Programowanie obiektowe i graficzne

Informatyka semestr IV

Dokumentacja projektu bazy danych: *System Rekrutacyjny*

Członkowie zespołu:

Mariusz Mańka

Tymoteusz Małkowski

Marcel Michalik

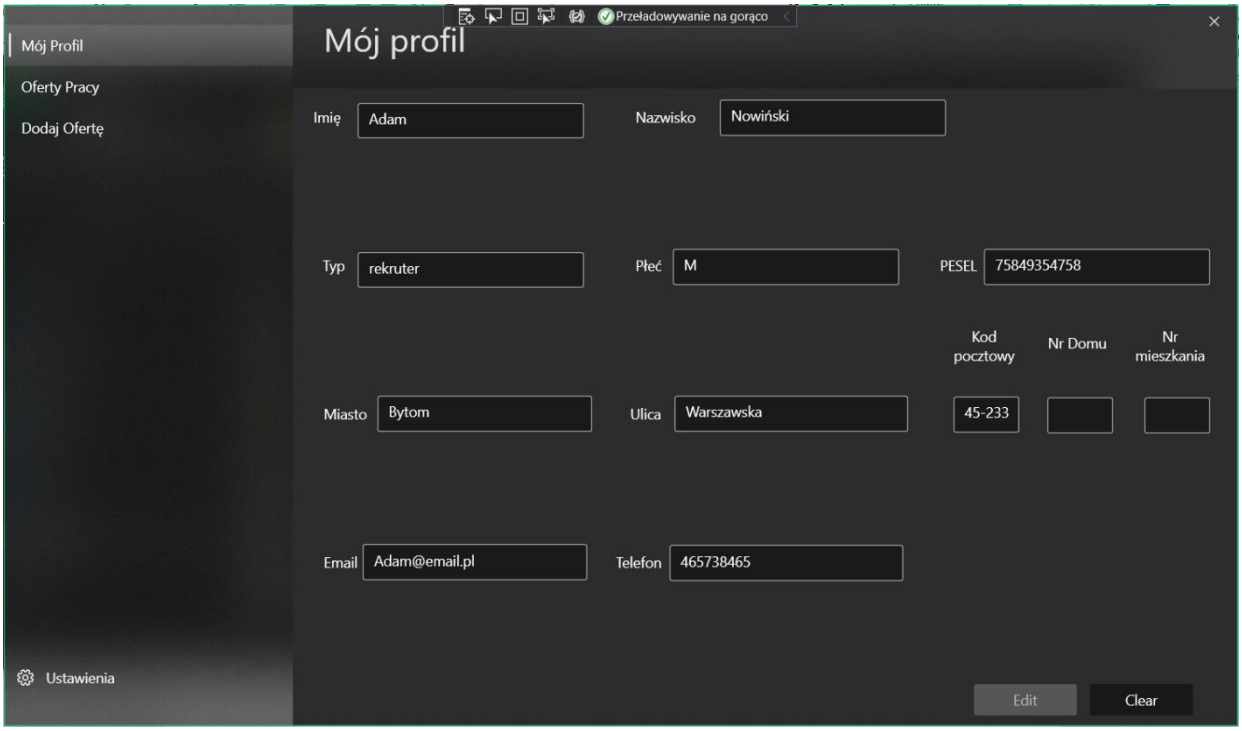
18 września 2021

1. Założenia i cele projektu.

Celem projektu było stworzenie systemu rekrutacyjnego podłączonego do bazy danych mysql. Projekt został wykonany w języku c# za pomocą wzorca *MVVM*. System rekrutacyjny jest przeznaczony dla rekruterów poszukujących odpowiednich kandydatów na stanowisko dla swoich pracodawców. Aplikacja jest również przeznaczona dla osób poszukujących pracy, które mogą przeglądać dostępne oferty pracy w których znajdują się konkretnie opisane wymagania co do stanowiska pracy.

