DOKUMENTACJA TECHNICZNA

1. Metryka Projektu

- **Nazwa systemu:** System Ewidencji Pracowników i Obliczania Wynagrodzeń
- **Wersja:** 1.0
- **Język programowania:** C# (.NET 6.0 / 8.0)
- **Technologia interfejsu:** WPF (Windows Presentation Foundation)
- **Format zapisu danych:** JSON (`System.Text.Json`)
- **Środowisko programistyczne:** Visual Studio 2022

2. Architektura Systemu

System został zaprojektowany zgodnie z paradygmatem programowania obiektowego (OOP). Architektura dzieli się na dwie główne warstwy:

2.1. Warstwa Logiki Biznesowej (Backend)

Odpowiada za przetwarzanie danych, walidację i obliczenia. Kluczowe elementy:

- **`Pracownik` (Klasa abstrakcyjna):** Definiuje wspólny interfejs dla wszystkich typów pracowników. Zawiera mechanizm walidacji numeru PESEL w setterze właściwości.
- **`PracownikEtatowy` i `Zleceniobiorca`:** Klasy pochodne implementujące specyficzne metody obliczania wynagrodzenia (`ObliczPensje`).
- **`SystemKadrowy`:** Klasa zarządzająca kolekcją (`List<T>`). Pełni rolę kontrolera – obsługuje dodawanie obiektów, zapobiega duplikatom oraz realizuje serializację danych.

2.2. Warstwa Prezentacji (Frontend)

- **`MainWindow.xaml`:** Definicja wyglądu aplikacji oparta na języku znaczników XAML using Grid layout.
- **`MainWindow.xaml.cs`:** Code-behind obsługujący zdarzenia interfejsu (kliknięcia przycisków, wybór z listy rozwijanej) i komunikujący się z obiektem `SystemKadrowy`.

3. Struktura Danych i Plików

Dane przechowywane są lokalnie w pliku `baza_kadrowa.json`. Serializacja odbywa się przy użyciu biblioteki `System.Text.Json` z opcją `WriteIndented = true`, co zapewnia czytelność pliku dla człowieka.

Przykładowa struktura zapisu (JSON):

JSON

```
[
  {
    "$type": "etat",
    "PensjaZasadnicza": 5000.0,
```

```
    "Imie": "Jan",
    "Nazwisko": "Kowalski",
    "Pesel": "90010112345"
  },
  {
    "$type": "zlecenie",
    "StawkaGgodzinowa": 50.0,
    "LiczbaGodzin": 160,
    "Imie": "Anna",
    "Nazwisko": "Nowak",
    "Pesel": "92030354321"
  }
]
```

Atrybut `$type` (dyskryminator) jest generowany automatycznie dzięki zastosowaniu atrybutu `[JsonDerivedType]` w klasie bazowej, co umożliwia polimorficzną deserializację.

4. Obsługa Błędów

System wykorzystuje własną klasę wyjątków `KadryException` do zgłaszania błędów logiki biznesowej.

- Próba dodania pracownika z błędnym PESEL (inna długość niż 11 znaków).
- Próba ustawienia ujemnej pensji lub stawki.
- Próba dodania duplikatu pracownika.

Wyjątki te są przechwytywane w warstwie GUI i prezentowane użytkownikowi w formie okien dialogowych `MessageBox`.