Programowanie obiektowe i graficzne
projekt grupowy
“LogisticApp”

Mateusz Tytoń, Nikodem Lewandowicz
Sem. IV - Informatyka niestacjonarna RMS
12.07.2021

1. Problem do rozwiązania

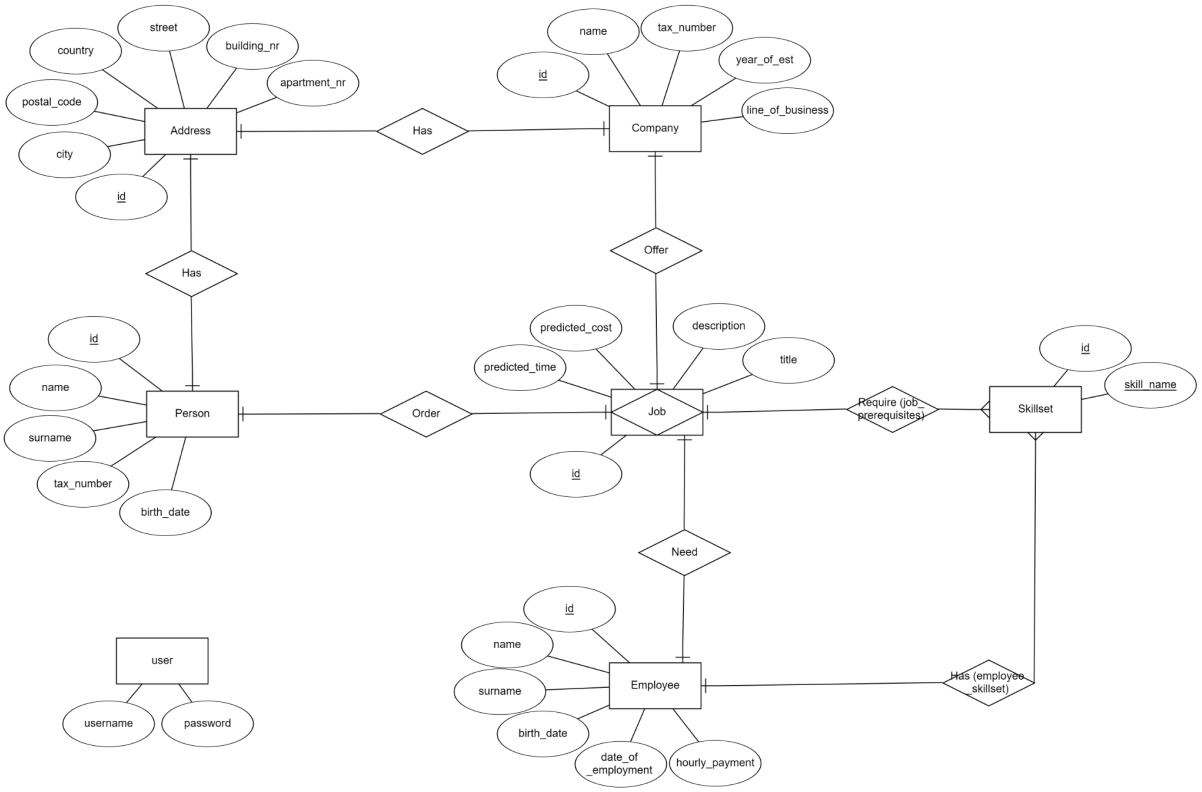
Aplikacja rozwiązuje problem zarządzania klientami, pracownikami oraz zleceniami w danej firmie. Aplikacja ma pomóc w tworzeniu instancji i wiązaniu ich ze sobą. Jest to prosty system do zarządzania, do wykorzystania przez dział HR.
Aplikacja posiada możliwość rozbudowy o liczne funkcjonalności w zależności od potrzeb konkretnej firmy.

