

Usługi sieciowe w biznesie

Anonimizacja danych wrażliwych. Automatyczne narzędzie do ochrony prywatności danych.

Julia Skalska

April 22, 2024

Contents

1	Wstęp	3
2	Opis problemu	3
3	Projekt i implementacja	3
3.1	Użyte biblioteki	3
3.2	Główna funkcja	4
3.3	Funkcje anonimizacji danych	5
3.4	Klasa Form	9
3.5	Import danych	11
3.6	Eksport danych i zapis do pliku	12
3.7	Formularz	13
3.8	Obsługa zdarzenia kliknięcia przycisku w aplikacji	14
4	Demonstracja działania programu	16
5	Przykład wykorzystania - analiza zanonimizowanych danych w Collab	18
6	Podsumowanie i wnioski	22

1 Wstęp

W dzisiejszych czasach ochrona danych osobowych staje się coraz ważniejsza. Imiona, nazwiska, numery pesel, czy telefonu, są szczególnie wrażliwe i mogą być wykorzystywane w nieodpowiedni sposób, jeśli nie są odpowiednio zabezpieczone. Organizacje są zobowiązane do przestrzegania przepisów dotyczących ochrony danych, takich jak Ogólne Rozporządzenie o Ochronie Danych (RODO) w Unii Europejskiej. W tej sytuacji istnieje pilna potrzeba skutecznych narzędzi do anonimizacji danych, które pozwolą organizacjom na zachowanie pełnej funkcjonalności swoich systemów, jednocześnie zapewniając odpowiedni poziom ochrony prywatności użytkowników.

2 Opis problemu

W odpowiedzi na rosnące wymagania związane z prywatnością użytkowników, postanowiłam zająć się problemem anonimizacji danych wrażliwych. Celem mojego projektu było stworzenie programu umożliwiającego zabezpieczanie informacji zawartych w plikach typu .csv oraz .txt.

Program został zaprojektowany w taki sposób, aby automatycznie identyfikować wrażliwe informacje w podanych plikach. Następnie przeprowadza proces anonimizacji, zamieniając je na bezosobowe lub pseudonimizowane dane, które nie pozwalają na identyfikację konkretnych osób. Po zakończeniu procesu anonimizacji, użytkownik ma możliwość zapisania zmodyfikowanych danych w tym samym formacie w wybranym miejscu na komputerze, zapewniając tym samym ochronę prywatności i zgodność z obowiązującymi przepisami dotyczącymi ochrony danych.

Zdecydowałam się stworzyć program w języku C sharp, który opiera się na paradygmacie programowania obiektowego. Dzięki temu mogłam zaprojektować intuicyjny interfejs w formie formularza, który umożliwia użytkownikowi łatwe operowanie narzędziem. Program obsługuje import danych, proces anonimizacji oraz zapis do plików, zapewniając użytkownikowi rozwiązanie w zakresie ochrony danych wrażliwych.

3 Projekt i implementacja

3.1 Użyte biblioteki

Biblioteki, które użyłam do programu to:

```
using System; //dostęp do podstawowych typów danych, takich jak string, int, double itd.
using System.Collections.Generic; //generyczne kolekcje, takie jak listy, czy słowniki
using System.Data; //funkcje do zarządzania danymi, tutaj DataTable
using System.IO; //odczytywanie i zapisywanie plików
using System.Linq; //kolekcje w stylu SQL
using System.Text; //operacje na ciągach znaków
using System.Text.RegularExpressions; //używanie wyrażeń regularnych
using System.Windows.Forms; //klasy i kontrolki
```

Wyjaśnienie znaczenia bibliotek:

- System: Podstawowa biblioteka, która zapewnia dostęp do podstawowych typów danych, takich jak string, int, double itd.
- System.Collections.Generic: Ta biblioteka zawiera generyczne kolekcje, takie jak listy, czy słowniki. W kodzie używana jest do listy do przechowywania imion i nazwisk.
- System.Data: Zapewnia funkcje do zarządzania danymi, takie jak DataTable, które są używane do przechowywania danych odczytanych z plików.
- System.IO: Umożliwia operacje wejścia/wyjścia, takie jak odczytywanie i zapisywanie plików. Używana jest do odczytu danych z plików i zapisu anonimizowanych danych do nowych plików.
- System.Linq: Umożliwia programowanie zapytań dla kolekcji w stylu SQL. W kodzie używane są wyrażenia LINQ do różnych operacji na kolekcjach.

- System.Text: Zapewnia różne operacje na ciągach znaków, takie jak manipulacje ciągami, kodowanie i dekodowanie.
- System.Text.RegularExpressions: Umożliwia używanie wyrażeń regularnych, co jest używane w tym kodzie do identyfikacji i anonimizacji różnych typów danych, takich jak imiona, nazwiska, numery PESEL, hasła itd.
- System.Windows.Forms: Jest to biblioteka GUI dla aplikacji Windows Forms. Zapewnia klasy i kontrolki, takie jak przyciski, pola tekstowe, itd., które są używane w tym kodzie.

3.2 Główna funkcja

Funkcja `AnonimizeData` przyjmuje jako argument `DataTable`, który zawiera dane do anonimizacji. Tworzy nowy obiekt `Random`, który będzie używany do generowania losowych wartości. Następnie funkcja przechodzi przez każdy wiersz i każdą kolumnę w `DataTable`. Funkcja dla różnych wartości, szuka i zmienia inne komórki. Dla: -Imienia, sprawdza czy wartość w kolumnie `IMIE` jest zgodna z jednym z imion na liście `combinedNamesList`, funkcja anonimizuje to imię, używając funkcji `AnonymizeName`.

-Nazwiska, sprawdza czy wartość w kolumnie `NAZWISKO` jest zgodna z jednym z nazwisk na liście `combinedSurnamesList`, funkcja anonimizuje to nazwisko, używając funkcji `AnonymizeName`.

-Peselu, jeśli wartość w kolumnie `PESEL` jest zgodna z określonym wzorcem, funkcja anonimizuje ten identyfikator, używając funkcji `AnonymizeIdentifier`.

-Hasła, jeśli wartość komórki pasuje do wzorca hasła, funkcja zastępuje hasło znakami maski, używając funkcji `MaskPassword`.

-E-mail, jeśli wartość komórki pasuje do wzorca adresu e-mail, funkcja anonimizuje ten adres e-mail, używając funkcji `AnonymizeEmail`.

-Numeru telefonu, jeśli wartość komórki pasuje do wzorca numeru telefonu, funkcja anonimizuje ten numer, używając funkcji `AnonymizePhoneNumber`.

Kod:

```
//Funkcja anonimizacji danych
private void AnonimizeData(DataTable dataTable)
{
    Random random = new Random(); //random używany do generowania losowych wartości

    foreach (DataRow row in dataTable.Rows) //dla każdego wiersza
    {
        foreach (DataColumn column in dataTable.Columns) //dla każdej kolumny
        {
            string cellValue = row[column].ToString(); //konwertuje dane na stringi
            string columnName = column.ColumnName.ToUpper(); //zmieniam nazwy kolumn
            na duże litery
            // Sprawdzamy, czy wartość z kolumny "IMIE" znajduje się na liście imion
            if (columnName == "IMIE" && combinedNamesList.Contains(cellValue.ToUpper()))
            {
                row[column] = AnonymizeName(cellValue, random, checkBox1.Checked);
                //anonimizujemy jeśli zaznaczony jest checkbox
            }
            else if (columnName == "NAZWISKO" && combinedSurnamesList.Contains(cellValue.
                ToUpper())) // Sprawdzamy, czy wartość z kolumny "NAZWISKO" znajduje się
            na liście imion
            {
                row[column] = AnonymizeName(cellValue, random, checkBox2.Checked);
            }
            else if (columnName == "PESEL")
            {
                if (IsMatchingPattern(cellValue, "[0-9]{11}$")) //Sprawdzamy czy wartość
                w kolumnie pesel jest zgodna ze wzorcem
                {
```


Kod:

```
//Anonimizacja imienia i nazwiska
private string AnonymizeName(string Name, Random random, bool anonymize)
{
    string pattern = @"^[A-ZĄĆĘŁŃÓŚŻźa-ząćęłńóśź]*$"; //wzorzec

    if (Regex.IsMatch(Name, pattern)) //jeśli pasuje do wzorca
    {
        char firstLetter = Name[0]; //pobieranie pierwszej litery
        string randomChars = GenerateRandomString(4, random); //generowanie 4 losowych znaków
        return anonymize ? firstLetter + randomChars : Name; //łącznie litery i znaków
    }
    else
    {
        return Name; //w przeciwnym razie zwraca nie zmienioną wartość
    }
}
```

W powyższym kodzie używamy funkcji GenerateRandomString. Umożliwia ona generowanie losowego ciągu znaków o określonej długości. Przyjmuje dwa argumenty:

length: żądana długość ciągu znaków do wygenerowania,

random: obiekt klasy Random, używany do generowania losowych liczb.