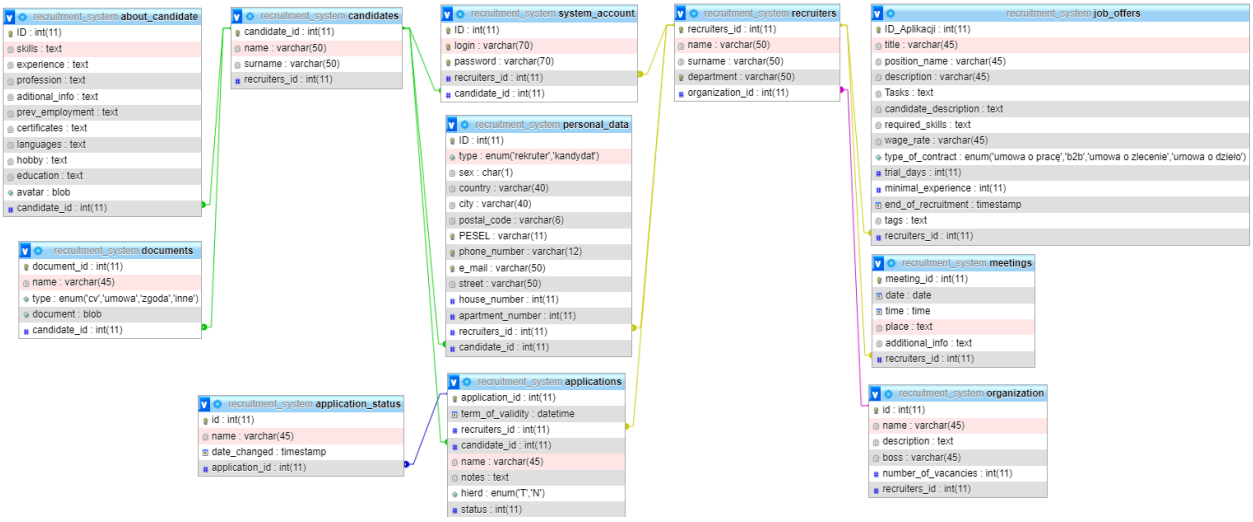
Podstawowe założenia funkcjonalności projektu:

- Przeglądanie i aktualizowanie przez użytkownika danych personalnych w zakładce *Mój profil*.
- Możliwość przeglądania ofert pracy przez użytkowników w zakładce *Oferty Pracy*.
- Dodawanie nowych ofert pracy w zakładce *Dodaj ofertę*.



