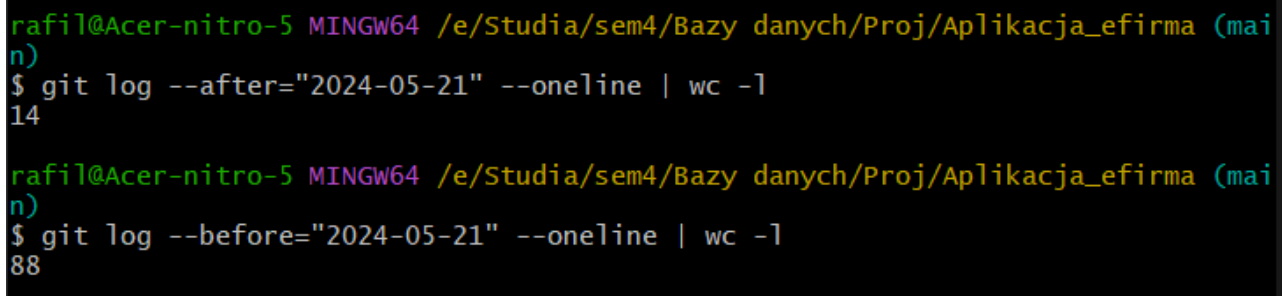
Zapytanie zwróci całkowity czas przebywania pracownika w firmie w określonym dniu w formacie TIME.

The screenshot displays a web-based SQL interface. At the top, a green status bar indicates 'Pokazano wiersze 0 - 0 (1 total, Wykonanie zapytania trwało 0,0004 sekund(y).)'. Below this, the SQL query is shown: `SELECT czas_przebywania_pracownika(7, '2024-05-27') AS czas_przebywania;`. A toolbar contains options like 'Profilowanie', 'Edytuj w linii', 'Edit', 'Explain SQL', 'Create PHP code', and 'Refresh'. Below the toolbar, there are controls for 'Show all', 'Liczba wierszy' (set to 25), and a search box labeled 'Przeszukaj tę tabelę'. An 'Extra options' button is also present. The result is shown in a table with one column, `czas_przebywania`, and one row containing the value `00:15:35`.

Rys. 7.1. Test działania funkcji

## 8. Git

### 8.1. Ilość commitów

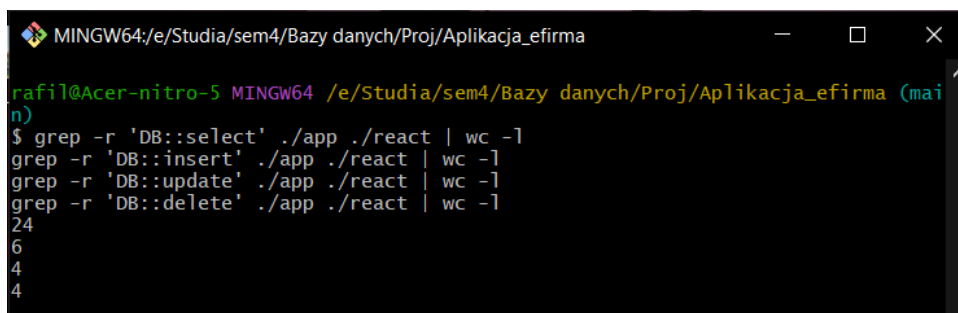


```
rafil@Acer-nitro-5 MINGW64 /e/Studia/sem4/Bazy danych/Proj/Aplikacja_efirma (main)
$ git log --after="2024-05-21" --oneline | wc -l
14

rafil@Acer-nitro-5 MINGW64 /e/Studia/sem4/Bazy danych/Proj/Aplikacja_efirma (main)
$ git log --before="2024-05-21" --oneline | wc -l
88
```