Używając metody Enumerable.Repeat, tworzony jest ciąg chars powtórzony length razy. Następnie, dla każdego elementu tego ciągu, wybierany jest losowy znak za pomocą metody random.Next(s.Length). Wynikowe losowe znaki są zbierane w tablicy i następnie konwertowane z powrotem na ciąg znaków za pomocą konstruktora new string().

kod:

```
//Generowanie losowego ciągu znaków o określonej długości
private string GenerateRandomString(int length, Random random)
{
    const string chars = "abcdefghijklmnopqrstuvwxyz"; //znaki do generowania ciągu
    return new string(Enumerable.Repeat(chars, length) //ciąg znaków powtórzony length razy
        .Select(s => s[random.Next(s.Length)]).ToArray());
    //wybierany jest kolejny losowy znak
}
```

Funkcja AnonymizeIdentifier używana jest do anonimizacji identyfikatorów, w tym przypadku numerów PESEL. Sprawdzany jest wzorzec w funkcji AnonimizeData, dlatego ta funkcja tylko anonimizuje dane wartości. Zmienna nextIdentifier służy do przechowywania kolejnego identyfikatora, który zostanie użyty do anonimizacji. Na początku ma wartość 0.

Funkcja przyjmuje dwa argumenty:

identifier: numer PESEL, który chcemy anonimizować,

anonymize: flaga logiczna, która decyduje, czy numer PESEL ma być anonimizowany czy nie. Określa ją checkbox.

Jeśli anonymize jest ustawione na true, aktualizuje identyfikator na wartość nextIdentifier uzupełnioną zerami z przodu do długości 11 znaków za pomocą metody PadLeft(11, '0'). Następnie zwiększa wartość nextIdentifier o 1. W przypadku false, nic się nie zmienia.

kod:

```
//Anonimizacja peselu
private int nextIdentifier = 0; //przechowywanie kolejnego identyfikatora
private string AnonymizeIdentifier(string identifier, bool anonymize)
{
    if (anonymize) //Jeżeli checkbox jest wciśnięty
    {
        identifier = nextIdentifier.ToString().PadLeft(11, '0'); //aktualizuje identyfikator
        nextIdentifier++; // następnie zwiększa się o jeden
    }
}
```

```

        return identifier;
    }
    else
    {
        return identifier;    // Jeśli checkbox jest wyłączony, zwróć oryginalny numer PESEL
    }
}

```

Funkcja MaskPassword jest używana do zastępowania hasła gwiazdkami w celu zanonimizowania go. Sprawdzany jest wzorzec w funkcji AnonimizeData, dlatego ta funkcja tylko anonimizuje dane wartości.

Przyjmuje dwa argumenty:

password: hasło, które chcemy zanonimizować,

anonymize: flaga logiczna(checkbox), która decyduje, czy hasło ma być zanonimizowane czy nie.

Jeśli anonymize jest ustawione na true, funkcja zwraca ciąg znaków składający się z gwiazdek ('*') o takiej samej długości jak długość hasła, co oznacza, że oryginalne hasło zostaje całkowicie zastąpione gwiazdkami.

kod:

```

//Anonimizacja hasła
private string MaskPassword(string password, bool anonymize)
{
    if (anonymize) //Jeżeli checkbox jest wciśnięty
    {
        return new string('*', password.Length); // Zwróć ciąg gwiazdek '*'
        o takiej samej długości jak hasło
    }
    else
    {
        return password; // W przeciwnym razie, zwróć oryginalne hasło
    }
}

```

Funkcja AnonymizeEmail służy do anonimizacji adresów e-mail na podstawie określonych reguł. Sprawdzany jest wzorzec w funkcji AnonimizeData, dlatego ta funkcja tylko anonimizuje dane wartości. Zmienna nextPersonNumber podobnie jak nextIdentifier, służy do przechowywania kolejnego numeru osoby, która zostanie użyty do anonimizacji adresów e-mail. Domyślnie jest ona ustawiona na wartość 1.

Przyjmuje dwa argumenty:

email: adres e-mail, który chcemy anonimizować,

anonymize: flaga logiczna(checkbox), która decyduje, czy adres e-mail ma być anonimizowany czy nie.

Określany jest wzorzec wyrażenia regularnego pattern dla adresu e-mail, aby sprawdzić, czy podany adres e-mail jest prawidłowy. Jeśli anonymize jest ustawione na true, sprawdzane jest, czy email pasuje do wzorca wyrażenia regularnego za pomocą metody Regex.IsMatch. Jeśli email pasuje do wzorca: Adres jest rozdzielany na dwie części: część przed znakiem @ i część po znaku @. Część przed @ jest zastępowana anonimowym numerem osoby (np. osoba1, osoba2, itd.), zwiększając nextPersonNumber o 1 za każdym razem. Zwracany jest anonimowy adres e-mail, w którym część przed @ jest zastąpiona anonimowym numerem osoby.

kod:

```

//Anonimizacja emaila
private int nextPersonNumber = 1; //przechowywanie kolejnego numeru osoby
private string AnonymizeEmail(string email, bool anonymize)
{
    string pattern = @"^[a-zA-Z0-9._%+~]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$"; // Wzorzec
    wyrażenia regularnego dla adresu e-mail

    if (anonymize) //Jeżeli checkbox jest wciśnięty
    {

```

```

        if (Regex.IsMatch(email, pattern)) //Jeśli wzorzec pasuje
        {
            string[] parts = email.Split('@'); // Rozdzielamy adres e-mail na dwie części:
            część przed @ i część po @
            string partBeforeAtSign = parts[0]; // Pobieramy część przed @
            return "osoba" + nextPersonNumber++ + "@" + parts[1]; // Zwracamy połączone
            anonimowe część przed @, @example.pl
        }
        else
        {
            return email;
        }
    }
    else
    {
        return email; // Jeśli checkbox jest wyłączony, zwróć oryginalny adres e-mail
    }
}

```

Funkcja AnonymizePhoneNumber służy do anonimizacji numerów telefonu na podstawie określonych reguł.

Przyjmuje dwa argumenty:

phoneNumber: numer telefonu, który chcemy anonimizować,

anonymize: flaga logiczna(checkbox), która decyduje, czy numer telefonu ma być anonimizowany czy nie.