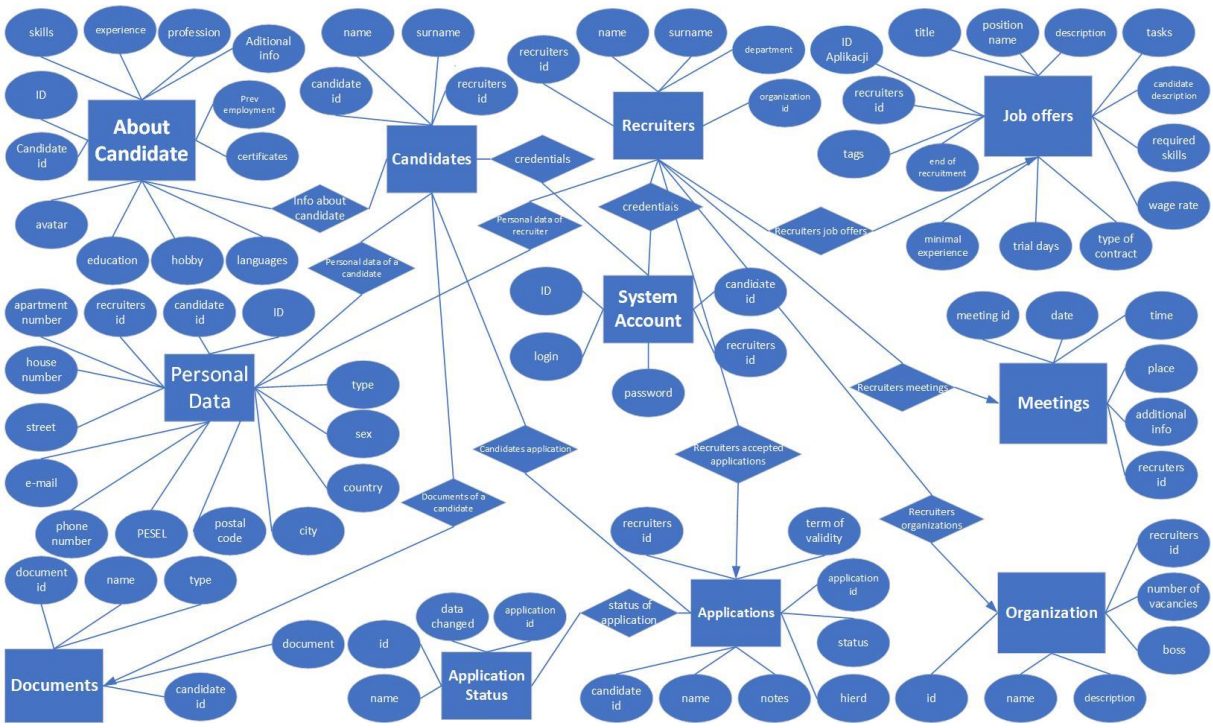
Zdj.1 *Zakładka Mój profil*

2. Model relacyjny Bazy Danych.



Zdj.2 Model relacyjny bazy danych.

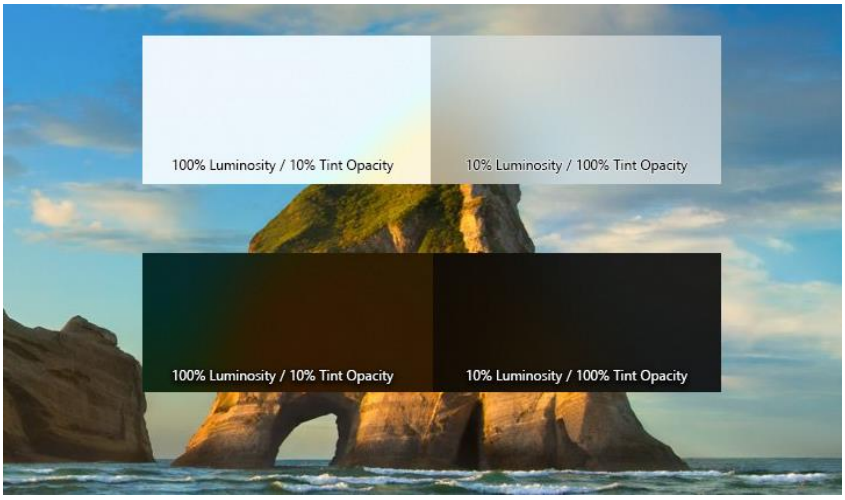
3. Diagram związków encji.



Zdj.3 Diagram związków encji.