Rys. 8.1. Ilość commitów, rozdzielona datą 21.05.2024


### 8.2. Ilość zapytań Raw Query



```
MINGW64:/e/Studia/sem4/Bazy danych/Proj/Aplikacja_efirma
rafil@Acer-nitro-5 MINGW64 /e/Studia/sem4/Bazy danych/Proj/Aplikacja_efirma (main)
$ grep -r 'DB::select' ./app ./react | wc -l
24
$ grep -r 'DB::insert' ./app ./react | wc -l
6
$ grep -r 'DB::update' ./app ./react | wc -l
4
$ grep -r 'DB::delete' ./app ./react | wc -l
4
```

Rys. 8.2. Wyświetlenie ilości użyć raw Query

### 8.3. Ilość zapytań ORM



```
rafil@Acer-nitro-5 MINGW64 /e/Studia/sem4/Bazy danych/Proj/Aplikacja_efirma (main)
$ grep -r '::all' ./app ./react | wc -l
3
$ grep -r '::create' ./app ./react | wc -l
7
$ grep -r '::find' ./app ./react | wc -l
14
$ grep -r '::findOrFail' ./app ./react | wc -l
1
$ grep -r '::update' ./app ./react | wc -l
4
$ grep -r '::delete' ./app ./react | wc -l
4
```

Rys. 8.3. Wyświetlenie ilości użyć ORM

## 9. Podsumowanie projektu

W tej części dokumentacji przedstawiono szczegółowy opis funkcji, które zostały zrealizowane w ramach projektu.

### 9.1. Zrealizowane zagadnienia

#### 1. Logowanie

Implementacja mechanizmu logowania, który pozwala użytkownikom na dostęp do systemu po podaniu prawidłowych danych uwierzytelniających (loginu i hasła).

#### 2. Wylogowywanie się

Funkcjonalność umożliwiająca użytkownikom bezpieczne wylogowanie się z systemu, aby zakończyć sesję użytkownika.

#### 3. Tworzenie tokenu użytkownika

Mechanizm generowania tokenów uwierzytelniających, które są używane do autoryzacji i uwierzytelniania użytkowników w systemie.

#### 4. Użytkownik podaje własne hasło po wpisaniu hasła tymczasowego, które powstało po założeniu konta użytkownika przez admina

Proces, w którym nowy użytkownik po zalogowaniu się za pomocą hasła tymczasowego jest zobowiązany do ustawienia własnego, trwałego hasła.

#### 5. Zaszycrowanie haseł oraz ich hashowanie

Implementacja algorytmów kryptograficznych do szyfrowania i hashowania haseł w celu zabezpieczenia danych użytkowników przed nieautoryzowanym dostępem.

#### 6. Stworzenie stron dla:

##### (a) Niezalogowanego użytkownika:

- Wyświetlanie aktualności pobranych z bazy  
Strona główna dla niezalogowanych użytkowników wyświetlająca najnowsze wiadomości i aktualności z bazy danych.
- Wyświetlanie informacji o firmie i kontakt  
Strona zawierająca informacje o firmie, jej misji, wizji, oraz dane kontaktowe dla potencjalnych klientów i partnerów.

##### (b) Zalogowanego pracownika:

- Panel główny z możliwością zgłaszania obecności/końca pracy  
Panel, który umożliwia pracownikom rejestrowanie godzin rozpoczęcia i zakończenia pracy.
- Ogłoszenia przypisane do danej grupy pracowników  
Sekcja z ogłoszeniami skierowanymi do konkretnej grupy pracowników, zawierająca ważne informacje i aktualności.
- Wyświetlanie informacji o karcie dostępu  
Strona pokazująca szczegóły dotyczące karty dostępu pracownika, w tym status i historię użycia.
- Skaner kodów QR znajdujących się na drzwiach  
Narzędzie umożliwiające pracownikom skanowanie kodów QR, aby uzyskać dostęp do różnych pomieszczeń w budynku.
- Informacje o miejscach zamieszkania pracownika  
Sekcja zawierająca dane adresowe pracowników, co może być przydatne dla celów administracyjnych i logistycznych.