Określany jest wzorzec wyrażenia regularnego pattern dla numeru telefonu, aby sprawdzić, czy podany numer telefonu jest prawidłowy. Jeśli anonymize jest ustawione na true, sprawdzane jest, czy numer pasuje do wzorca wyrażenia regularnego za pomocą metody Regex.IsMatch. Jeśli phoneNumber pasuje do wzorca, to z numeru telefonu usuwane są znaki specjalne i plus za pomocą Regex.Replace. Sprawdzane jest, czy po wyczyszczeniu numeru telefonu pozostaje przynajmniej trzy cyfry. Jeśli tak, trzy pierwsze cyfry numeru są zachowywane, a pozostałe cyfry są zastępowane znakami 'x'. Zwracany jest zanonimizowany numer telefonu.

kod:

```

//Anonimizacja numeru telefonu
private string AnonymizePhoneNumber(string phoneNumber, bool anonymize)
{
    string pattern = @"^\+?[0-9]{3}-?[0-9]{6,12}$"; // Wzorzec wyrażenia regularnego
    dla numeru telefonu

    if (anonymize) //jeśli checkbox jest włączony
    {
        if (Regex.IsMatch(phoneNumber, pattern)) //jeśli pasuje do wyrażenia regularnego
        {
            string cleanedPhoneNumber = Regex.Replace(phoneNumber, @"[~0-9]", "");
            //Usuwanie znaki specjalne i plus z numeru telefonu
            if (cleanedPhoneNumber.Length >= 3)
            //Sprawdzamy, czy numer telefonu ma przynajmniej trzy cyfry
            {
                string firstThreeDigits = cleanedPhoneNumber.Substring(0, 3);
                //Zachowujemy trzy pierwsze cyfry
                string restOfDigits = new string('x', cleanedPhoneNumber.Length - 3);
                //Zastępujemy resztę cyfr 'x'
                return firstThreeDigits + restOfDigits;
                //Zwracamy zanonimizowany numer telefonu
            }
            else
            {

```



```

        return cleanedPhoneNumber;
        //Jeśli numer telefonu ma mniej niż trzy cyfry, zwracamy go bez zmian
    }
}
else
{
    return phoneNumber;
}
}
else
{
    return phoneNumber; // Jeśli checkbox jest wyłączony,
    zwracamy oryginalny numer telefonu
}
}
}

```

3.4 Klasa Form

Na początku programu zaraz po imporcie bibliotek mamy zdefiniowaną klasę Form1 w przestrzeni nazw Projekt, która dziedziczy po klasie Form. Najpierw zdefiniowane są prywatne statyczne listy ciągów (List<string>), takie jak: femaleNameList: lista imion żeńskich, maleNameList: lista imion męskich, femaleSurnameList: lista nazwisk żeńskich, maleSurnameList: lista nazwisk męskich, combinedNamesList: połączona lista imion, combinedSurnamesList: połączona lista nazwisk. Następnie w konstruktorze klasy Form1, po wywołaniu metody InitializeComponent(), która inicjalizuje wszystkie komponenty zdefiniowane w formularzu, ładowane są listy imion i nazwisk z określonych plików CSV za pomocą metody LoadNameList.

Są to pobierane dane z Głównego Urzędu Statystycznego, które zawierają w sobie imiona i nazwiska męskie oraz żeńskie. Następnie robione są z nich wspólne listy combinedNamesList i combinedSurnamesList, które wykorzystywane są w głównej funkcji do anonimizacji imion i nazwisk. Pomagają one w sprawdzeniu czy wartości wprowadzone z wartościami z listy są takie same. Wspólne listy uzyskiwane są za pomocą metody MergeNameLists.

kod:

```

namespace Projekt
{
    public partial class Form1 : Form
    {
        //Wprowadzam listy, które będą służyć do przechowywania danych z plików
        do anonimizacji imienia i nazwiska
        private static List<string> femaleNameList = new List<string>();
        private static List<string> maleNameList = new List<string>();
        private static List<string> femaleSurnameList = new List<string>();
        private static List<string> maleSurnameList = new List<string>();
        private static List<string> combinedNamesList = new List<string>();
        private static List<string> combinedSurnamesList = new List<string>();

        public Form1()
        {
            InitializeComponent(); //inicjowanie komponentów z formularza
            //Pobieranie danych do list
            LoadNameList(@"C:\Users\Asus\Downloads\8_-_Wykaz_imion_żeńskich_osób_żyjących_wg_pola_
            _imię_pierwsze_występujących_w_rejestrze_PESEL_bez_zgonów.csv", femaleNameList);
            LoadNameList(@"C:\Users\Asus\Downloads\8_-_Wykaz_imion_męskich_osób_żyjących_wg_
            pola_imię_pierwsze_występujących_w_rejestrze_PESEL_bez_zgonów.csv", maleNameList);
            LoadNameList(@"C:\Users\Asus\Downloads\nazwiska_żeńskie-z_uwzględnieniem_
            osób_zmarłych.csv", femaleSurnameList);
        }
    }
}

```

```

        LoadNameList(@"C:\Users\Asus\Downloads\nazwiska_męskie-z_uwzględnieniem_
osób_zmarłych_.csv", maleSurnameList);
//Tworzenie wspólnych list dla kobiet i mężczyzn
combinedNamesList = MergeNameLists(femaleNameList, maleNameList);
combinedSurnamesList = MergeNameLists(femaleSurnameList, maleSurnameList);
    }

    ...

```

Metoda LoadNameList wczytuje listę imion lub nazwisk z pliku CSV do podanej listy.

Argumenty funkcji to:

filePath: ścieżka do pliku CSV zawierającego listę imion lub nazwisk,

nameList: lista, do której zostaną dodane imiona lub nazwiska.

Funkcja używa StreamReader do odczytu pliku. Następnie, dopóki nie dojdzie do końca pliku (!sr.EndOfStream), czyta kolejne linie z pliku, dzieli je na części za pomocą metody Split(',') (rozdzielając je po przecinku) i dodaje pierwszą część (imię lub nazwisko) do listy nameList. Dzieje się tak, ponieważ w danych tylko pierwsza kolumna odnosi się do imienia i nazwiska, pozostałe to są niepotrzebne kolumny, dlatego bierzemy tylko pierwszą część.

kod:

```

//Funkcja wczytywania danych z pliku do podanej listy
private void LoadNameList(string filePath, List<string> nameList)
{
    using (StreamReader sr = new StreamReader(filePath))
    //zastosowano instancję StreamReader do odczytania tekstu z pliku
    {
        while (!sr.EndOfStream)
        {
            string line = sr.ReadLine();
            string[] parts = line.Split(','); //podział za pomocą metody split po przecinku
            nameList.Add(parts[0]); // Dodajemy tylko imię lub nazwisko do listy,
            reszta kolumn pomijamy
        }
    }
}

```

Metoda MergeNameLists łączy dwie listy imion lub nazwisk w jedną listę. Zostało to zrobione, żeby połączyć imiona męskie i żeńskie w jedną listę.

Argumenty funkcji to:

list1: pierwsza lista imion lub nazwisk,

list2: druga lista imion lub nazwisk.

Funkcja tworzy nową pustą listę mergedList, a następnie dodaje do niej wszystkie elementy z list1 i list2 za pomocą metody AddRange(). W tym programie argumentem list1 to femaleNameList lub femaleSurnameList, a list2 to maleNameList lub maleSurnameList. Na koniec zwraca połączoną listę.

kod:

```

// Funkcja do połączenia obu list w jedną
private static List<string> MergeNameLists(List<string> list1, List<string> list2)
{
    List<string> mergedList = new List<string>();
    //tworzona jest nowa lista o nazwie mergedList
    mergedList.AddRange(list1); //za pomocą addrange dodawane są listy z argumentów
    mergedList.AddRange(list2);
    return mergedList;
}

```

3.5 Import danych

ReadText funkcja służy do wczytywania danych z pliku tekstowego i zapisywania ich w obiekcie DataTable.

Argumentem jest filePath: ścieżka do pliku tekstowego, który ma być wczytany.

Tworzy ona nowy obiekt DataTable. Używa StreamReader do odczytu pliku tekstowego z określoną kodowaniem (Encoding.UTF8). Odczytuje pierwszą linię pliku, aby ustalić nazwy kolumn (nagłówki). Jeśli pierwsza linia nie jest pusta, dodaje nazwy kolumn do DataTable. Odczytuje kolejne linie pliku, dzieli je na poszczególne elementy za pomocą przecinka i dodaje je jako wiersze do DataTable. Sprawdza, czy liczba elementów w każdym wierszu odpowiada liczbie kolumn w DataTable. Jeśli nie, wypisuje komunikat o błędzie.

