



PROJEKT

Temat: Aplikacja do śledzenia wydatków własnych (Financial Helper)

Wykonali:

Hudzishevskyi Oleksii

Retkowski Marcin

Kierunek: Informatyka NS, Semestr 6, Rok III

Przedmiot: Programowania aplikacji bazodanowych

Prowadzący: mgr. Inż. Tomasz Czerwiec

Gorzów Wielkopolski 07.07.2023

1. Spis treści

1. Spis treści.....	1
2. Cel projektu	2
3. Wykorzystane technologie	2
4. Realizacja projektu	3
4.1 Dodatkowe pakietu Nu-Get.....	3
4.2 Kod programowy.....	3
4.2.1 Entities	4
4.2.2 Contracts	7
4.2.3 Services	8
4.2.4 Shared Models	14
4.2.5 Formsy	15
5. Działanie aplikacji	21
6. Wnioski.....	26
7. Bibliografia.....	27
8. Spis ilustracji	29
9. Spis snippet'ów.....	29

2. Cel projektu

Celem projektu jest stworzenie aplikacji, która pomoże użytkownikom śledzić i zarządzać swoimi wydatkami osobistymi. Aplikacja ma zapewnić użytkownikom narzędzie, które umożliwi im lepsze zrozumienie i kontrolę nad swoimi finansami, pomagając oszczędzać pieniądze, planować budżet oraz monitorować postępy w osiągnięciu ich celów finansowych. Jako źródło eksportowanych danych w formacie CSV został wybrany bank PKO.

3. Wykorzystane technologie

W projekcie wykorzystane zostały takie technologie: .NET, Windows Forms, Entity Framework oraz SQLite.

.NET to platforma programistyczna opracowana przez firmę Microsoft. Zapewnia ona środowisko do tworzenia, wdrażania i uruchamiania różnorodnych aplikacji, w tym aplikacji desktopowych, webowych i mobilnych. .NET obsługuje wiele języków programowania, takich jak C#, VB.NET i F#, co daje programistom elastyczność w wyborze preferowanego języka.

Windows Forms to część framework'a .NET, która umożliwia tworzenie aplikacji desktopowych dla systemu operacyjnego Windows. Windows Forms zapewnia zestaw narzędzi i kontrolek, które ułatwiają budowanie interfejsu użytkownika. Programiści mogą projektować aplikacje, korzystając z drag-and-drop wizualnego projektanta, dodając przyciski, pola tekstowe, listy rozwijane i inne elementy interfejsu.

SQLite to lekka, serwerowa baza danych, która działa lokalnie na urządzeniu. Jest to popularna technologia wykorzystywana do przechowywania i zarządzania danymi w aplikacjach desktopowych i mobilnych. SQLite zapewnia prosty i efektywny sposób przechowywania danych w plikach baz danych, eliminując potrzebę zewnętrznego serwera bazy danych. W projekcie SQLite będzie wykorzystywane do przechowywania danych finansowych użytkowników, takich jak informacje o profilu, o zrealizowanych przelewach i kategoriach przypisanych do nich.

Entity Framework to technologia ORM (Object-Relational Mapping) będąca częścią platformy .NET. Jest to zestaw narzędzi i bibliotek, które ułatwiają mapowanie obiektowo-relacyjne między bazą danych a modelem obiekowym aplikacji. Entity Framework umożliwia pro-

gramistom definiowanie modelu danych w postaci klas i relacji między nimi, a następnie automatycznie generuje zapytania SQL i zarządza interakcją z bazą danych. W projekcie Entity Framework zostanie użyty wraz z SQLite do zarządzania operacjami na bazie danych.

4. Realizacja projektu

W tym rozdziale opisane zostały wykorzystane pakiety Nu-Get a także opisane poszczególne metody oraz klasy zrealizowane w samej aplikacji.

4.1 Dodatkowe pakietu Nu-Get

W celu zainstalowania wszystkich niezbędnych bibliotek i framework'ów zostały wykonane polecenia w Package Manager Console. Polecenia zaprezentowane na *Snippet 1*

Snippet 1. Polecenia Install-Package

```
// Entity Framework
Install-Package Microsoft.EntityFrameworkCore
Install-Package Microsoft.EntityFrameworkCore.Relational
Install-Package Microsoft.EntityFrameworkCore.Abstractions
Install-Package Microsoft.EntityFrameworkCore.Design
Install-Package Microsoft.EntityFrameworkCore.Tools
Install-Package Microsoft.EntityFrameworkCore.Proxies
Install-Package Microsoft.EntityFrameworkCore.SqlServer
Install-Package Microsoft.EntityFrameworkCore.Sqlite
// Praca z kodowaniem "windows-1250"
Install-Package System.Text.Encoding
Install-Package System.Text.Encoding.CodePages
// Szyfrowanie
Install-Package BCrypt.Net-Next
```

4.2 Kod programowy

W tym podrozdziale zostały opisane poszczególne części kodu aplikacji podzielone na takie sekcje jak **Entities** (kontekst bazy danych i encje w postaci klas C#), **Contracts** (interfejsy dla serwisów), **Services** (klasy w których realizowana jest logika współpracy z kontekstem bazy danych), **Shared Models** (modele, które pełnią rolę specjalnej, rozszerzonej lub zmodyfikowanej wersji przedstawionych danych wyciągniętych z bazy danych), **Forms** (klasy,

które przedstawiają okna aplikacji, i w których jest realizowana logika interakcji z użytkownikiem)

4.2.1 Entities

Snippet 2 reprezentuje kontekst bazy danych, w którym zdefiniowane są tablice z bazy danych i zdefiniowany jest connectionString do połączenia się z bazą danych.

Snippet 2. Kontekst bazy danych

```
using Microsoft.EntityFrameworkCore;

namespace FinancialHelper.Entities
{
    public class DatabaseContext : DbContext
    {
        public DbSet<User> Users { get; set; } = null!;
        public DbSet<BankData> BankDatas { get; set; } = null!;
        public DbSet<Category> Categories { get; set; } = null!;

        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder.UseSqlite(@"Data Source=app.db");
        }
    }
}
```

Snippet 3 reprezentuje klasę, która zawiera w sobie takie informacje jak: data operacji, data waluty, typ transakcji, kwota, waluta, saldo po transakcji, opisy transakcji, a także dwa klucze obce – id użytkownika i id kategorii.