(c) Administratora:

- Administrator posiada panel pracownika z guzikiem zmiany na panel administratora  
Administratorzy mają dostęp do swojego panelu, który zawiera również funkcje dostępne dla pracowników, z możliwością przełączania się między trybem pracownika a administratorem.
- Lista pracowników wraz z możliwością edycji pracownika i możliwości zablokowania jego kart w przypadku zablokowania pracownika (CRUD pracownicy)  
Funkcje zarządzania danymi pracowników, w tym tworzenie, odczyt, aktualizacja i usuwanie (CRUD), oraz blokowanie kart dostępu w razie potrzeby.
- Rejestracja nowego użytkownika z automatycznym hasłem tymczasowym i wysyłaniem go na maila  
Proces rejestracji nowych użytkowników, w tym generowanie i wysyłanie tymczasowych haseł na adres e-mail nowego użytkownika.
- CRUD karty dostępu  
Zarządzanie kartami dostępu, w tym tworzenie, odczyt, aktualizacja i usuwanie danych kart.
- CRUD drzwi, drukowanie ich kodów QR i edycja  
Zarządzanie informacjami o drzwiach, w tym tworzenie, edytowanie, usuwanie oraz drukowanie kodów QR przypisanych do drzwi.
- CRUD stref dostępu

Zarządzanie strefami dostępu, w tym dodawanie, edytowanie, usuwanie stref i przypisywanie do nich uprawnień dostępu.

- CRUD budynków

Zarządzanie danymi budynków, w tym tworzenie, edytowanie i usuwanie informacji o budynkach.

- CRUD ogłoszenia

Zarządzanie ogłoszeniami, w tym dodawanie, edytowanie i usuwanie ogłoszeń dla pracowników.

- Generowanie raportów: przepracowanych godzin, ilości prób dostępu

Narzędzia do generowania różnych raportów, takich jak raporty dotyczące przepracowanych godzin przez pracowników oraz raporty dotyczące ilości prób dostępu do różnych stref.

## 9.2. Niewykonane zagadnienia

1. Token co 60s

Funkcjonalność generowania nowego tokenu co 60 sekund, co mogłoby zwiększyć bezpieczeństwo sesji użytkownika, nie została zrealizowana.

2. Branie urlopów

Mechanizm umożliwiający pracownikom składanie wniosków urlopowych oraz zarządzanie nimi nie został wdrożony.

3. Wybór wykorzystania nadgodzin

Funkcjonalność pozwalająca pracownikom na wybór sposobu wykorzystania nadgodzin (np. jako dodatkowy czas wolny lub dodatkowe wynagrodzenie) nie została zaimplementowana.

## Bibliografia

- [1] *Wikipedia o laravel*. URL: <https://en.wikipedia.org/wiki/Laravel> (term. wiz. 09.03.2024).
- [2] *Wikipedia o react*. URL: [https://en.wikipedia.org/wiki/React\\_\(software\)](https://en.wikipedia.org/wiki/React_(software)) (term. wiz. 09.03.2024).
- [3] *Wikipedia o MySQL*. URL: <https://pl.wikipedia.org/wiki/MySQL> (term. wiz. 09.03.2024).

## Spis rysunków

4.1. Diagram przypadków użycia Administratora dla aplikacji E-firma . . . . .	7
4.2. Diagram przypadków użycia Pracownika dla aplikacji E-firma . . . . .	8
4.3. Diagram przypadków użycia Gościa dla aplikacji E-firma . . . . .	8
5.1. Działające triggerzy . . . . .	19
7.1. Test działania funkcji . . . . .	25
8.1. Ilość commitów, rozdzielona datą 21.05.2024 . . . . .	26
8.2. Wyświetlenie ilości użyć raw Query . . . . .	26
8.3. Wyświetlenie ilości użyć ORM . . . . .	26

## Spis listingów

1. Przykładowe użycie funkcji `czas_przebywania_pracownika` . . . . . 25