ReadCSV funkcja, która również służy do wczytywania danych z pliku CSV i zapisywania ich w obiekcie DataTable.

Argumentem jest filePath: ścieżka do pliku CSV, który ma być wczytany.

Funkcja działa w ten sam sposób co funkcja ReadText.

kod:

```
//Funkcja wczytania danych z pliku tekstowego
private DataTable ReadText(string filePath)
{
    DataTable dataTable = new DataTable(); //tworzenie nowego obiektu datatable
    using (StreamReader sr = new StreamReader(filePath, Encoding.UTF8))
    //odczyt danych z pliku
    {
        string line = sr.ReadLine();

        if (!string.IsNullOrEmpty(line)) // Jeśli pierwszy wiersz nie jest pusty,
            traktuj go jako dane - nagłówki kolumn
        {
            string[] headers = line.Split(','); //podział nagłówków po przecinkach
            foreach (string header in headers)
            {
                dataTable.Columns.Add(header); //dodaje nazwy kolumn jeśli ich nie mają
            }
        }

        // Odczytywanie danych z reszty pliku
        while (!sr.EndOfStream)
        {
            string[] rows = sr.ReadLine().Split(','); //podział danych po przecinkach

            if (rows.Length == dataTable.Columns.Count) // Sprawdzenie, czy liczba
                elementów w wierszu odpowiada liczbie kolumn
            {
                DataRow dataRow = dataTable.NewRow(); //nowy obiekt
                for (int i = 0; i < rows.Length; i++)
                {
                    dataRow[i] = rows[i]; //dla każdego wiersza przypisuje do niego wartości
                }
                dataTable.Rows.Add(dataRow); //dodaje wiersz do tabeli datatable
            }
            else
            {
                // Obsługa przypadku, gdy
                liczba elementów w wierszu jest różna od liczby kolumn
                Console.WriteLine("Nieprawidłowa liczba elementów w wierszu.
                Pomijanie wiersza.");
            }
        }
    }
}
```

```

    }
    }
    }
    return dataTable;
}
//Funkcja wczytania danych z pliku csv
private DataTable ReadCSV(string filePath)
{
    DataTable dataTable = new DataTable();
    using (var reader = new StreamReader(filePath, Encoding.UTF8))

    {
        string line;
        bool isFirstRow = true;
        while ((line = reader.ReadLine()) != null)
        {
            string[] data = line.Split(',');
            if (isFirstRow)
            {
                foreach (var item in data)
                {
                    dataTable.Columns.Add(new DataColumn(item));
                }
                isFirstRow = false;
            }
            else
            {
                DataRow row = dataTable.NewRow();
                row.ItemArray = data;
                dataTable.Rows.Add(row);
            }
        }
    }
    return dataTable;
}

```

3.6 Eksport danych i zapis do pliku

Funkcja SaveText zapisuje dane z obiektu DataTable do pliku tekstowego.

Przyjmuje ona dwa argumenty:

dataTable: obiekt DataTable zawierający dane, które mają zostać zapisane.

filePath: ścieżka do pliku, w którym mają zostać zapisane dane.

Tworzy nowy obiekt StringBuilder, który będzie używany do konstrukcji zawartości pliku tekstowego. Pobiera nazwy kolumn z obiektu DataTable i dodaje je jako pierwszą linię pliku. Dla każdego wiersza w obiekcie DataTable, konwertuje jego elementy na ciągi znaków i dodaje je jako kolejne linie pliku. Zapisuje zawartość StringBuilder do pliku tekstowego za pomocą File.WriteAllText().

Funkcja SaveCSV zapisuje dane z obiektu DataTable do pliku CSV.

Przyjmuje ona dwa argumenty:

dataTable: obiekt DataTable zawierający dane, które mają zostać zapisane.

filePath: ścieżka do pliku, w którym mają zostać zapisane dane.

Działa ona w ten sam sposób, jak funkcja SaveText.

kod:

```

//Zapis danych do pliku tekstowego
private void SaveText(DataTable dataTable, string filePath)
{
    StringBuilder sb = new StringBuilder(); //tworzy nowy obiekt do zawartości pliku

```

```

// Dodanie nagłówków
IEnumerable<string> columnNames = dataTable.Columns.Cast<DataColumn>()
.Select(column => column.ColumnName);
sb.AppendLine(string.Join(",", columnNames)); //dodaje przecinki

// Dodanie danych
foreach (DataRow row in dataTable.Rows)
{
    IEnumerable<string> fields = row.ItemArray.Select(field => field.ToString());
    //konwertuje na ciągi znaków
    sb.AppendLine(string.Join(",", fields));
}

// Zapis do pliku tekstowego
File.WriteAllText(filePath, sb.ToString());
}

//Zapis danych do pliku csv
private void SaveCSV(DataTable dataTable, string filePath)
{
    StringBuilder sb = new StringBuilder();

    IEnumerable<string> columnNames = dataTable.Columns.Cast<DataColumn>()
.Select(column => column.ColumnName);
sb.AppendLine(string.Join(",", columnNames));

    foreach (DataRow row in dataTable.Rows)
    {
        IEnumerable<string> fields = row.ItemArray.Select(field => field.ToString());
        sb.AppendLine(string.Join(",", fields));
    }

    File.WriteAllText(filePath, sb.ToString());
}
}

```

3.7 Formularz

W projekcie formularza pierwszym elementem interfejsu jest textbox, który powita użytkownika tekstem: "Wybierz, które elementy chcesz zanonimizować". Poniżej znajduje się sześć pól wyboru (checkboxów), które umożliwiają wybór elementów do zanonimizowania: imienia, nazwiska, numeru PESEL, hasła, adresu e-mail oraz numeru telefonu. Po dokonaniu wyboru użytkownik może uruchomić proces zanonimizowania danych, naciskając przycisk button1.

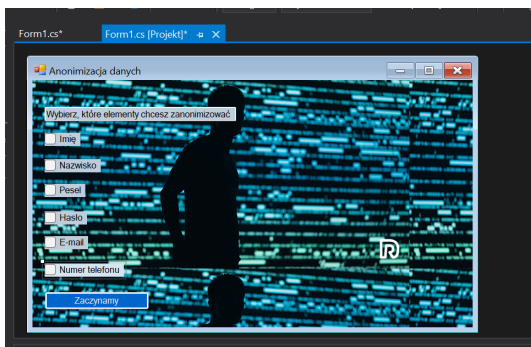


Figure 1: Projekt Formularza

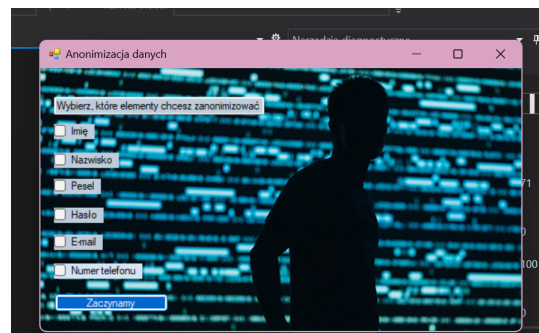


Figure 2: Formularz

3.8 Obsługa zdarzenia kliknięcia przycisku w aplikacji

Po dodaniu przycisku Button1, można określić procedurę obsługi zdarzenia jego kliknięcia.

Użytkownik ma możliwość wyboru pliku do wczytania, anonimizacji i zapisania w innym formacie.

-Wybór pliku: Tworzony jest obiekt OpenFileDialog, który pozwala użytkownikowi wybrać plik do wczytania. Ustawiane są filtry dla typów plików, które można wybrać: pliki tekstowe .txt, pliki CSV .csv oraz wszystkie pliki *.*. Jeśli użytkownik wybierze plik i kliknie "OK", jego ścieżka jest przechowywana w zmiennej filePath.