Snippet 3. Encja "BankData"

```
namespace FinancialHelper.Entities
{
    public class BankData
    {
        public int Id { get; set; }
        public string OperationDate { get; set; }
        public DateTime ValueDate { get; set; }
        public string TransactionType { get; set; }
        public decimal Amount { get; set; }
        public string Currency { get; set; }
        public decimal SaldoAfterTransaction { get; set; }
        public string TransactionDescription { get; set; }
        public string? TransactionDescriptionAdditional1 { get; set; }
        public string? TransactionDescriptionAdditional2 { get; set; }
        public string? TransactionDescriptionAdditional3 { get; set; }
        public string? TransactionDescriptionAdditional4 { get; set; }
        public string? TransactionDescriptionAdditional5 { get; set; }
        public string? TransactionDescriptionAdditional6 { get; set; }
        public string? TransactionDescriptionAdditional7 { get; set; }

        public int UserId { get; set; }
        public User? User { get; set; }

        public int? CategoryId { get; set; }
        public Category? Category { get; set; }
    }
}
```

Snippet 4 reprezentuje klasę, której rolą jest przypisywanie wcześniej zaprezentowanej encji do specjalnej kategorii, na podstawie których później będą obliczane statystyki wydatków. W tej klasie została nadpisana metoda *ToString()*, żeby przy rzutowaniu obiektu tej klasy na typ string zwracana była tylko nazwa kategorii, która jest przechowywana we właściwości **Name**.

Snippet 4. Encja "Category"

```
namespace FinancialHelper.Entities
{
    public class Category
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Commentary { get; set; }

        public int UserId { get; set; }
        public User? User { get; set; }

        public virtual List<BankData>? BankDatas { get; set; } = new();

        public override string ToString()
        {
            return Name;
        }
    }
}
```

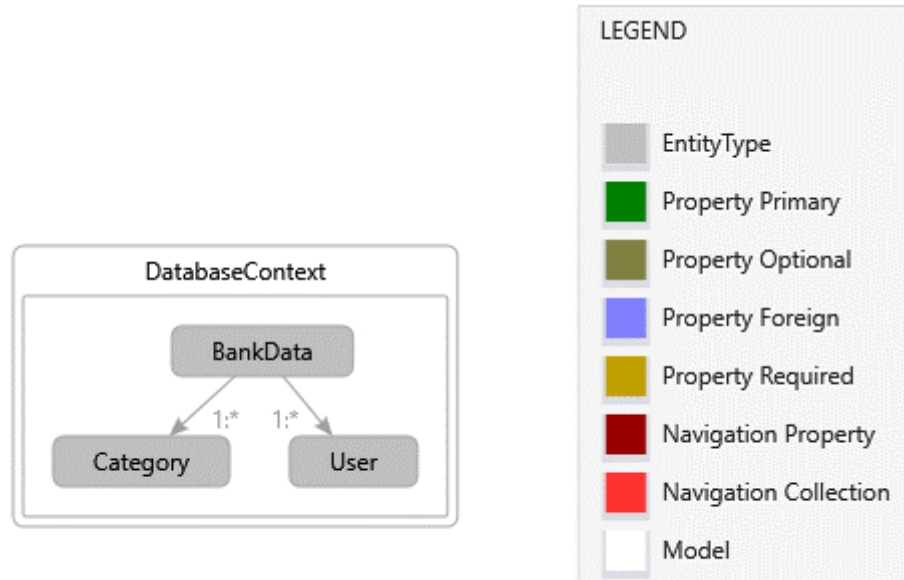
Snippet 5 reprezentuje klasę, której rolą jest przechowywanie informacji o użytkowniku, np., takie podstawowe dane jak hasło i login do aplikacji.

Snippet 5. Encja "User"

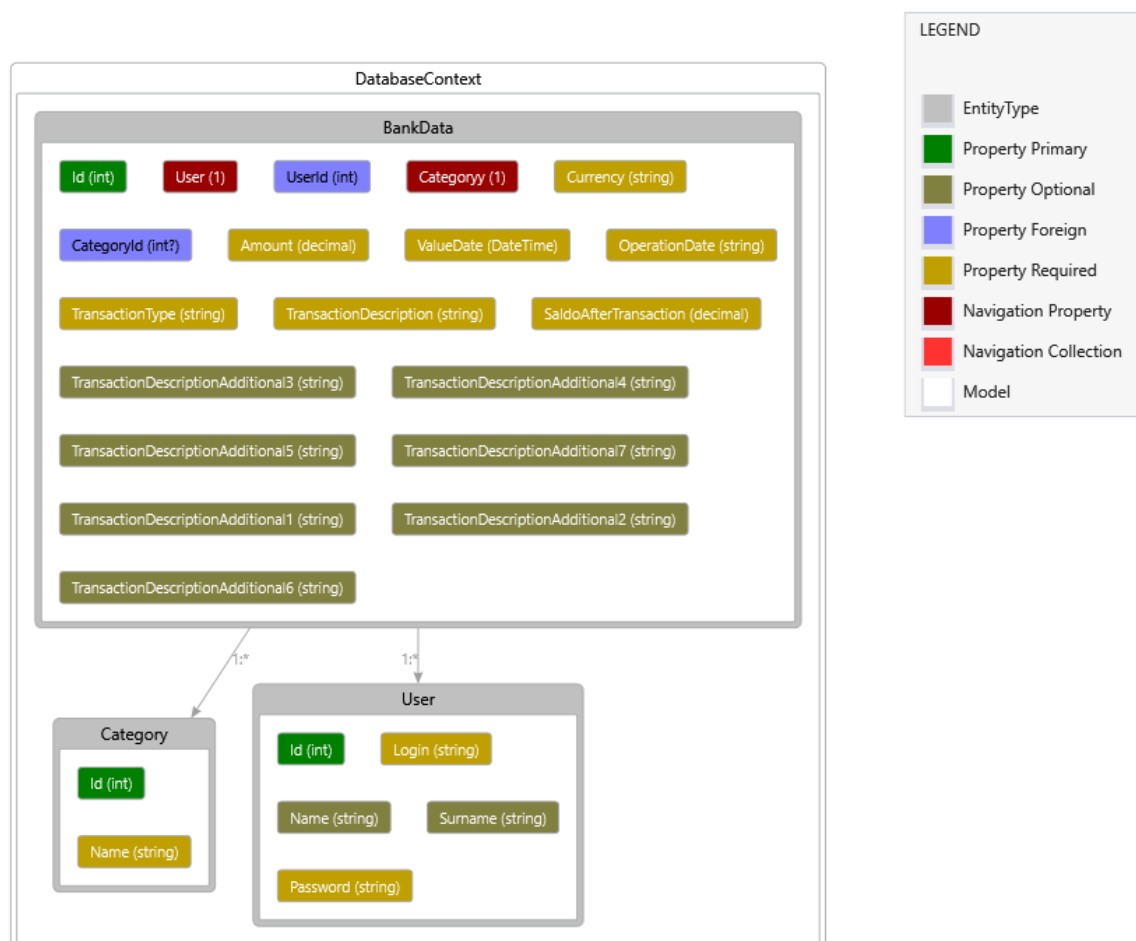
```
namespace FinancialHelper.Entities
{
    public class User
    {
        public int Id { get; set; }
        public string? Name { get; set; }
        public string? Surname { get; set; }
        public string Login { get; set; }
        public string Password { get; set; }

        public virtual List<Category> Categories { get; set; } = new();
        public virtual List<BankData> BankDatas { get; set; } = new();
    }
}
```

Na *Rysunek 1. Diagram klas kontekstu bazy danych, wersja podstawowa* i *Rysunek 2. Diagram klas kontekstu bazy danych, wersja rozszerzona* zostały przedstawione wizualnie wcześniej omówione klasy i relacje pomiędzy nimi.