2. Założenia

Projekt w założeniu ma ułatwić funkcjonowanie firmy poprzez dostarczenie narzędzia do wewnętrznego zarządzania danymi. Projekt ma pozwolić na:

- Dodanie rekordu firmy klienckiej
- Dodanie rekordu klienta jako osoby fizycznej
- Dodanie rekordu pracownika
- Dodanie rekordu zlecenia
- Aktualizację rekordu firmy klienckiej
- Aktualizację rekordu klienta jako osoby fizycznej
- Aktualizację rekordu pracownika
- Aktualizację rekordu zlecenia
- Usunięcie rekordu firmy klienckiej
- Usunięcie rekordu klienta jako osoby fizycznej
- Usunięcie rekordu pracownika
- Usunięcie rekordu zlecenia
- Przeglądanie danych

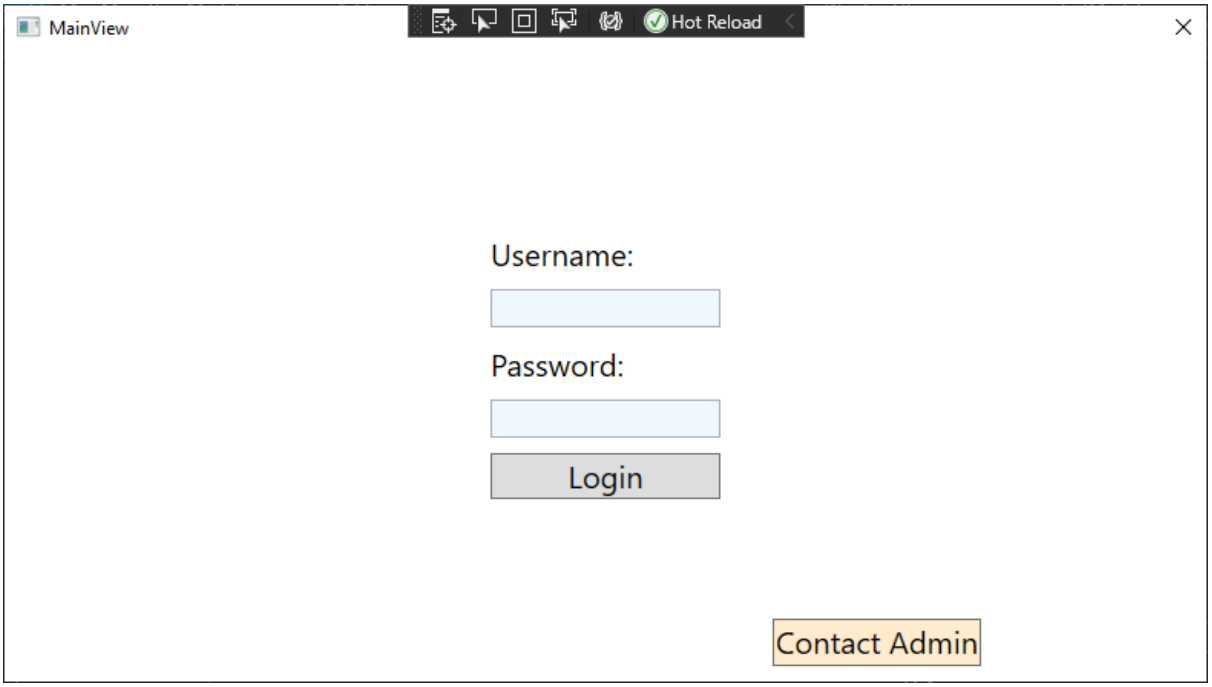
3. Baza danych



4. Interfejs graficzny

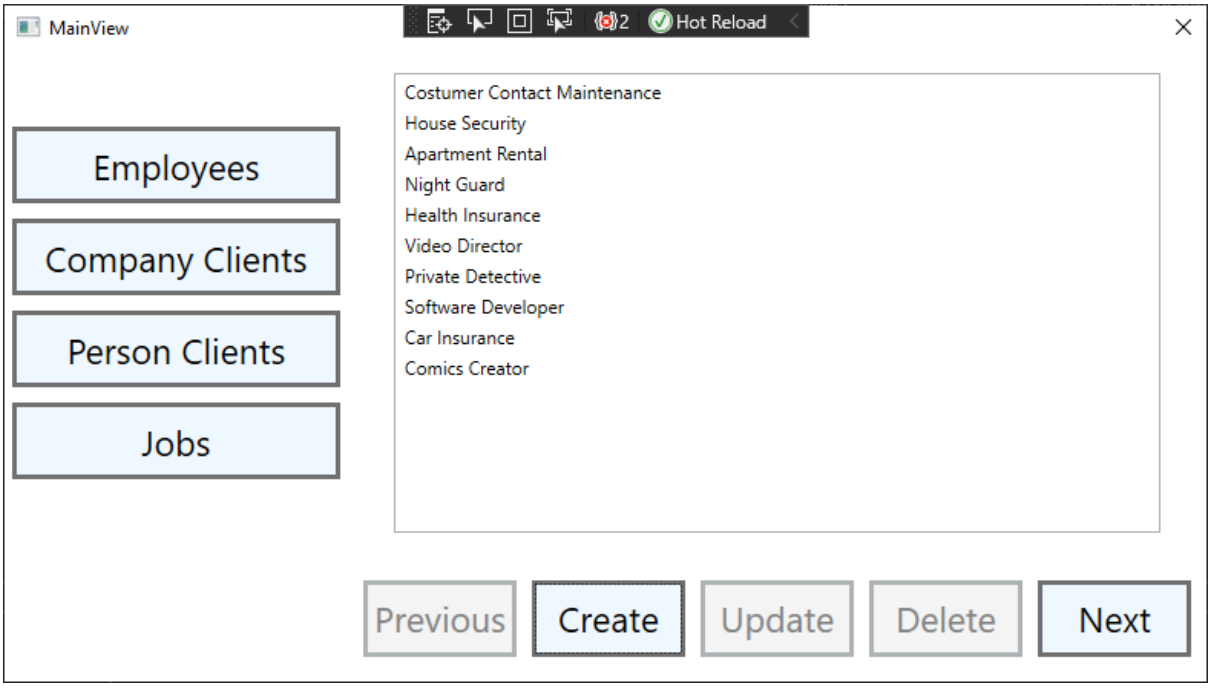
Ekrany aplikacji:

- Ekran Logowania:



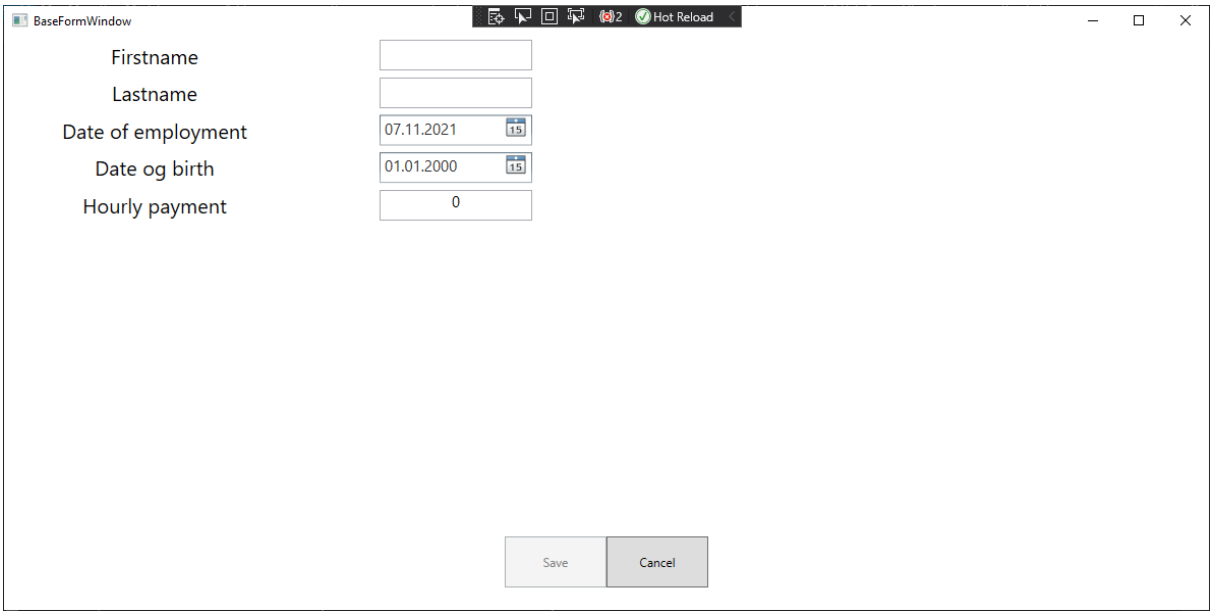
Na ekranie logowania podajemy nazwę użytkownika i hasło, następnie klikamy login, ekran ten służy jedynie do autoryzacji użytkownika.