-Odczyt danych z pliku do DataTable: Sprawdzane jest rozszerzenie wybranego pliku: Jeśli rozszerzenie to .txt, dane są wczytywane do DataTable za pomocą funkcji ReadText(filePath). Jeśli rozszerzenie to .csv, dane są wczytywane do DataTable za pomocą funkcji ReadCSV(filePath). Jeśli rozszerzenie nie jest obsługiwane, użytkownikowi wyświetlany jest komunikat o błędzie i procedura zostaje zakończona.

-Anonimizacja danych: Po wczytaniu danych do DataTable, jest ona przekazywana do funkcji AnonimizeData(dataTable), która zajmuje się anonimizacją danych w tabeli.

-Zapis danych do nowych plików: Tworzony jest obiekt SaveFileDialog, który pozwala użytkownikowi wybrać miejsce i nazwę pliku do zapisania wczytanych i anonimizowanych danych. Ustawiane są filtry dla typów plików, w których można zapisać dane: pliki tekstowe .txt, pliki CSV .csv. Jeśli użytkownik wybierze miejsce i nazwę pliku oraz kliknie "OK", dane są zapisywane w wybranym formacie:

Jeśli rozszerzenie wybranego pliku to .txt, dane są zapisywane za pomocą funkcji SaveText(dataTable, newFilePath).

Jeśli rozszerzenie wybranego pliku to .csv, dane są zapisywane za pomocą funkcji SaveCSV(dataTable, newFilePath).

Po zapisaniu danych użytkownikowi wyświetlany jest komunikat informacyjny, że dane zostały zanonimizowane i zapisane.

kod:

```
//Procedura obsługi zdarzenia przycisku
private void button1_Click(object sender, EventArgs e)
{
    //Wybór pliku
    OpenFileDialog openFileDialog = new OpenFileDialog(); //nowy obiekt do wyświetlania
    okna dialogowego
    openFileDialog.Filter = "Pliki tekstowe (*.txt)|*.txt|Pliki CSV (*.csv)|*.csv|Wszystkie
    pliki (*.*)|*.*"; //typy plików do wyboru
    openFileDialog.RestoreDirectory = true; //powrót do oryginalnego katalogu po zamknięciu
    - formularza wyboru

    //Odczyt danych z pliku do Datatable
    if (openFileDialog.ShowDialog() == DialogResult.OK)
    {
        string filePath = openFileDialog.FileName; //wybieramy ścieżkę pliku
        DataTable dataTable = null; //tworzymy obiekt datatable

        if (filePath.EndsWith(".txt")) //sprawdzamy rozszerzenie pliku
        {
            dataTable = ReadText(filePath); //wczytanie pliku
        }
        else if (filePath.EndsWith(".csv")) //sprawdzamy rozszerzenie pliku
        {
            dataTable = ReadCSV(filePath); //wczytanie pliku
        }
        else
        {
            MessageBox.Show("Nieobsługiwany format pliku."); //komunikat o błędzie
            return;
        }
    }
}
```

```

    }

    //Anonimizacja danych
    AnonimizeData(dataTable); //przekazanie danych do funkcji

    //Zapis danych do nowych plików, podobnie jak z otwarciem
    SaveFileDialog saveFileDialog = new SaveFileDialog();
    saveFileDialog.Filter = "Pliki tekstowe (*.txt)|*.txt|Pliki CSV (*.csv)|*.csv";
    saveFileDialog.RestoreDirectory = true;

    if (saveFileDialog.ShowDialog() == DialogResult.OK)
    {
        string newFilePath = saveFileDialog.FileName; //wybór ścieżki i nazwa pliku
        if (newFilePath.EndsWith(".txt"))
        {
            SaveText(dataTable, newFilePath); //zapis pliku
        }
        else if (newFilePath.EndsWith(".csv"))
        {
            SaveCSV(dataTable, newFilePath); //zapis pliku
        }
        MessageBox.Show("Dane zostały zanonimizowane i zapisane."); //komunikat
        informacyjny
    }
}
}
}

```

4 Demonstracja działania programu

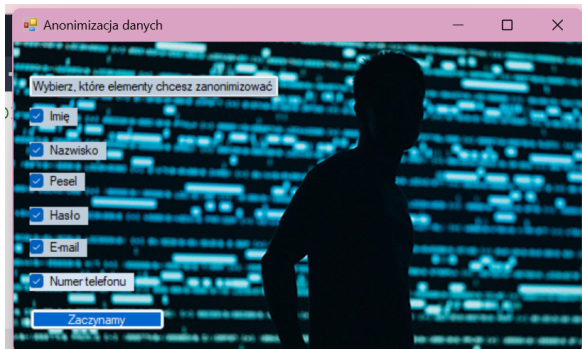


Figure 3: Interfejs

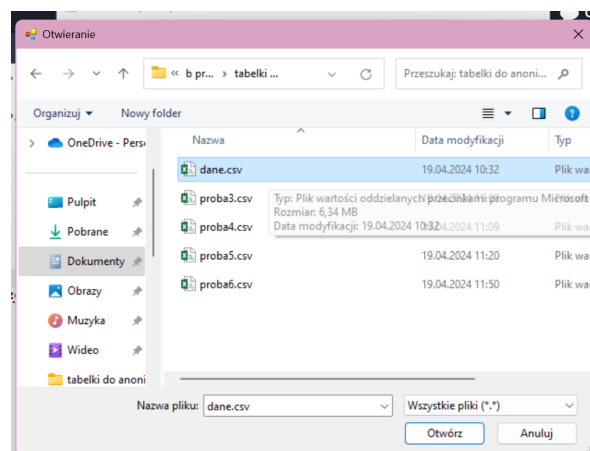


Figure 4: Otwieranie pliku

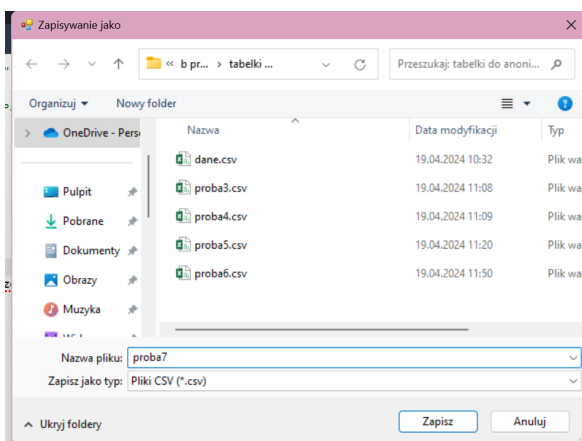


Figure 5: Zapis pliku

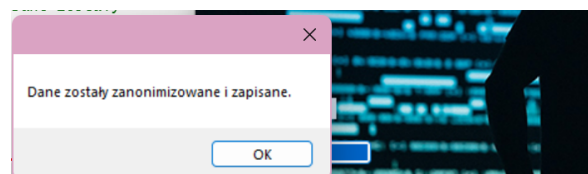


Figure 6: Alert potwierdzenia zapisu

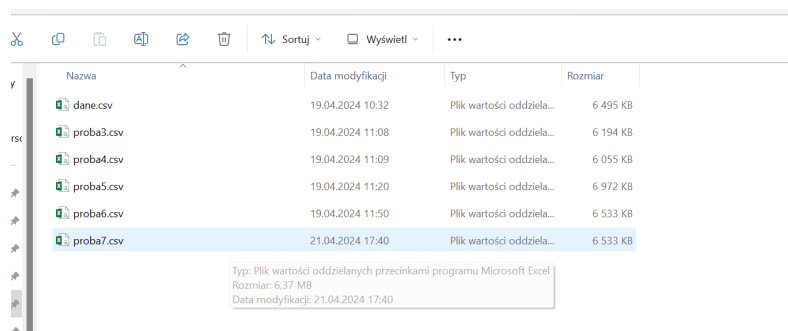


Figure 7: Zapisany plik

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
2	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
3	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
4	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
5	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
6	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
7	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
8	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
9	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
10	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
11	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
12	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
13	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
14	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
15	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
16	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
17	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
18	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
19	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
20	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
21	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
22	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		