Rysunek 1. Diagram klas kontekstu bazy danych, wersja podstawowa



Rysunek 2. Diagram klas kontekstu bazy danych, wersja rozszerzona

4.2.2 Contracts

Na *Snippet 6*, *Snippet 7*, *Snippet 8*, *Snippet 9* zaprezentowane są interfejsy wraz z metodami, które zostaną zrealizowane, dla serwisów o odpowiednich nazwach

Snippet 6. Interfejs IBankDataService

```
namespace FinancialHelper.Shared.Contracts
{
    public interface IBankDataService : IDisposable
    {
        List<Entities.BankData> AddDataToDB(List<Shared.BankData> bankDatas, int userId, int?
categoryId);
        Entities.BankData ModifyCategory(int bankDataId, int categoryId);
        List<Entities.BankData> GetUserData(int userId);
        List<Entities.BankData> GetSearchedData(string searchedPhrase, int userId);
        void Dispose();
        void Dispose(bool disposing);
    }
}
```

Snippet 7. Interfejs ICategoryService

```
namespace FinancialHelper.Shared.Contracts
{
    public interface ICategoryService : IDisposable
    {
        List<CategoryChartData> GetCategoryChartDatas(DateTime from, DateTime to, int userId);
        Entities.Category CreateNewCategory(string name, string commentary, int userId);
        Entities.Category DeleteCategory(int id, int userId);
        List<Entities.Category> GetCategories(int userId);
        Entities.Category ModifyCategory(int categoryId, string name, string commentary, int
userId);
        void Dispose();
        void Dispose(bool disposing);
    }
}
```

Snippet 8. Interfejs ICSVService

```
namespace FinancialHelper.Shared.Contracts
{
    public interface ICSVService : IDisposable
    {
        List<BankData> GetDataFromFile(string path);
        void Dispose();
        void Dispose(bool disposing);
    }
}
```

Snippet 9. Interfejs UserService

```
namespace FinancialHelper.Shared.Contracts
{
    public interface IUserService : IDisposable
    {
        Entities.User LogUser(string login, string password);
        Entities.User CreateUser(string login, string password);
        Entities.User DeleteUser(string login, string password);
        void Dispose();
        void Dispose(bool disposing);
    }
}
```


4.2.3 Services

Snippet 10 reprezentuje serwis, w którym zostały zrealizowane metody komunikujące się głównie z tabelą **BankData** i pozwalające dodać zaimportowane metody do profilu użytkownika (**AddDataToDB()**), zmodyfikować kategorię konkretnego rekordu **BankData** (**ModifyCateogry()**), wyciągnąć informację o wydatkach użytkownika (**GetUserData()**) a także wyciągnąć wyszukiwane informacje o wydatkach użytkownika (**GetSearchedData()**).

Snippet 10. Serwis BankDataService

```
using FinancialHelper.Entities;
using FinancialHelper.Shared.Contracts;
using Microsoft.EntityFrameworkCore;

namespace FinancialHelper.Shared.Services
{
    public class BankDataService : IBankDataService
    {
        public List<Entities.BankData> AddDataToDB(List<BankData> bankDatas, int userId, int?
categoryId)
        {
            using (DatabaseContext databaseContext = new())
            {
                List<Entities.BankData> addedRows = new();

                foreach (Shared.BankData bankData in bankDatas)
                {
                    Entities.BankData bankDataEntity = new()
                    {
                        OperationDate = bankData.OperationDate,
                        ValueDate = bankData.ValueDate,
                        TransactionType = bankData.TransactionType,
                        Amount = bankData.Amount,
                        Currency = bankData.Currency,
                        SaldoAfterTransaction = bankData.SaldoAfterTransaction,
                        TransactionDescription = bankData.TransactionDescription,
                        TransactionDescriptionAdditional1 = bankData.TransactionDescriptionAd-
ditional1,
                        TransactionDescriptionAdditional2 = bankData.TransactionDescriptionAd-
ditional2,
                        TransactionDescriptionAdditional3 = bankData.TransactionDescriptionAd-
ditional3,
                        TransactionDescriptionAdditional4 = bankData.TransactionDescriptionAd-
ditional4,
                        TransactionDescriptionAdditional5 = bankData.TransactionDescriptionAd-
ditional5,
                        TransactionDescriptionAdditional6 = bankData.TransactionDescriptionAd-
ditional6,
                        TransactionDescriptionAdditional7 = bankData.TransactionDescriptionAd-
ditional7,

                        UserId = userId,
                        CategoryId = categoryId,
                    };

                    if (!databaseContext.BankDatas.Contains(bankDataEntity))
                    {
                        addedRows.Add(bankDataEntity);
                    }
                }

                databaseContext.AddRange(addedRows);
                databaseContext.SaveChanges();

                return addedRows;
            }
        }
    }
}
```

```
    }
}

public Entities.BankData ModifyCategory(int bankDataId, int categoryId)
{
    using(DatabaseContext databaseContext = new())
    {
        var category = databaseContext.Categories.FirstOrDefault(c => c.Id == categoryId);

        Entities.BankData modifiedData = databaseContext.BankDatas
            .Where(bd => bd.Id == bankDataId)
            .FirstOrDefault();

        modifiedData.Category = category;
        databaseContext.SaveChanges();

        return modifiedData;
    }
}

public List<Entities.BankData> GetUserData(int userId)
{
    using(DatabaseContext databaseContext = new())
    {
        return databaseContext.BankDatas
            .Include(bd => bd.Category)
            .OrderByDescending(bd => bd.OperationDate)
            .Where(bd => bd.UserId == userId)
            .ToList();
    }
}

public List<Entities.BankData> GetSearchedData(string searchedPhrase, int userId)
{
    using(DatabaseContext databaseContext = new())
    {
        var result = databaseContext.BankDatas
            .Include(bd => bd.Category)
            .Where(bd => (
                bd.Id + " " +
                bd.Category!.Name + " " +
                bd.OperationDate + " " +
                bd.ValueDate + " " +
                bd.TransactionType + " " +
                bd.Amount + " " +
                bd.Currency + " " +
                bd.SaldoAfterTransaction + " " +
                bd.TransactionDescription + " " +
                bd.TransactionDescriptionAdditional1 + " " +
                bd.TransactionDescriptionAdditional2 + " " +
                bd.TransactionDescriptionAdditional3 + " " +
                bd.TransactionDescriptionAdditional4 + " " +
                bd.TransactionDescriptionAdditional5 + " " +
                bd.TransactionDescriptionAdditional6 + " " +
                bd.TransactionDescriptionAdditional7)
                .ToLower()
                .Contains(searchedPhrase.ToLower()))
            .ToList();

        return result;
    }
}

public void Dispose()
{
    Dispose(true);
    GC.SuppressFinalize(this);
}

public void Dispose(bool disposing)
{
    if (disposing) { }
}
```

```
}  
}
```

Snippet 11 reprezentuje serwis, w którym zostały zrealizowane metody komunikujące się głównie z tabelą **Category** i pozwalające utworzyć nową kategorię (**CreateNewCategory()**), usunąć kategorię (**DeleteCategory()**), zmodyfikować kategorię (**ModifyCategory()**), wyciągnąć informację o wszystkich kategoriach (**GetCategories()**) a także przygotować zestawienia dla statystyki o wydatkach dla konkretnych kateogrii (**GetCategoryChartDatas()**).

