# **PROJEKT**

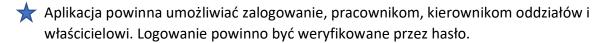
## Michał Kalina, Igor Cichecki

### PODSTAWOWE INFORMACJE

Tematem projektu jest aplikacja bazodanowa dla firmy PEPET – sieci fastfoodów działających na terenie Polski, podzielonej na oddziały zarządzane przez kierowników.

### WYMAGANIA OPISANE PR7E7 KLIENTA

🛖 Baza powinna przechowywać dane pracowników, pełnione przez nich stanowiska, i wartości pensji jakie należy wypłacić pracownikom.



racownik powinien mieć możliwość sprawdzenia informacji o sobie, powinien móc sprawdzić swoje imię, nazwisko, pensję i pełnione w firmie stanowisko.

🛖 Każde stanowisko ma przyporządkowaną pensję minimalną których w żadnym razie nie możemy przekroczyć.

racownik nie może uzyskać dostępu do prywatnych informacji pozostałych pracowników.



★ Kierownik oddziału powinien móc:

- Modyfikować pensje podopiecznych.
- Dodawać nowych pracowników do bazy, usuwać istniejących.
- Kierownik oddziału nie może wpływać na informacje o pracownikach zawartych w innych oddziałach.

🛖 Właściciel firmy nie życzy sobie żadnych ograniczeń. Może "dowolnie" modyfikować przechowywane w bazie dane.

rowinno być możliwe wykonanie kopii przechowywanych w bazie danych, i wczytanie ich w wypadku awarii.

# DOOKREŚI ENIE WYMAGAŃ

- Baza danych powinna mieć trzy poziomy dostępu: pracownik, kierownik, właściciel.
- Możliwość wyświetlenia przez pracownika jego prywatnych danych.
- Możliwość modyfikowania danych przez kierownika, wprowadzone, przez kierownika dane muszą być kontrolowane.
- Hasła pracowników i kierowników muszą być przechowywane dość bezpiecznie ( niestety w naszym wypadku oznacza to zwykłe haszowanie)
- Możliwość dokonywania backapu danych, jedynie przez właściciela, nasz zleceniodawca nie życzy sobie żadnych administratorów którym trzeba by dodatkowo płacić!
- Zachowanie spójności danych, pracownicy nie mogą wyjść poza widełki pensji przyporządkowanych ich stanowisku, nie wolno zatrudniać niepełnoletnich, pracownik nie może pełnić stanowiska nie istniejącego w bazie...

### KILKA UWAG

- Tworzymy aplikację bazodanową, przy czym baza powinna być zaprojektowana tak, by nie było problemów z podłączeniem do niej różnych GUI.
- Za logikę aplikacji powinna odpowiadać gdy tylko to możliwe baza.
- Spójność danych wprowadzanych przez użytkowników kontrolujemy przez trigery opisane na diagramie UML i klucze obce.
- Stosujemy model OLTP przewidujemy częste modyfikacje tabelek z informacjami o pracownikach, masowe zwolnienia są dzisiaj na porządku dziennym.
- Staramy się zachować trzecią postać normalną przez stosowanie wielu sztucznych kluczy ( id ).
- Do uniknięcia niebezpieczeństw związanych z masowymi podwyżkami stosujemy transakcje.
- Staramy się obronić przed dependency injection przez stosowanie prepare statment.

# STRUKTURA BAZY DANYCH

TABELKI: (Atrybuty można znaleźć na diagramie UML, pogrubiamy klucze)

- pracownicy
- dane\_pracownikow
- stanowiska
- oddziały
- kierownicy
- logowanie\_pracownikow ( nazwa robocza )
- logowanie\_kierownikow (nazwa robocza )

# TRIGERY -> DiagramUML na dole tak powinno być je widać troszkę czytelniej

# KLUCZE OBCE:

TABELA A	POLE	TABELA B	POLE
pracownicy	id_stanowiska	stanowiska	Id
pracownicy	id_oddziału	oddziały	Id
kierownicy	id_oddziału	oddziały	Id
logowanie_pracownikow	id_pracownika	pracownicy	Id
logowanie_kierownikow	id_kierownika	kierownicy	Id

Klucze nie pozwalają dodawać do tabelek A wartości nie istniejących w B.

Powinny pomóc nam uniknąć istnienia w bazie pracowników o nieistniejących stanowiskach i podobnych nieprzyjemnych sytuacji.

#### INDEXY:

pracownicy:
pensja (BTREE)
etat (BTREE)
danePracowników:
PESEL (BTREE)

Oraz indexy zapewnione nam przez planowane klucze

### DODATKOWE INFORMACJE

### Planowane Funkcje/Procedury:

- bonus(Nazwa działu, procent) powinna wykonywać się tranzakcyjnie
- modyfikacja(id pracownika, kwota) podwyżka, obniżka pensji
- modyfinkcja\_kierownika(id\_kierownika, kwota) podwyżka, obniżka pensji
- ..

# **DIAGRAM UML**

