CompanyMenager

1. Cel aplikacji:

   Aplikacja "CompanyMenager" ma na celu przechowywanie oraz zarządzanie informacjami o pracownikach danej firmy z różnych poziomów dostępu w oparciu o relacyjną bazę danych MySQL.

2. Wymagania projektu:

   Aplikacja powinna składać się z dedykowanej bazy danych MySQL oraz GUI stworzonego w języku Java. Aplikacja bedzie implementować również 4-poziomowy dostęp do danych skutkujące blokowaniem poszczególnych funkcji aplikacji.

3. Funkcjonalności aplikacji:
   - pozwala rozszeżać listę danych osobowych o kolejne rekordy.
   - umożliwia rekrutacje oraz zwolnienie pracowników spośród osób znajdujących sie w bazie danych.
   - umożliwia prawidłowe przydzielanie urlopu pracownikom.
   -powala na zmianę pensji danego pracownika i przechowuje jej hisorie .
   -przeprowadza tranzakcje wypłaty pensji grupom pracowniczym, gupowanymi zawaodami oraz zapisuje historie tranzakcji .
   - Zarządzaie portfelem firmy: księgowanie wypłat i wpłat.

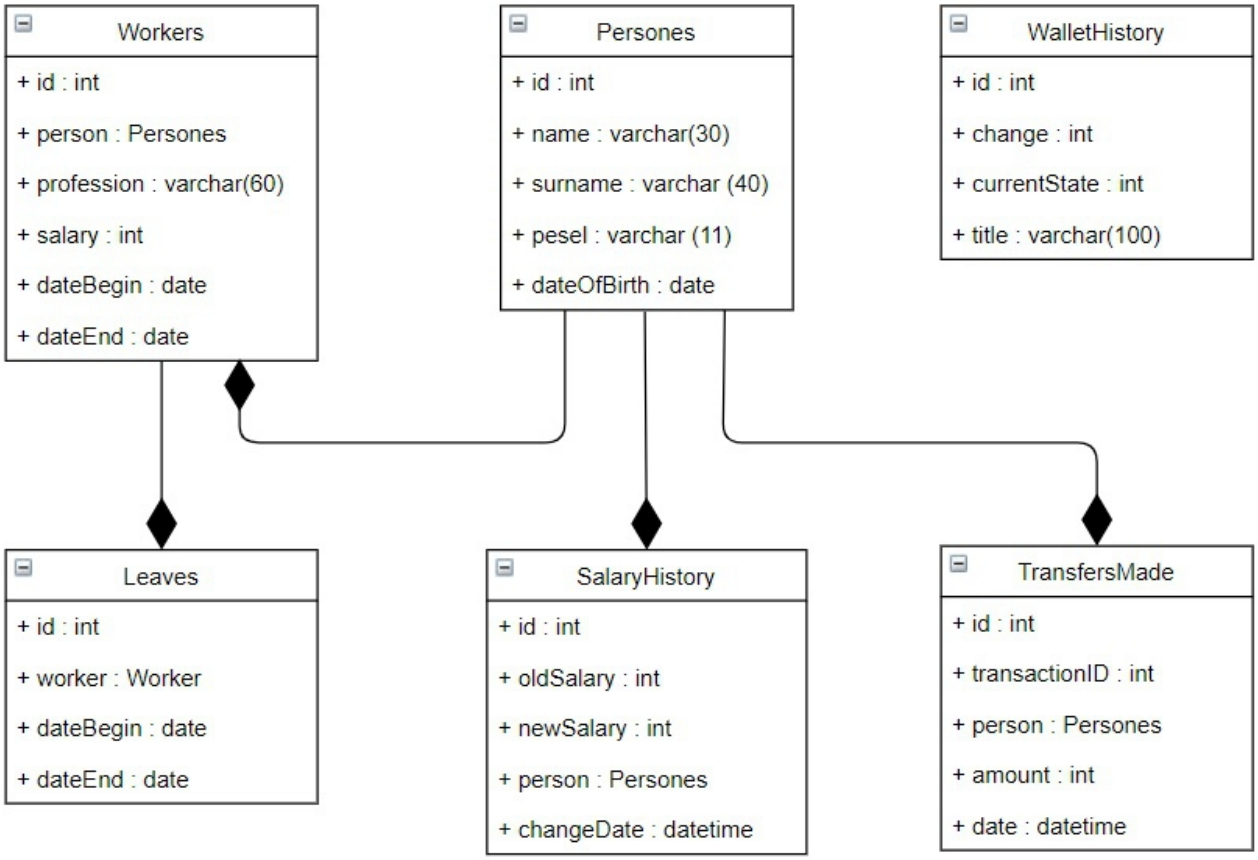
4. Prezentacja encji:
   - ➔ **Persons** – Tabela przechowująca dane osobowe osób w bazie firmy.
     Pola: id, name, surname, pesel, dateOfBirth
   - ➔ **Workers** – Tabela przechowująca informacje o aktualnych ibyłych pracownikach.
     Pola: id, **person**, profession, salary, dateBegin, dateEnd
   - ➔ **Leaves** – Tabela przechowująca informacje o odbytym urlopie pracownika.
     Pola: id, **worker**, dateBegin, dateEnd,
   - ➔ **SalaryHistory** – Tabela przechowująca informacje o zmianie pensji pracownika.
     Pola: id, newSalary, oldSalary, **person**, changeDate
   - ➔ **TransfersMade** – Tabela przechowująca historię dokonanych przelewów.
     Pola: id, transactionID, **person**, amount
   - ➔ **WalletHistory –** Tabela przechowuję historię zmian w portfelu firmy.
     Pola: id, change, currentState, title