Snippet 11. Serwis CategoryService

```
using FinancialHelper.Entities;  
using FinancialHelper.Shared.Contracts;  
using Microsoft.EntityFrameworkCore;  
  
namespace FinancialHelper.Shared.Services  
{  
    public class CategoryService : ICategoryService  
    {  
        public List<CategoryChartData> GetCategoryChartDatas(DateTime from, DateTime to, int  
userId)  
        {  
            using(DatabaseContext databaseContext = new())  
            {  
                var result = databaseContext.BankDatas  
                    .Include(bd => bd.Category)  
                    .Where(bd => bd.UserId == userId && (bd.ValueDate >= from && bd.ValueDate  
<= to))  
                    .GroupBy(bd => bd.Category!.Name)  
                    .Select(bd => new CategoryChartData()  
                    {  
                        Name = bd.Key,  
                        AmountPlus = bd.Where(s => s.Amount > 0).Sum(s => (double)s.Amount),  
                        AmountMinus = bd.Where(s => s.Amount < 0).Sum(s => (double)s.Amount),  
                    })  
                    .ToList();  
  
                return result;  
            }  
        }  
  
        public Category CreateNewCategory(string name, string commentary, int userId)  
        {  
            using(DatabaseContext databaseContext = new())  
            {  
                Category category = new()  
                {  
                    Name = name,  
                    Commentary = commentary,  
                    UserId = userId  
                };  
  
                if (databaseContext.Categories.FirstOrDefault(c => c.Name == name) != null)  
throw new Exception("Such category already exists!");  
  
                databaseContext.Categories.Add(category);  
  
                databaseContext.SaveChanges();  
  
                return category;  
            }  
        }  
  
        public Category DeleteCategory(int id, int userId)  
        {  

```

```
        using (DatabaseContext databaseContext = new())
        {
            var relatedData = databaseContext.BankDatas
                .Where(bd => bd.CategoryId == id)
                .ToList();

            foreach(var item in relatedData)
            {
                item.CategoryId = null;
            }

            Category category = databaseContext.Categories.FirstOrDefault(c => c.Id ==
id);

            databaseContext.Categories.Remove(category);

            databaseContext.SaveChanges();

            return category;
        }
    }

    public List<Category> GetCategories(int userId)
    {
        using (DatabaseContext databaseContext = new())
        {
            return databaseContext.Categories
                .Where(c => c.UserId == userId)
                .ToList();
        }
    }

    public Category ModifyCategory(int categoryId, string name, string commentary, int
userId)
    {
        using (DatabaseContext databaseContext = new())
        {
            Category modifiedCategory = databaseContext.Categories.FirstOrDefault(c =>
c.Id == categoryId);

            modifiedCategory.Name = name;
            modifiedCategory.Commentary = commentary;

            databaseContext.SaveChanges();

            return modifiedCategory;
        }
    }

    public void Dispose()
    {
        Dispose(true);
        GC.SuppressFinalize(this);
    }

    public void Dispose(bool disposing)
    {
        if (disposing) { }
    }
}
```

Snippet 12 reprezentuje serwis, w którym została zrealizowana metoda (**GetDataFrom-File()**), za pomocą której jest obrabiany wyeksportowany z aplikacji bankowej plik o formacie CSV.

Snippet 12. Serwis CSVService

```
using CsvHelper;
```

```
using CsvHelper.Configuration;
using FinancialHelper.Shared.Contracts;
using System.Diagnostics;
using System.Globalization;
using System.Text;

namespace FinancialHelper.Shared.Services
{
    public class CSVService : ICSVService
    {
        public void Dispose()
        {
            Dispose(true);
            GC.SuppressFinalize(this);
        }

        public void Dispose(bool disposing)
        {
            if (disposing) { }
        }

        public List<BankData> GetDataFromFile(string path)
        {
            System.Text.Encoding.RegisterProvider(System.Text.CodePagesEncodingProvider.Instance);

            List<BankData> bankData = new();

            CsvConfiguration csvConfig = new(CultureInfo.InvariantCulture)
            {
                Encoding = Encoding.GetEncoding("windows-1250"),
                HasHeaderRecord = true,
                Delimiter = ",",
                MemberTypes = MemberTypes.Properties,
                HeaderValidated = null,
                MissingFieldFound = null,
            };

            using (StreamReader streamReader = new(path, Encoding.GetEncoding("windows-1250")))
            using (CsvReader csvReader = new(streamReader, csvConfig))
            {
                bankData = csvReader.GetRecords<BankData>().ToList();
                Debug.WriteLine($"Rows imported: {bankData.Count}");
            }

            return bankData;
        }
    }
}
```

Snippet 13 reprezentuje serwis, w którym zostały zrealizowane metody komunikujące się głównie z tabelą **User** i pozwalające na dodanie (**CreateUser()**) i usunięcie (**DeleteUser()**), a także na przeprowadzenie procesu zalogowania się do aplikacji (**LogUser()**).

Snippet 13. Serwis UserService

```
using FinancialHelper.Entities;
using FinancialHelper.Shared.Contracts;
using static BCrypt.Net.BCrypt;

namespace FinancialHelper.Shared.Services
{
    public class UserService : IUserService
    {
        public User DeleteUser(string login, string password)
        {
            using (DatabaseContext databaseContext = new())
            {
                // ...
            }
        }
    }
}
```

```
        User deletedUser = databaseContext.Users
            .Where(u => (u.Login == login))
            .FirstOrDefault();

        var pass = Verify(password, deletedUser.Password);

        if (deletedUser == null || !pass) return null;

        databaseContext.Users.Remove(deletedUser);
        databaseContext.SaveChanges();

        return deletedUser;
    }
}

public User CreateUser(string login, string password)
{
    int salt = 666;
    using (DatabaseContext databaseContext = new())
    {
        User checkedUser = databaseContext.Users
            .Where(u => (u.Login == login))
            .FirstOrDefault();

        if (checkedUser != null) return null;

        User newUser = new()
        {
            Login = login,
            Password = HashPassword(password, salt),
        };

        databaseContext.Users.Add(newUser);
        databaseContext.SaveChanges();

        return newUser;
    }
}

public User LogUser(string login, string password)
{
    using (DatabaseContext databaseContext = new())
    {
        User loggedUser = databaseContext.Users
            .Where(u => u.Login == login)
            .FirstOrDefault();

        if (loggedUser == null) return null;
        var pass = Verify(password, loggedUser.Password);

        if (pass) return loggedUser;

        return null;
    }
}

public void Dispose()
{
    Dispose(true);
    GC.SuppressFinalize(this);
}

public void Dispose(bool disposing)
{
    if (disposing) { }
}
}
```

4.2.4 Shared Models

Snippet 14 reprezentuje model, który jest wykorzystywany przy wczytywaniu danych z pliku CSV, właściwości tej klasy zostały oznaczone atrybutami z framework'u CSVHelper.