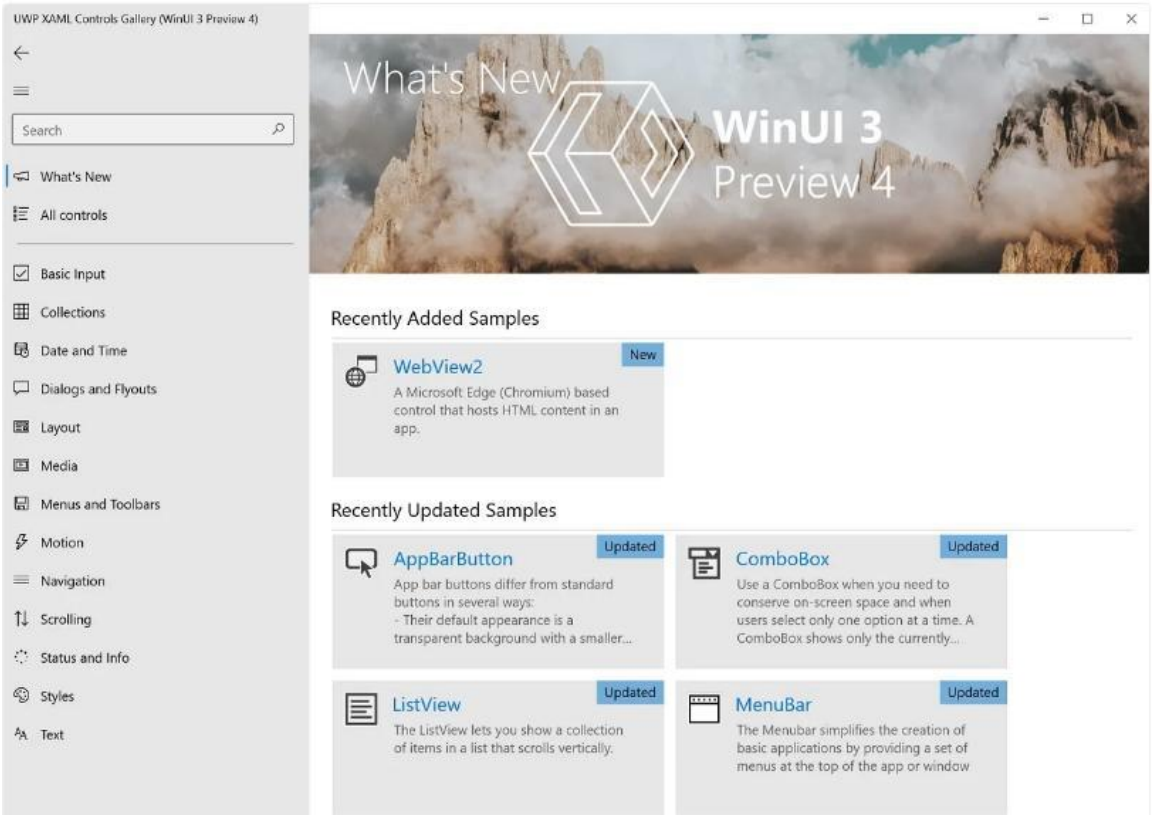
4. Opis interfejsu graficznego.

Do interfejsu graficznego aplikacji zostały wykorzystane dwie biblioteki, *FluentWPF* oraz *ModernWPFUI*. Ze względu na to, iż aktualnie głównym rozwijanym przez Microsoft środowiskiem interfejsu graficznego użytkownika jest środowisko *UWP*, w środowisku *WPF* brakuje wielu elementów oraz funkcji specjalnie zaprojektowanych z myślą o nowszych wersjach systemu Windows. W przypadku aplikacji opartych o środowisko *WPF*, dodanie przeźroczystości w aplikacji, która by spełniała wyznaczone przez Microsoft wytyczne w kwestii wyglądu interfejsu graficznego, wymaga użycia dodatkowych bibliotek importujących nowe funkcje dostępne dla środowiska *UWP*. Dzięki bibliotece *FluentWPF*, aplikacja otrzymała przeźroczystość całej aplikacji z efektem rozmycia tła oraz szumem predefiniowanym przez wytyczne Microsoft. Efekt ten jest nazywany *akrylem*.



Zdj. 4&5 Przykład akrylowego efektu na tłach.