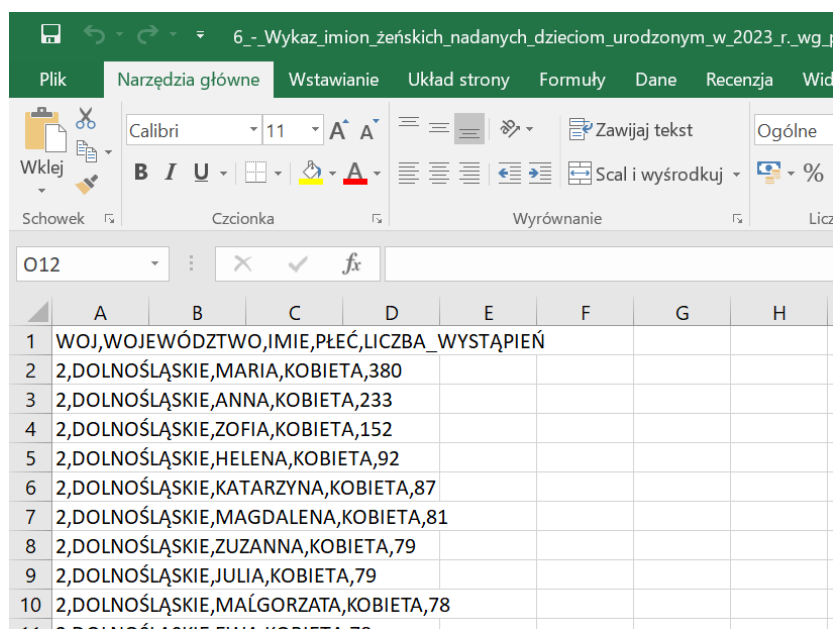
Figure 8: Dane po anonimizacji

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
2	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
3	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
4	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
5	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
6	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
7	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
8	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
9	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
10	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
11	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
12	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
13	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
14	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
15	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
16	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
17	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
18	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
19	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
20	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
21	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		
22	imię,nazwisko,peSEL,hasło,email,numer_telefonu																		

Figure 9: Dane przed anonimizacją

5 Przykład wykorzystania - analiza zanonimizowanych danych w Collab

Zdecydowałam się przetestować program pod kątem jego funkcjonalności, przeprowadzając dwie analizy. Pierwsza z nich koncentruje się na ilości wystąpień kobiet w poszczególnych województwach, natomiast druga analiza skupia się na liczbie wystąpień dziewczynek, uwzględniając pierwszą literę imienia. Pobrałam dane ze strony <https://dane.gov.pl> o nazwie: Wykaz imion żeńskich nadanych dzieciom urodzonym w 2023 r. Dane mają 5 kolumn o nazwach: woj,województwo,imie,płeć,liczba wystąpień.



	A	B	C	D	E	F	G	H
1	WOJ,WOJEWÓDZTWO,IMIE,PŁEĆ,LICZBA_WYSTĄPIEŃ							
2	2,DOLNOŚLĄSKIE,MARIA,KOBIETA,380							
3	2,DOLNOŚLĄSKIE,ANNA,KOBIETA,233							
4	2,DOLNOŚLĄSKIE,ZOFIA,KOBIETA,152							
5	2,DOLNOŚLĄSKIE,HELENA,KOBIETA,92							
6	2,DOLNOŚLĄSKIE,KATARZYNA,KOBIETA,87							
7	2,DOLNOŚLĄSKIE,MAGDALENA,KOBIETA,81							
8	2,DOLNOŚLĄSKIE,ZUZANNA,KOBIETA,79							
9	2,DOLNOŚLĄSKIE,JULIA,KOBIETA,79							
10	2,DOLNOŚLĄSKIE,MAŁGORZATA,KOBIETA,78							
11	2,DOLNOŚLĄSKIE,EMMA,KOBIETA,78							

Figure 10: Dane do analizy

Wgrałam dane do programu, ograniczając jego działanie do anonimizacji imienia. Zapisując plik, nadałam mu nazwę Anonimowe dane wystąpień imion dzieci.

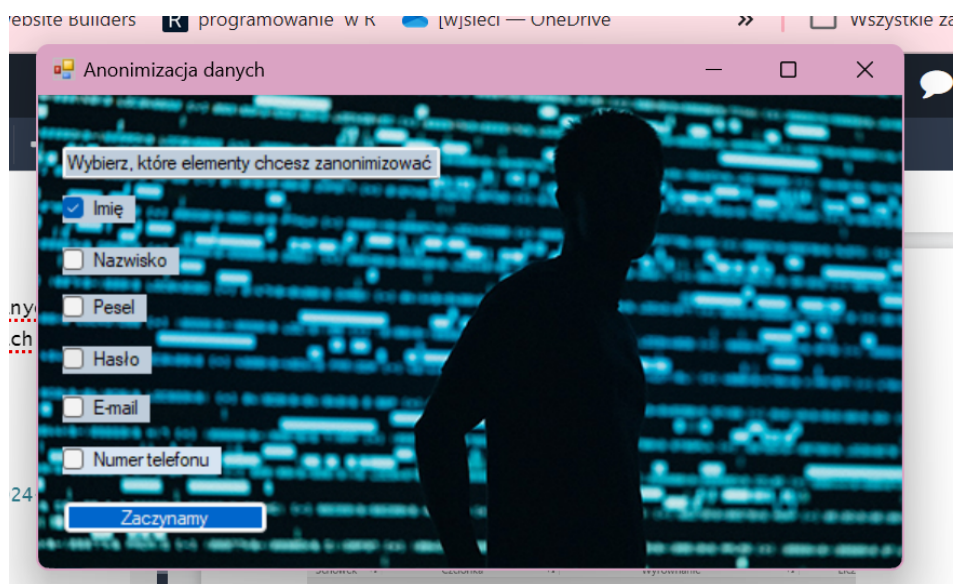


Figure 11: Program

anonymowe_dane_wystapien_imion_dzieci.csv - Excel (Ak)

	A	B	C	D	E	F	G	H	I	J
1	WOJ,WOJEWdż"DZTWO,IMIE,Pdż"Edż",LICZBA_WYSTdż"PIEŃ									
2	2,DOLNOdż"Ldż"SKIE,Mpgep,KOBIETA,380									
3	2,DOLNOdż"Ldż"SKIE,Aybbb,KOBIETA,233									
4	2,DOLNOdż"Ldż"SKIE,Zival,KOBIETA,152									
5	2,DOLNOdż"Ldż"SKIE,Hisgv,KOBIETA,92									
6	2,DOLNOdż"Ldż"SKIE,Kifke,KOBIETA,87									
7	2,DOLNOdż"Ldż"SKIE,Mpfyi,KOBIETA,81									
8	2,DOLNOdż"Ldż"SKIE,Zgwmi,KOBIETA,79									
9	2,DOLNOdż"Ldż"SKIE,Jxdef,KOBIETA,79									
10	2,DOLNOdż"Ldż"SKIE,Maomw,KOBIETA,78									
11	2,DOLNOdż"Ldż"SKIE,Elbah,KOBIETA,78									
12	2,DOLNOdż"Ldż"SKIE,Abene,KOBIETA,78									

Figure 12: Dane po anonimizacji

Proces anonimizacji przebiegł pomyślnie. Następnie wybrałam odpowiednie narzędzie, które ułatwi mi wykorzystanie moich zanonimizowanych danych. Wybrałam Google Colab. Jest to platforma do analizy danych i uczenia maszynowego w chmurze, dostarcza ona środowisko Jupyter Notebook w przeglądarce internetowej, co umożliwia tworzenie, edycję i uruchamianie notatników Jupyter bezpośrednio w chmurze. Językiem programowania, którym będę robiła analizy będzie Python.

projekt

Plik Edytuj Widok Wstaw Środowisko wykonawcze Narzędzia Pomoc Wszystkie zmiany zostały zapisane