Snippet 14. Model BankData

```
using CsvHelper.Configuration.Attributes;

namespace FinancialHelper.Shared
{
    public class BankData
    {
        [Name("Data operacji")]
        [Index(0)]
        public string OperationDate { get; set; }

        [Name("Data waluty")]
        [Index(1)]
        public DateTime ValueDate { get; set; }

        [Name("Typ transakcji")]
        [Index(2)]
        public string TransactionType { get; set; }

        [Name("Kwota")]
        [Index(3)]
        public decimal Amount { get; set; }

        [Name("Waluta")]
        [Index(4)]
        public string Currency { get; set; }

        [Name("Saldo po transakcji")]
        [Index(5)]
        public decimal SaldoAfterTransaction { get; set; }

        [Name("Opis transakcji")]
        [Index(6)]
        public string TransactionDescription { get; set; }

        [Index(7)]
        public string? TransactionDescriptionAdditional1 { get; set; }

        [Index(8)]
        public string? TransactionDescriptionAdditional2 { get; set; }

        [Index(9)]
        public string? TransactionDescriptionAdditional3 { get; set; }

        [Index(10)]
        public string? TransactionDescriptionAdditional4 { get; set; }

        [Index(11)]
        public string? TransactionDescriptionAdditional5 { get; set; }

        [Index(12)]
        public string? TransactionDescriptionAdditional6 { get; set; }

        [Index(13)]
        public string? TransactionDescriptionAdditional7 { get; set; }
    }
}
```

Snippet 15 reprezentuje model klasy pomocniczej, której obiekty są przesyłane do kontrolki Chart, na której wyświetlane są statystyki.

Snippet 15. Model CategoryChartData

```
namespace FinancialHelper.Shared
{
    public class CategoryChartData
    {
        public string Name { get; set; }
        public double AmountPlus { get; set; }
        public double AmountMinus { get; set; }
    }
}
```

4.2.5 Formsy

Dla aplikacji zostały utworzone dwa okna:

- Login form – logowanie, utworzenie użytkownika i usunięcie
- Main form – importowanie danych z CSV, wyświetlanie wszystkich zaimportowanych wcześniej danych, wyświetlanie statystyki i tworzenie nowych kategorii.

4.2.5.1 Login Form

Snippet 16 reprezentuje kod, który został przypisany do kontrolek w oknie logowania, m.in. dla przycisku „Submit”, „Add new profile” i „Delete profile”.

Snippet 16. Login Form

```
using FinancialHelper.Entities;
using FinancialHelper.Shared.Services;

namespace FinancialHelper
{
    public partial class LoginForm : Form
    {
        public LoginForm()
        {
            InitializeComponent();
        }

        private void loginFormLoginTextBox_TextChanged(object sender, EventArgs e)
        {
            if (loginFormLoginTextBox.Text != "" && loginFormPasswordTextBox.Text != "") submitLoginButton.Enabled = true;
            if (loginFormLoginTextBox.Text != "" && loginFormPasswordTextBox.Text != "") registerButton.Enabled = true;
            if (loginFormLoginTextBox.Text != "" && loginFormPasswordTextBox.Text != "") deleteProfileButton.Enabled = true;
        }

        private void loginFormPasswordTextBox_TextChanged(object sender, EventArgs e)
        {
            if (loginFormLoginTextBox.Text != "" && loginFormPasswordTextBox.Text != "") submitLoginButton.Enabled = true;
            if (loginFormLoginTextBox.Text != "" && loginFormPasswordTextBox.Text != "") registerButton.Enabled = true;
            if (loginFormLoginTextBox.Text != "" && loginFormPasswordTextBox.Text != "") deleteProfileButton.Enabled = true;
        }

        private void submitLoginButton_Click(object sender, EventArgs e)
        {
            try
```



```
{
    using (UserService userService = new())
    {
        User loggedUser = userService.LogUser(
            loginFormLoginTextBox.Text,
            loginFormPasswordTextBox.Text);

        if (loggedUser == null) throw new Exception("Such user does not exist!");

        MainForm mainForm = new MainForm(loggedUser.Id);
        mainForm.Show();
        this.Hide();
    }
}
catch (Exception ex)
{
    MessageBox.Show($"Error occured upon logging. More details:\n{ex.Message}",
        "Error");
}

private void registerButton_Click(object sender, EventArgs e)
{
    try
    {
        using (UserService userService = new())
        {
            User loggedUser = userService.CreateUser(
                loginFormLoginTextBox.Text,
                loginFormPasswordTextBox.Text);

            if (loggedUser == null) throw new Exception("Such user with this login al-
ready exists!");

            MainForm mainForm = new MainForm(loggedUser.Id);
            mainForm.Show();
            this.Hide();
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Error occured upon creating new profile. More deta-
ils:\n{ex.Message}", "Error");
    }
}

private void deleteProfileButton_Click(object sender, EventArgs e)
{
    try
    {
        using (UserService userService = new())
        {
            User loggedUser = userService.DeleteUser(
                loginFormLoginTextBox.Text,
                loginFormPasswordTextBox.Text);

            if (loggedUser == null) throw new Exception("Such user does not exist!!");

            MessageBox.Show("Profile successfully deleted");
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Error occured upon deleting profile. More details:\n{ex.Mes-
sage}", "Error");
    }
}
}
```

4.2.5.2 Main Form

Snippet 17 reprezentuje kod, który został przypisany do kontrolek w oknie głównym na podstronie „Import Data”. Wyróżnić tutaj można utworzenie i wyświetlenie obiektu dialogowego okna, za pomocą którego jest wybierany plik, z którego zaciągane dane do profilu.

Snippet 17. Main Form - Import page

```
private void importButton_Click(object sender, EventArgs e)
{
    OpenFileDialog openFileDialog = new()
    {
        InitialDirectory = @"C:\",
        Title = "Import Data from CSV",

        CheckFileExists = true,
        CheckPathExists = true,

        DefaultExt = "csv",
        Filter = "CSV UTF-8 (Comma delimited)(*.csv)|*.csv",
        FilterIndex = 2,
        RestoreDirectory = true,

        ReadOnlyChecked = true,
        ShowReadOnly = true,
    };

    if (openFileDialog.ShowDialog() == DialogResult.OK)
    {
        try
        {
            List<Shared.BankData> importedData = new();

            using (CSVService csvService = new())
            {
                importCSVProgressBar.PerformStep();
                importCSVProgressBar.Update();

                importedData = csvService.GetDataFromFile(openFileDialog.FileName);
            }

            MessageBox.Show($"Imported rows: {importedData.Count}", "Information");

            importCSVProgressBar.PerformStep();
            importCSVProgressBar.Update();

            this.bankDataDataGridView.DataSource = new BindingList<Shared.BankData>(imported-
Data);
            this.ImportedBankDatas = importedData;

            if (importedData.Count > 0)
            {
                addToProfileButton.Enabled = true;
            }

            importCSVProgressBar.Value = 100;
            importCSVProgressBar.Update();
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Error occurred upon data importation. More details:\n{ex.Mes-
sage}", "Error");
        }
        finally
        {
            importCSVProgressBar.Value = 0;
            importCSVProgressBar.Update();
        }
    }
}

private void addToProfileButton_Click(object sender, EventArgs e)
```

```
{
    try
    {
        using (BankDataService bankDataService = new())
        {
            this.BankDatas = bankDataService.AddDataToDB(this.ImportedBankDatas, loggedUserId,
null);

            MessageBox.Show($"Added Rows to Your Profile: {this.BankDatas.Count}\nDuplicates:
{this.ImportedBankDatas.Count - this.BankDatas.Count}", "Information");

            refreshYourProfileData();
            clearImportDataGridView();

            addToProfileButton.Enabled = false;
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Error occured upon adding data to your profile. More deta-
ils:\n{ex.Message}", "Error");
    }
}
```