5. Planowane elementy:
   - ➔ Procedury:
     - makeTransfer(varchar(60) profession)
     - changeSalary(int workerID, int newSalary)
     - addPerson(varchar(30) name, varchar(40) surname, varchar (11) pesel ,date dateOfBirth)
     - recruitEmploye(int workerID, varchar(60) profession, int salary, date dateBegin, date dateEnd)
     - releaseEmploye(int employeId)
     - giveLeave(int employeId, date dateBegin, date dateEnd)
     - walletOperation(int change, varchar(100) title)

➔ Funkcje:
- takeNewTransactionId();

➔ Trigery:
- Persons :
  - ✗ Poprawność pesel oraz zgodność z datą urodzenia.
  - ✗ Imię oraz nazwisko złożeone z liter (bez cyfr i znaków niebędących literami)
  - ✗ Data urodzenia nie późniejsza niż aktualna.
- TransfersMade :
  - ✗ Kwota dodatnia.
- SalaryHistory :
  - ✗ Nowa pensja dodatnia.
- Workers :
  - ✗ Osoba musi być pełnoletnia.
  - ✗ Pensja nieujemna.
  - ✗ Data zakończenia nie może być wcześniejsza niż data rozpoczęcia.
  - ✗ Róznica między Datą rozpoczęcia pracy a datą urodzenia nie może być mmniejsza niż 18 lat.
- Leaves :
  - ✗ Ograniczenie udzielania urlopu, zgodnie z aktualnymi zasadami.
    Ilość dni urlopu zwiakszana kazdego przepracowanego miesiąca.
    W skali roku 26 dni.
  - ✗ Data końca nie może być wcześniejsza niż data rozpoczęcia.
- WalletHistory :
  - ✗ Nie można wykonać operacji podwodującej zejście stanu konta poniżej 0.

6. UML :



7. Poziomy dostępów:
   - ➔ Boss
     - changeSalary
     - addPerson
     - recruitEmploye
     - releaseEmploye
     - giveLeave
     - makeTransfer
     - walletOperation
   - ➔ Menager
     - changeSalary
     - addPerson
     - recruitEmploye
     - releaseEmploye
     - giveLeave
   - ➔ Recruiter
     - addPerson
     - recruitEmploye
   - ➔ Accountant
     - makeTransfer
     - walletOperation

Projekt przygotowuje :

Szafraniak Krzysztof