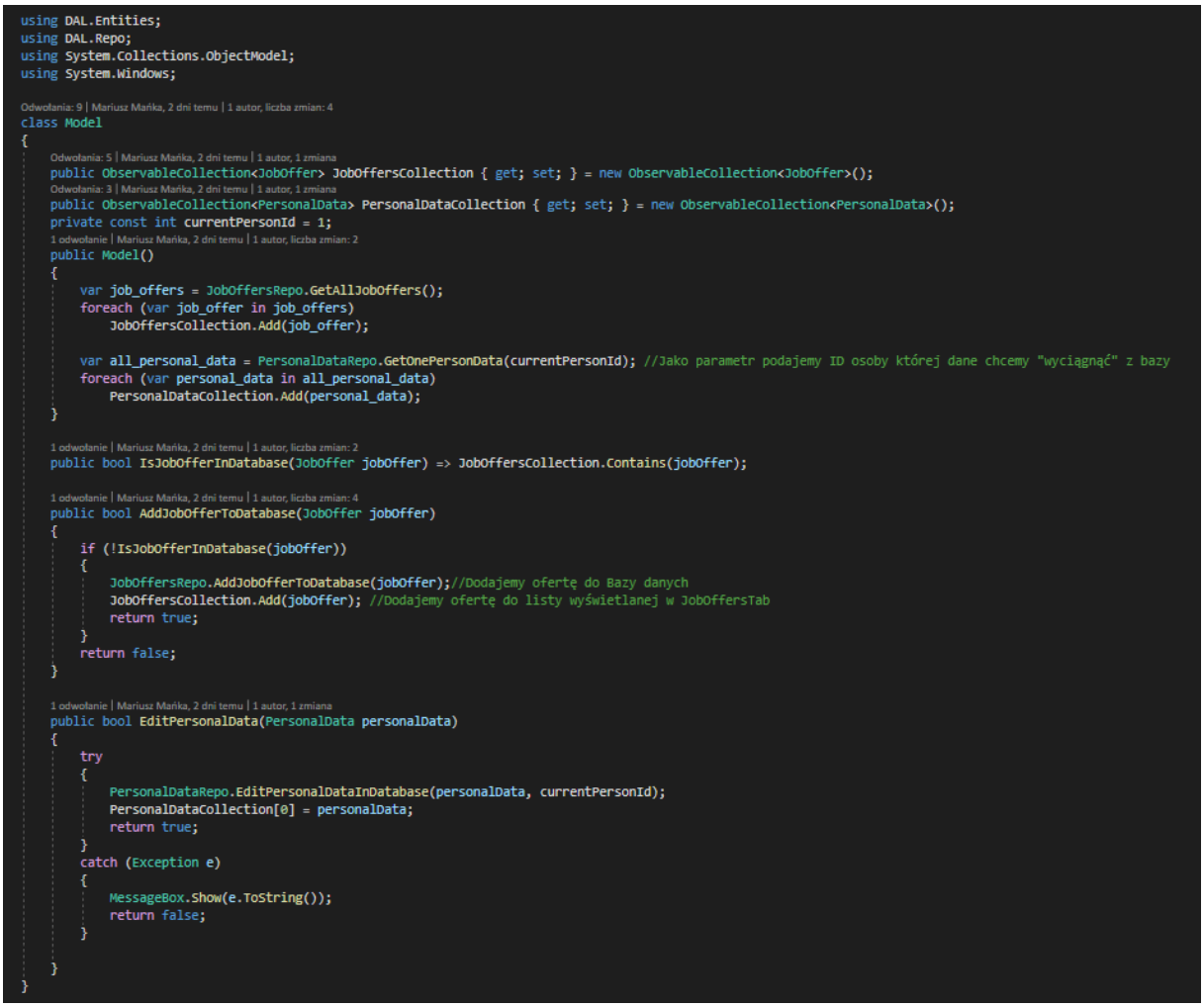
Kolejnym modułem ułatwiającym stworzenie aplikacji spełniającej aktualne ogólnie przyjęte standardy w kwestii interfejsów graficznych jest *ModernWPFUI*. Dzięki tej bibliotece deweloper uzyskuje dostęp do większości elementów *WinUI 3* oraz *WinUI 2* dostępnych w środowisku *UWP* przy zachowaniu ich łatwości w implementacji. Łatwość implementacji interfejsu sięga również w tym przypadku motywów aplikacji (również zależnych od motywu systemowego). Po uprzednim zapoznaniu się z biblioteką implementacja elementów „przeportowanych" dzięki tej bibliotece jest nie wiele cięższa od domyślnego sposobu w przypadku aplikacji opartych na środowisku *UWP*.