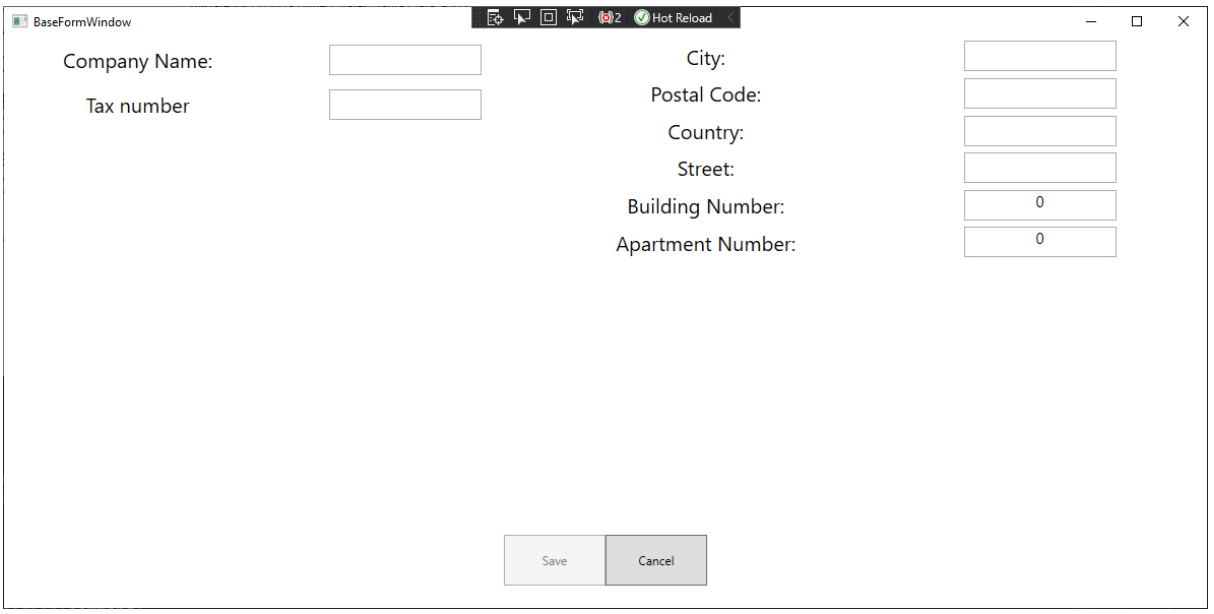
- Ekran przeglądania danych:

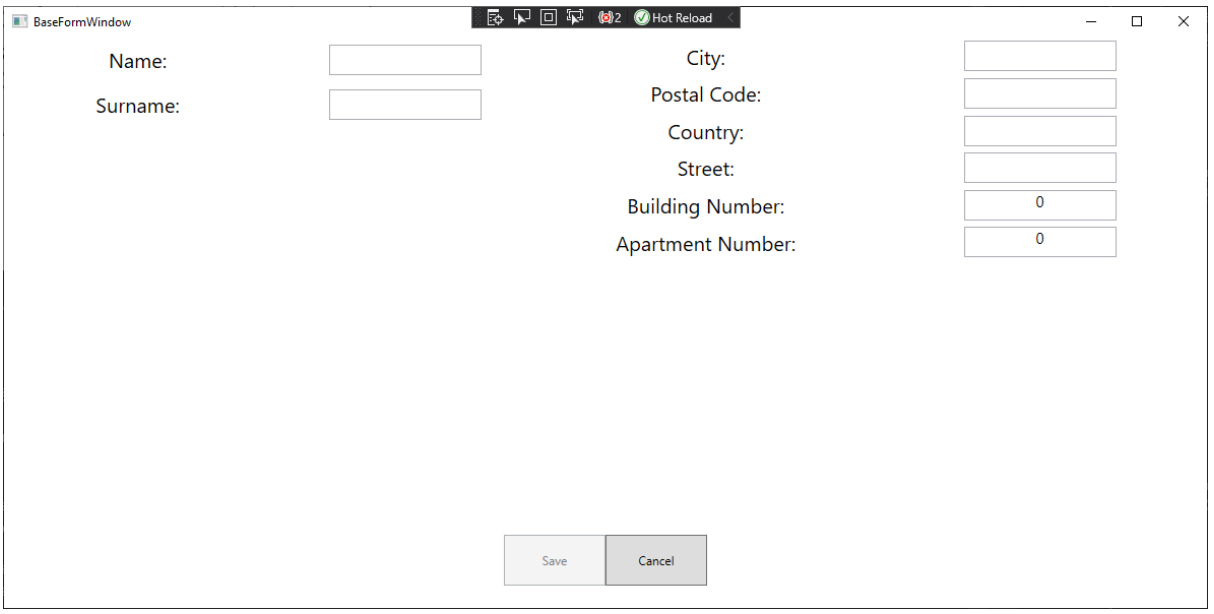


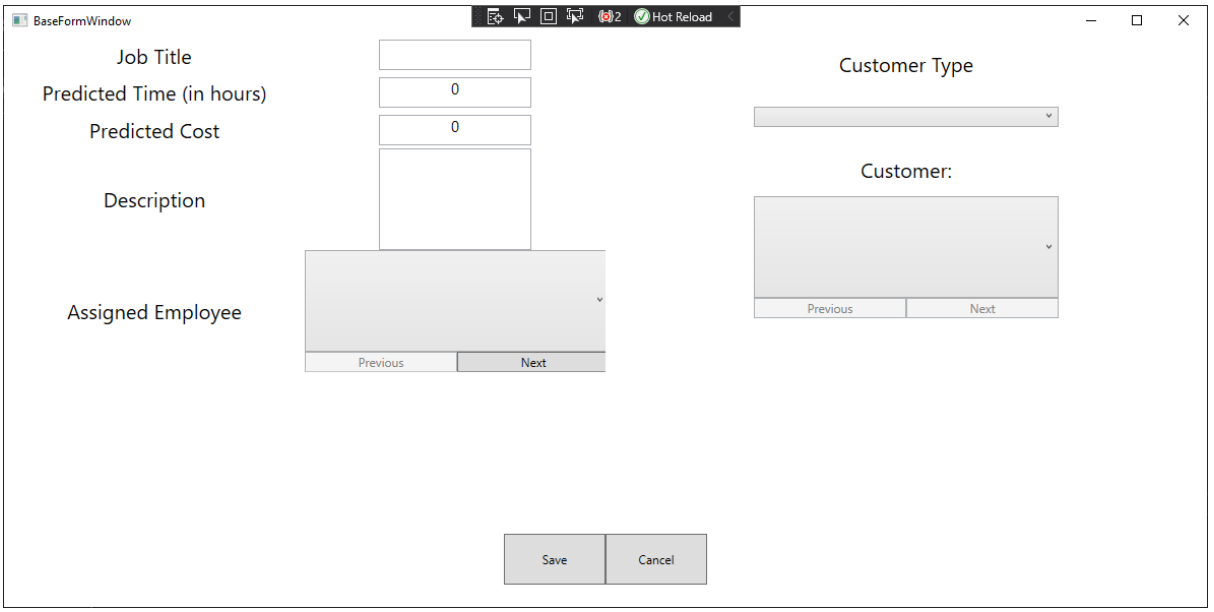
Na ekranie przeglądania danych mamy możliwość wybrać przy pomocy przycisków z boku interesujące nas dane do przeglądania. Na dole znajdują się przyciski do konkretnych operacji:
Previous - cofa stronę danych wcześniej
Create - otwiera formularz dla tworzenia rekordu (rekord z aktualnie wybranego modelu)
Update - otwiera formularz aktualizacji aktualnie zaznaczonego rekordu
Delete - usuwa zaznaczony rekord
Next - przechodzi stronę danych dalej