Snippet 18 reprezentuje kod, który został przypisany do kontrolek w oknie głównym na podstronie „Your finances”. Wyróżnić tutaj można obrabianie zdarzenia zmiany w zaznaczonej komórce w DataGridView, za pomocą tego zdarzenia jest obserwowana zmiana kategorii.

Snippet 18. Main Form - Your Finances Page

```
private void searchButton_Click(object sender, EventArgs e)
{
    try
    {
        using (BankDataService bankDataService = new())
        {
            var result = this.BankDatas = bankDataService.GetSearchedData(searchTextBox.Text,
loggedUserId);
            if (result != null)
            {
                this.profileDataGridView.DataSource = new BindingList<Entities.BankData>(re-
sult);
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Error occured upon adding data to your profile. More deta-
ils:\n{ex.Message}", "Error");
    }
}

private void profileDataGridView_CellValueChanged(object sender, DataGridViewCellEventArgs e)
{
    if (this.profileDataGridView.Rows.Count > 0)
    {
        try
        {
            string newCategory = this.profileDataGridView.Rows[e.RowIndex].Cells[0].Value.To-
String();
            var category = this.Categories.Where(c => c.Name == newCategory).FirstOrDefault();
            var dataId = Int32.Parse(this.profileDataGridView.Rows[e.RowIndex].Cells[1].Va-
lue.ToString());
            var data = this.BankDatas.Where(bd => bd.Id == dataId).FirstOrDefault();
            if (category.Id != data.CategoryId)
            {

```

```
        using (BankDataService bankDataService = new())
        {
            bankDataService.ModifyCategory(dataId, category.Id);

            this.BankDatas = bankDataService.GetUserData(loggedUserId);
            this.profileDataGridView.DataSource = new BindingList<Entities.BankData>(this.BankDatas);
            this.profileDataGridView.Refresh();
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Error occured upon modificating data category. More details:\n{ex.Message}", "Error");
    }
}
```

Snippet 19 reprezentuje kod, który został przypisany do kontrolek w oknie głównym na podstronie „Your categories”.

Snippet 19. Main Form - Categories Page

```
private void submitCategoryButton_Click(object sender, EventArgs e)
{
    try
    {
        using (CategoryService categoryService = new())
        {
            categoryService.CreateNewCategory(
                newCategoryNameTextBox.Text,
                newCategoryCommentaryRichTextBox.Text,
                loggedUserId);
        }

        MessageBox.Show($"New category has benn successfully added", "Information");
        refreshYourCategoryDate();
        refreshYourProfileData();
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Error occured upon adding new category. More details:\n{ex.Message}", "Error");
    }
    finally
    {
        newCategoryCommentaryRichTextBox.Text = "";
        newCategoryNameTextBox.Text = "";
    }
}

private void newCategoryNameTextBox_TextChanged(object sender, EventArgs e)
{
    if (newCategoryCommentaryRichTextBox.Text != "" && newCategoryNameTextBox.Text != "") submitCategoryButton.Enabled = true;
}

private void newCategoryCommentaryRichTextBox_TextChanged(object sender, EventArgs e)
{
    if (newCategoryCommentaryRichTextBox.Text != "" && newCategoryNameTextBox.Text != "") submitCategoryButton.Enabled = true;
}

private void categoriesDataGridView_RowStateChanged(object sender, DataGridViewRowStateChangedEventArgs e)
{
    if (categoriesDataGridView.SelectedRows.Count > 0)
    {

```

```
        modifyCategoryGroupBox.Visible = true;

        modifyCategoryIdTextBox.Text = e.Row.Cells[0].Value.ToString();
        modifyCategoryNameTextBox.Text = e.Row.Cells[1].Value.ToString();
        modifyCategoryCommentaryRichTextBox.Text = e.Row.Cells[2].Value.ToString();
    }
}

private void modifyCategoryButton_Click(object sender, EventArgs e)
{
    try
    {
        using (CategoryService categoryService = new())
        {
            categoryService.ModifyCategory(
                Int32.Parse(modifyCategoryIdTextBox.Text),
                modifyCategoryNameTextBox.Text,
                modifyCategoryCommentaryRichTextBox.Text,
                loggedUserId);

            refreshYourCategoryDate();
            refreshYourProfileData();
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Error occurred upon editing your category. More details:\n{ex.Message}", "Error");
    }
}

private void deleteCategoryButton_Click(object sender, EventArgs e)
{
    try
    {
        using (CategoryService categoryService = new())
        {
            categoryService.DeleteCategory(
                Int32.Parse(modifyCategoryIdTextBox.Text),
                loggedUserId);

            refreshYourCategoryDate();
            refreshYourProfileData();

            modifyCategoryGroupBox.Visible = false;
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Error occurred upon deleting your category. More details:\n{ex.Message}", "Error");
    }
}
```

Snippet 20 reprezentuje kod, który został przypisany do kontrolek w oknie głównym na podstronie „Your Statistics”. Wyróżnić tutaj można generowanie serii dla wykresu na podstawie danych wyciągniętych za pomocą metody **GetCategoryChartData()**.