Pliki

sample_data

```
[ ] import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

[ ] path = r"/content/anonymowe_dane_wystapien_imion_dzieci.csv"
data = pd.read_csv(path, encoding='utf-8')

[ ] data.columns = ['WOJ', 'WOJEWODZTWO', 'IMIE', 'PLEC', 'LICZBA_WYSTAPIEN']

print(data)
```

	WOJ	WOJEWODZTWO	IMIE	PLEC	LICZBA_WYSTAPIEN
0	2	DOLNOŁĘSKIE	Mpgep	KOBIETA	380
1	2	DOLNOŁĘSKIE	Aybbb	KOBIETA	233
2	2	DOLNOŁĘSKIE	Zival	KOBIETA	152
3	2	DOLNOŁĘSKIE	Hisgv	KOBIETA	92
4	2	DOLNOŁĘSKIE	Kifke	KOBIETA	87
...
2541	32	ZACHODNIOPOMORSKIE	Zrgau	KOBIETA	2
2542	32	ZACHODNIOPOMORSKIE	Ekicc	KOBIETA	2
2543	32	ZACHODNIOPOMORSKIE	Hruau	KOBIETA	2
2544	32	ZACHODNIOPOMORSKIE	Wlhm	KOBIETA	2
2545	32	ZACHODNIOPOMORSKIE	Lnmm	KOBIETA	2

[2546 rows x 5 columns]

Figure 13: Import bibliotek i pliku do analizy

W moim notatniku rozpoczęłam od importowania niezbędnych bibliotek. Wykorzystałam pandas do manipulacji i analizy danych, matplotlib.pyplot do tworzenia różnorodnych wizualizacji danych oraz seaborn do bardziej zaawansowanych wizualizacji. Następnie zdefiniowałam ścieżkę do pliku CSV zawierającego dane i wczytałam ten plik do zmiennej data za pomocą funkcji read_csv z biblioteki pandas. Zauważyłam niepoprawne znaki w nazwach kolumn i dokonałam ich korekty, zastępując je odpowiednimi polskimi znakami. Wyświetliłam wczytane dane.

```

województwa_poprawne = {
    'DOLNOŚLĄSKIE': 'Dolnośląskie',
    'KUJAWSKO-POMORSKIE': 'Kujawsko-Pomorskie',
    'LUBELSKIE': 'Lubelskie',
    'LUBUSKIE': 'Lubuskie',
    'MAZOWIECKIE': 'Mazowieckie',
    'MAŁOPOLSKIE': 'Małopolskie',
    'OPOLSKIE': 'Opolskie',
    'PODKARPACKIE': 'Podkarpackie',
    'PODLASKIE': 'Podlaskie',
    'POMORSKIE': 'Pomorskie',
    'WARMIŃSKO-MAZURSKIE': 'Warmińsko-Mazurskie',
    'WIELKOPOLSKIE': 'Wielkopolskie',
    'ZACHODNIOPOMORSKIE': 'Zachodniopomorskie',
    'ŚLĄSKIE': 'Śląskie',
    'ŚWIĘTOKRZYSKIE': 'Świętokrzyskie',
    'ŁÓDZKIE': 'Łódzkie'
}

# Zmiana nazw w kolumnie 'WOJEWODZTWO'
data['WOJEWODZTWO'] = data['WOJEWODZTWO'].map(województwa_poprawne)

# Wyświetlenie zmienionych nazw
print(data['WOJEWODZTWO'].unique())

['Dolnośląskie' 'Kujawsko-Pomorskie' 'Lubelskie' 'Lubuskie' 'Łódzkie'
'Małopolskie' 'Mazowieckie' 'Opolskie' 'Podkarpackie' 'Podlaskie'
'Pomorskie' 'Śląskie' 'Świętokrzyskie' 'Warmińsko-Mazurskie'
'Wielkopolskie' 'Zachodniopomorskie']

[ ] grouped_data = data[data['PLEC'] == 'KOBIETA'].groupby('WOJEWODZTWO')['LICZBA_WYSTAPIEN'].sum().reset_index()

[ ] print(grouped_data)

   WOJEWODZTWO  LICZBA_WYSTAPIEN
0  Dolnośląskie             4236
1  Kujawsko-Pomorskie         1757
2  Lubelskie                 1203
3  Lubuskie                   943
4  Mazowieckie              5362
5  Małopolskie              8780
6  Opolskie                  2135
7  Podkarpackie             3482
8  Podlaskie                  183
9  Pomorskie                 4285
10 Warmińsko-Mazurskie         653
11 Wielkopolskie             3084
12 Zachodniopomorskie        1664

```

Figure 14: Dane po anonimizacji

Rozpoczęłam od zdefiniowania słownika 'województwa poprawne', który przypisuje niepoprawne nazwy województw do ich poprawnych odpowiedników z polskimi znakami. Następnie wyświetliłam zmienione nazwy województw. Pogrupowałam dane według województwa dla płci żeńskiej i zsumowałam liczbę wystąpień dziewczyn o każdym województwie. Wyniki prezentujące liczbę urodzeń dziewcząt w poszczególnych województwach pozwoliły mi na przegląd wyników.

Kolejnym krokiem było przedstawienie danych w formie wykresu w celu lepszego zobrazowania analizy. Utworzyłam wykres kołowy, który ilustruje procentowy udział liczby wystąpień dziewcząt w każdym województwie w stosunku do ogółu. Na podstawie wykresu oraz wcześniej przygotowanych danych można stwierdzić, że województwo Śląskie (10754) jest liderem w nadawaniu imion dziewcząt. Na kolejnych miejscach znalazły się województwa Małopolskie (8780) i Mazowieckie (5362). Najmniejsza liczba imion została nadana w województwie Podlaskim (183). Analizę tę można odnieść do liczby urodzeń, ponieważ nadawanie imion jest związane z tym zjawiskiem.

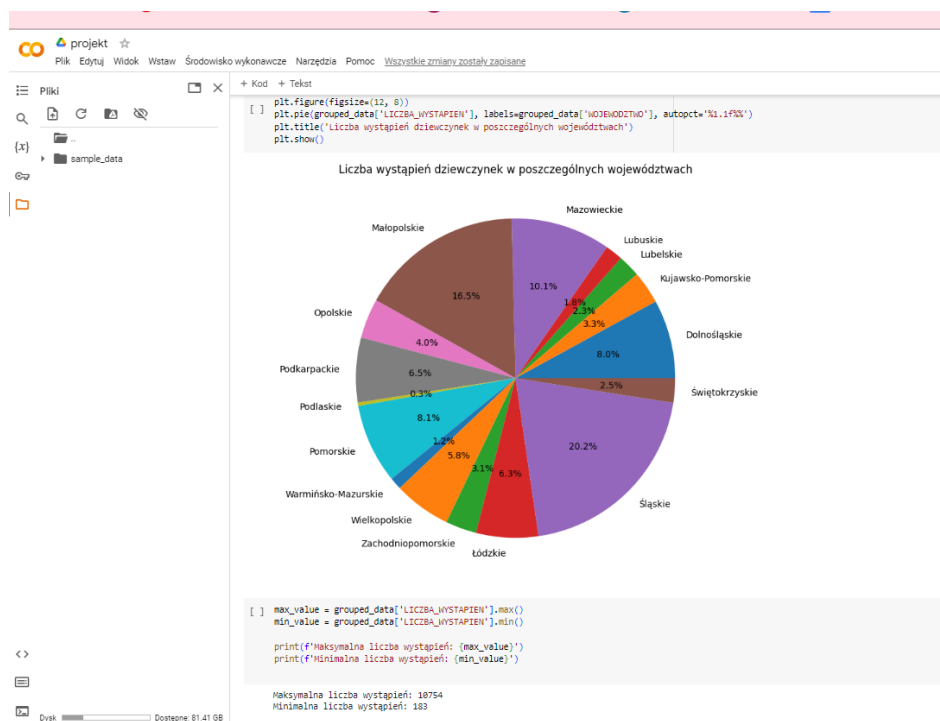


Figure 15: Dane po anonimizacji

W celu sprawdzenia poprawności analizy, wykonałam obliczenia minimum i maksimum z pogrupowanych danych. Potwierdziło to obliczenia.