- Formularz tworzenia/aktualizowania rekordu









Wszystkie formularze wyglądają podobne, ich zawartość jest zależna od

5. Charakterystyka modelu

Database Access Layer:

Encje - są reprezentacją encji z bazy danych w programie, powiązania pomiędzy encjami reprezentowane są jako obiekty tych encji w encji nadrzędnej
(korzystamy tutaj ze składania klas). Encje wykorzystane w programie:

- ➢ Address - adres firmy lub klienta jako osoby fizycznej
- ➢ Company - firma kliencka
- ➢ Person - klient jako osoba fizyczna
- ➢ Employee - pracownik, zatrudniony w firmie do której należy program
- ➢ Job - zlecenie, składane przez firmę/osobę fizyczną z możliwością przypisania pracownika
- ➢ User- użytkownik programu

DAO (Data Access Object) - obiekty dostępu do danych, W programie nie implementowano wzorca Repozytorium ze względu na to, że część operacji wykonywanych przez ten wzorzec nie jest wykorzystywana w programie. Z tego względu zdecydowano się zaimplementować zwykły Data Access Object. Każda encja posiada odpowiadający jej Data Access Object.

DatabaseConnection - klasa odpowiedzialna za połączenie z bazą danych. Klasa jest zaimplementowana jako Singleton, ponieważ na czas działania programu nie potrzebujemy więcej niż jednej instancji połączenia do bazy.

Model:

Pagination:

BasePagination - interfejs, który deklaruje podstawowe metody, które powinny być zaimplementowane przez klasę paginatora.

OffsetPagination - klasa będąca konkretną implementacją paginatora, w razie potrzeby można zmienić sposób paginacji, klasa implementuje interfejs BasePagination.

Utils:

WindowObserver - klasa będąca implementacją wzorca obserwator i singleton. Klasa ta potrzebna jest w warstwie ViewModel do komunikowania się pomiędzy oknami, gdy okno tworzenia się zamknie musi o tym dać znać oknu, w którym wyświetlane są dane.

AuthModel - klasa, której zadaniem jest autentykacja użytkownika, sprawdza ona czy poświadczenia są poprawne.

Creator - klasa służąca do tworzenia/aktualizowania rekordów, korzysta ze zjawiska poliformizmu.

ListModel - model do obsługi widoku listy

DataAccessFacade - wzorzec projektowy fasady, ułatwiający dostęp do obiektów DAO. wystarczy wywołać operację i podać nazwę modelu.

Session - model zajmujący się obsługą sesji, klasa zaimplementowana jako Singleton ponieważ, na jedną instancję programu chcemy mieć tylko jedną sesję.