Snippet 20. Main Form - Your Statistics Page

```
private void submitDateRangeButton_Click(object sender, EventArgs e)
{
    try
    {
        if (fromDateTimePicker.Value < toDateTimePicker.Value)
        {
            using (CategoryService categoryService = new CategoryService())
```

```
{
    var result = categoryService.GetCategoryChartDatas(
        fromDatePicker.Value,
        toDatePicker.Value,
        loggedUserId);

    foreach (var series in categoriesChart.Series)
    {
        series.Points.Clear();
    }
    categoriesChart.Series.Clear();
    categoriesChart.Titles.Clear();

    categoriesChart.Titles.Add("Category Chart");
    foreach (var data in result)
    {
        if (data.AmountMinus != 0)
        {
            categoriesChart.Series.Add(new Series(data.Name == null ? "uncategorized" + " (minus)" : data.Name + " (minus)"));
            categoriesChart.Series[data.Name == null ? "uncategorized" + " (minus)" : data.Name + " (minus)"].IsValueShownAsLabel = true;
            categoriesChart.Series[data.Name == null ? "uncategorized" + " (minus)" : data.Name + " (minus)"].Points.AddXY(data.Name == null ? " " : data.Name, data.AmountMinus);
        }

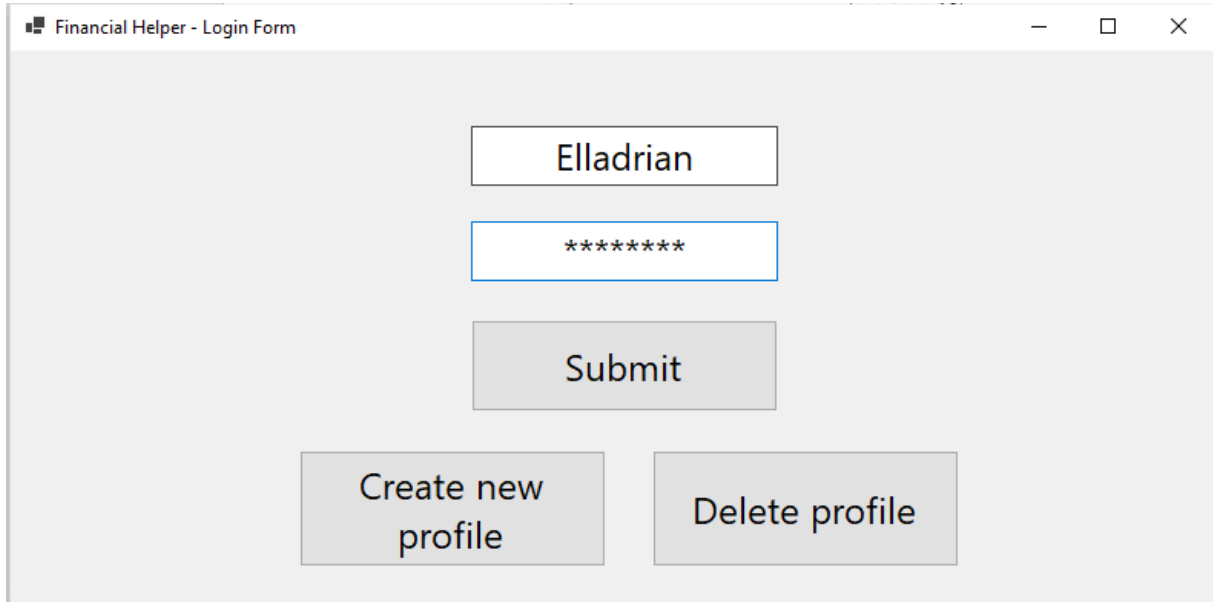
        if (data.AmountPlus != 0)
        {
            categoriesChart.Series.Add(new Series(data.Name == null ? "uncategorized" + " (plus)" : data.Name + " (plus)"));
            categoriesChart.Series[data.Name == null ? "uncategorized" + " (plus)" : data.Name + " (plus)"].IsValueShownAsLabel = true;
            categoriesChart.Series[data.Name == null ? "uncategorized" + " (plus)" : data.Name + " (plus)"].Points.AddXY(data.Name == null ? " " : data.Name, data.AmountPlus);
        }
    }

    categoriesChart.ChartAreas["ChartArea1"].AxisX.LabelAutoFitStyle = LabelAutoFitStyles.LabelsAngleStep45;
    categoriesChart.ChartAreas["ChartArea1"].AxisY.LabelAutoFitStyle = LabelAutoFitStyles.LabelsAngleStep45;
}
else
{
    MessageBox.Show($"Please select correct date range!", "Error");
}
catch (Exception ex)
{
    MessageBox.Show($"Error occured upon generating chart. More details:\n{ex.Message}", "Error");
}
}
```

5. Działanie aplikacji

Po uruchomieniu aplikacji zostanie wyświetlone pierwsze okno z logowaniem, gdzie widoczne są dwa TextBox'y i trzy Button'y. Po wpisaniu poprawnych danych do logowania lub utworzeniu nowego profilu użytkownik zostanie natychmiast przekierowany do głównego okna

aplikacji z głównymi funkcjonalnościami. Przy wpisaniu błędnych danych zostanie wyświetlony stosowny komunikat. Przy wpisaniu poprawnych danych i wybraniu opcji usunięcia profilu – profil zostanie usunięty. Przykładowy wygląd tego okna jest pokazany na *Rysunek 3*