Zdj. 6 Przykład aplikacji demonstrującej elementy WinUI 3

5. Charakterystyka modelu obiektowego.

Projekt zgodnie z wzorcem *MVVM* został podzielony na 3 warstwy: Model, ViewModel oraz View. Dodatkowo została wykonana warstwa dostępu do bazy danych: *Database Access Layer*, która odpowiada za komunikację z baza danych. W warstwie *DAL* znajdują się odwzorowane klasy modelu bazy danych: *JobOffer*, *PersonalData* wraz z metodami tworzącymi zapytania do bazy takie jak *INSERT*, *SELECT* czy *UPDATE*. W warstwie *ViewModel* znajdują się 3 klasy odwzorowujące każdą z zakładek oraz główna klasa View Modelu – *MainViewModel*. Warstwa modelu zawiera klasę *Model*, której implementacje można zobaczyć na poniższym zdjęciu:



```
using DAL.Entities;
using DAL.Repo;
using System.Collections.ObjectModel;
using System.Windows;

Odwołania: 9 | Mariusz Mańka, 2 dni temu | 1 autor, liczba zmian: 4
class Model
{
    Odwołania: 5 | Mariusz Mańka, 2 dni temu | 1 autor, 1 zmiana
    public ObservableCollection<JobOffer> JobOffersCollection { get; set; } = new ObservableCollection<JobOffer>();
    Odwołania: 3 | Mariusz Mańka, 2 dni temu | 1 autor, 1 zmiana
    public ObservableCollection<PersonalData> PersonalDataCollection { get; set; } = new ObservableCollection<PersonalData>();
    private const int currentPersonId = 1;
    1 odwołanie | Mariusz Mańka, 2 dni temu | 1 autor, liczba zmian: 2
    public Model()
    {
        var job_offers = JobOffersRepo.GetAllJobOffers();
        foreach (var job_offer in job_offers)
            JobOffersCollection.Add(job_offer);

        var all_personal_data = PersonalDataRepo.GetOnePersonData(currentPersonId); //Jako parametr podajemy ID osoby której dane chcemy "wyciągnąć" z bazy
        foreach (var personal_data in all_personal_data)
            PersonalDataCollection.Add(personal_data);
    }

    1 odwołanie | Mariusz Mańka, 2 dni temu | 1 autor, liczba zmian: 2
    public bool IsJobOfferInDatabase(JobOffer jobOffer) => JobOffersCollection.Contains(jobOffer);

    1 odwołanie | Mariusz Mańka, 2 dni temu | 1 autor, liczba zmian: 4
    public bool AddJobOfferToDatabase(JobOffer jobOffer)
    {
        if (!IsJobOfferInDatabase(jobOffer))
        {
            JobOffersRepo.AddJobOfferToDatabase(jobOffer);//Dodajemy ofertę do Bazy danych
            JobOffersCollection.Add(jobOffer); //Dodajemy ofertę do listy wyświetlanej w JobOffersTab
            return true;
        }
        return false;
    }

    1 odwołanie | Mariusz Mańka, 2 dni temu | 1 autor, 1 zmiana
    public bool EditPersonalData(PersonalData personalData)
    {
        try
        {
            PersonalDataRepo.EditPersonalDataInDatabase(personalData, currentPersonId);
            PersonalDataCollection[0] = personalData;
            return true;
        }
        catch (Exception e)
        {
            MessageBox.Show(e.ToString());
            return false;
        }

    }
}
```

Zdj.7 Implementacja modelu.

6. Wnioski.

Po właściwym rozbudowaniu aplikacji na pewno mogła by być ona przydatnym narzędziem zarówno dla firm poszukujących odpowiednich pracowników oraz dla osób poszukujących pracy.

Pomysły na rozbudowę aplikacji w przyszłości:

- Stworzenie logowania dla każdego użytkownika.
- Możliwość przesyłania dokumentów za pomocą aplikacji.
- Umawianie spotkań miedzy pracodawcą a kandydatem.
- Postawienie aplikacji na serwerze zewnętrznym, umożliwiając użytkownikom logowanie z różnych urządzeń za pomocą internetu.