Zdecydowałam się na kolejną analizę, dotyczącą liczby wystąpień pierwszych liter w imionach dziewczynek. Kod został zaprojektowany tak, aby pobierać pierwszą literę z anonimizowanego imienia i następnie grupować dane według tych liter, sumując liczbę ich wystąpień. Wyniki tej analizy zostały wyświetlone w postaci pogrupowanych danych.

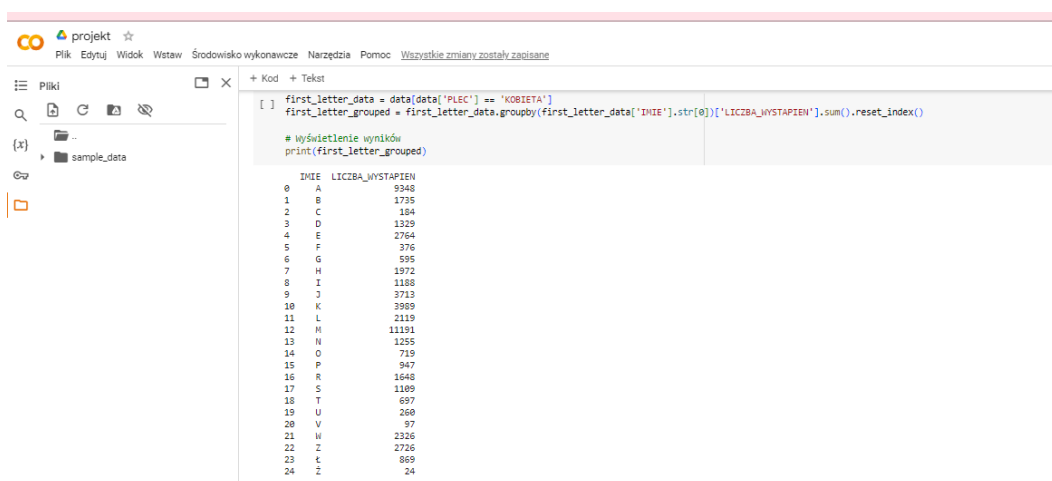


Figure 16: Dane po anonimizacji

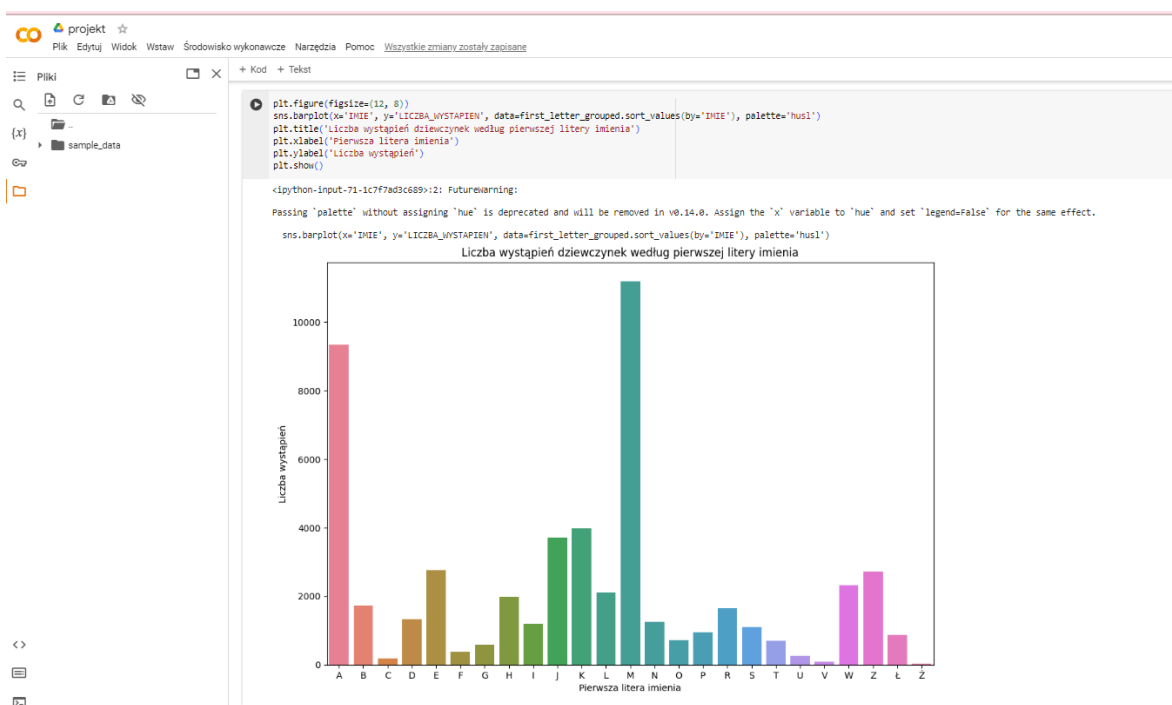


Figure 17: Dane po anonimizacji

Utworzyłam wykres słupkowy w celu bardziej czytelnej wizualizacji wyników analizy. Wykres wskazuje, że litera "M" dominuje w liczbie wystąpień, za nią plasują się litery "A" i "K". Najrzadziej występującą literą jest "Z".

Przeprowadzone analizy ukazują wartość zanonimizowanych danych w kontekście różnorodnych analiz. Dzięki zachowanej pierwszej literze w każdym imieniu możemy obserwować częstotliwość występowania poszczególnych liter oraz dokonywać obliczeń dotyczących liczby urodzeń, grupując je według województw, gdyż nadawanie imienia jest związane z tym zjawiskiem.

6 Podsumowanie i wnioski

Projektowanie narzędzia do anonimizacji danych w plikach CSV i TXT miało na celu ochronę prywatności użytkowników i spełnienie wymogów przepisów, takich jak RODO w Unii Europejskiej. Narzędzie to zostało stworzone w języku C sharp z wykorzystaniem paradygmatu programowania obiektowego, co pozwoliło na zaprojektowanie intuicyjnego interfejsu użytkownika. Kluczowe funkcje narzędzia to automatyczna identyfikacja i anonimizacja wrażliwych danych, takich jak imiona, nazwiska, numery PESEL, hasła, adresy e-mail oraz numery telefonów. Do identyfikacji i anonimizacji danych wykorzystano wyrażenia regularne, co zapewnia precyzyjne dopasowanie wzorców i efektywne przetwarzanie.

Narzędzie oferuje różne metody anonimizacji, takie jak zastępowanie imion i nazwisk losowymi pseudonimami, zastępowanie numerów PESEL oraz adresów e-mail unikatowymi identyfikatorami oraz maskowanie haseł gwiazdkami i anonimizacja numerów telefonów. Ponadto narzędzie umożliwia wczytywanie i zapisywanie danych z plików tekstowych i CSV, a także anonimizację imion i nazwisk. Interfejs użytkownika został stworzony przy użyciu platformy Windows Forms, co zapewnia wygodę i intuicyjność obsługi.

Dzięki wykorzystaniu języka C sharp i bibliotek .NET Framework udało się stworzyć narzędzie o wysokiej funkcjonalności i intuicyjnym interfejsie. Użycie wyrażeń regularnych w procesie anonimizacji danych pozwoliło na precyzyjne i elastyczne dopasowanie wzorców, co jest kluczowe dla skuteczności anonimizacji różnych typów danych.

W ramach projektu przeprowadzono również demonstrację działania narzędzia oraz jego wykorzystanie do analizy danych w środowisku Google Colab. Przeprowadzone analizy obejmowały sprawdzenie liczby wystąpień imion żeńskich w poszczególnych województwach oraz analizę częstotliwości

występowania pierwszych liter imion. Wyniki analizy zostały przedstawione w formie wykresów, co ułatwiło wizualizację i interpretację danych. Badania potwierdziły, że narzędzie do anonimizacji danych może być użyteczne również w analizie zaszyfrowanych danych.