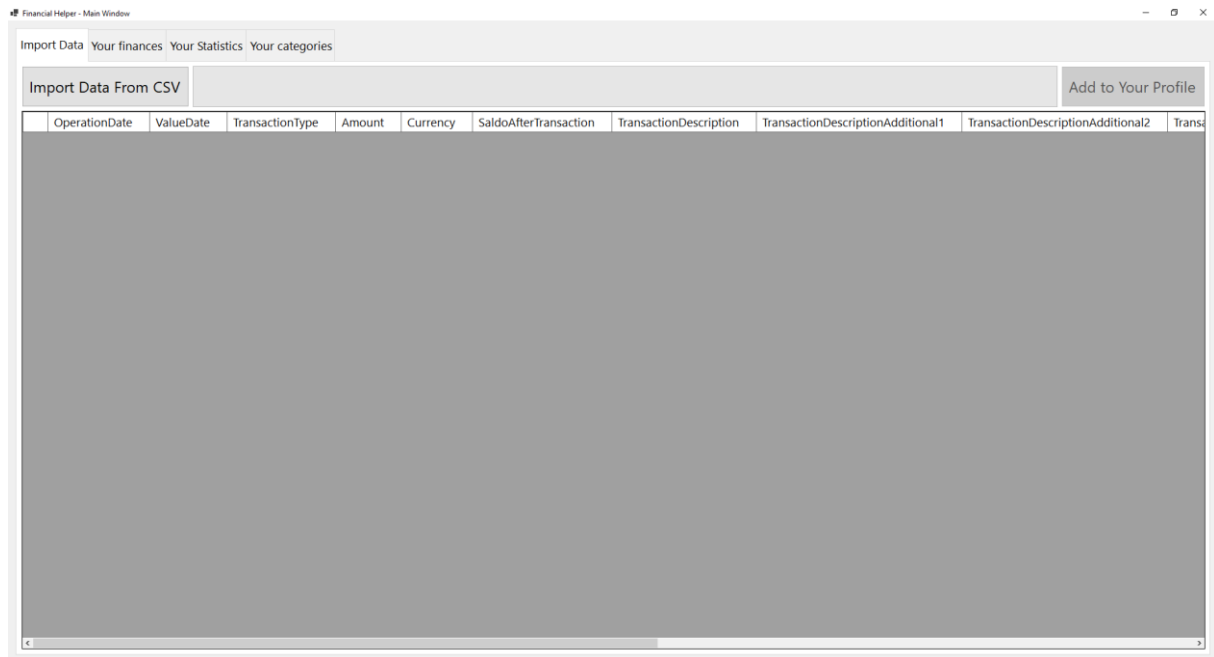


Rysunek 3. Okno Login Form

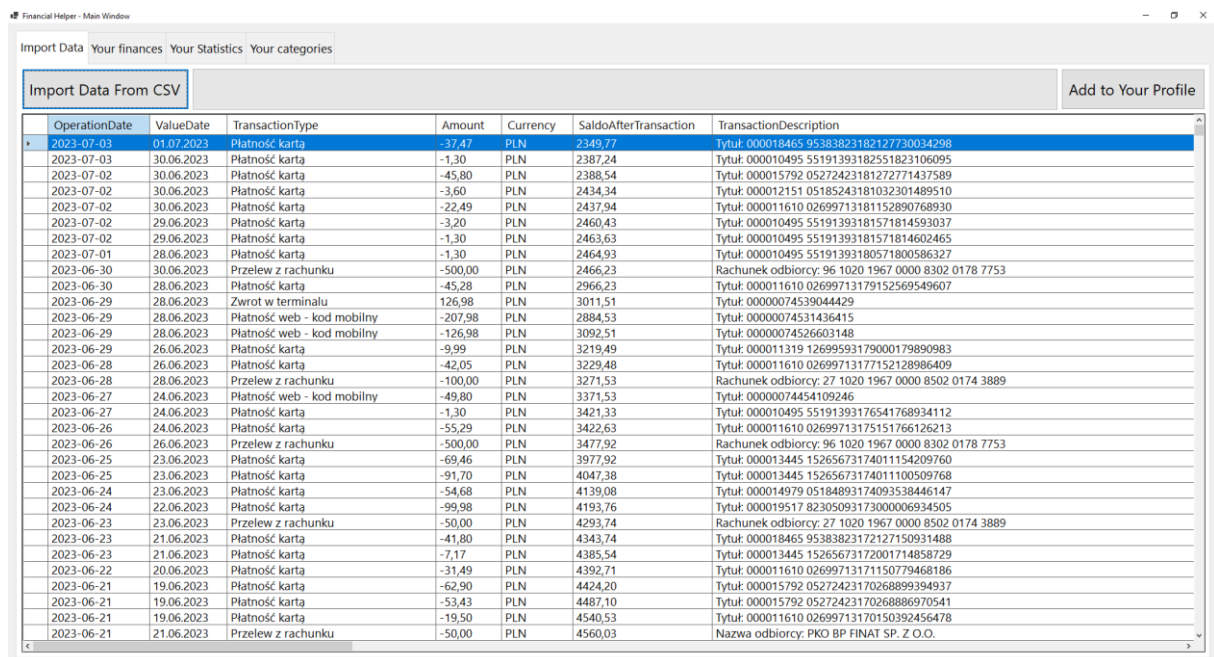
Po zalogowaniu pierwszym widokiem, który zobaczy użytkownik będzie podstrona „Import Data”. Na tej stronie użytkownik może zaimportować plik CSV za pomocą opcji „Import Data From CSV” i po poprawnym wczytaniu te dane zostaną wyświetlone na DataGridView. Następnie użytkownik może dodać te dane do swojego profile (dane, które już są w profilu użytkownika nie zostaną dodane podwójnie). Przykładowy wygląd okna jest pokazany na *Rysunek 4* a także przykładowy wygląd po zaimportowaniu danych z pliku pokazany jest na *Rysunek 5*.

Akademia im. Jakuba z Paradyża w Gorzowie Wielkopolskim

Wydział Techniczny



Rysunek 4. Podstrona Import Data (1)



Rysunek 5. Podstrona Import Data (2)

Na podstronie „Your finances” użytkownik może przeglądać wcześniej zaimportowane dane, za pomocą pola tekstowego do wyszukiwania wyszukać interesujące dane, a także za pomocą listbox’a przypisać wcześniej utworzone kategorie do konkretnego wiersza. Przykładowy wygląd tej podstrony pokazany jest na *Rysunek 6*. Przypisanie kategorii pokazane jest na *Rysunek 7*.

Akademia im. Jakuba z Paradyża w Gorzowie Wielkopolskim
Wydział Techniczny

Financial Helper - Main Window

Import Data Your finances Your Statistics Your categories

Search Text Search

Select Category	Id	Category	OperationDate	ValueDate	TransactionType	Amount	Currency	SaldoAfterTransaction	TransactionDescription
▼	1939	Wynagrodzenie	2023-07-03	01.07.2023	Płatność karta	-37,47	PLN	2349,77	Tytul: 000018465 9538382318212773003429
▼	1940		2023-07-03	30.06.2023	Płatność karta	-1,3	PLN	2387,24	Tytul: 000010495 5519139318255182310609
▼	1941		2023-07-02	30.06.2023	Płatność karta	-45,8	PLN	2388,54	Tytul: 000015792 0527242318127277143758
▼	1942		2023-07-02	30.06.2023	Płatność karta	-3,6	PLN	2434,34	Tytul: 000012151 0518524318103230148951
▼	1943		2023-07-02	30.06.2023	Płatność karta	-22,49	PLN	2437,94	Tytul: 000011610 0269971318115289076893
▼	1944		2023-07-02	29.06.2023	Płatność karta	-3,2	PLN	2460,43	Tytul: 000010495 5519139318157181459303
▼	1945		2023-07-02	29.06.2023	Płatność karta	-1,3	PLN	2463,63	Tytul: 000010495 5519139318157181460246
▼	1946		2023-07-01	28.06.2023	Płatność karta	-1,3	PLN	2464,93	Tytul: 000010495 5519139318057180058632
▼	1947		2023-06-30	30.06.2023	Przelew z rachunku	-500,0	PLN	2466,23	Rachunek odbiorcy: 96 1020 1967 0000 8302
▼	1948		2023-06-30	28.06.2023	Płatność karta	-45,28	PLN	2966,23	Tytul: 000011610 0269971317915256954960
▼	1949		2023-06-29	28.06.2023	Zwrot w terminalu	126,98	PLN	3011,51	Tytul: 00000074539044429
▼	1950		2023-06-29	28.06.2023	Płatność web - kod mobilny	-207,98	PLN	2884,53	Tytul: 00000074531436415
▼	1951		2023-06-29	28.06.2023	Płatność web - kod mobilny	-126,98	PLN	3092,51	Tytul: 00000074526603148
▼	1952		2023-06-29	26.06.2023	Płatność karta	-9,99	PLN	3219,49	Tytul: 000011319 1269959317900017989098
▼	1953		2023-06-28	26.06.2023	Płatność karta	-42,05	PLN	3229,48	Tytul: 000011610 02699713177152128986405
▼	1954		2023-06-28	28.06.2023	Przelew z rachunku	-100,0	PLN	3271,53	Rachunek odbiorcy: 27 1020 1967 0000 8502
▼	1955		2023-06-27	24.06.2023	Płatność web - kod mobilny	-49,8	PLN	3371,53	Tytul: 00000074454109246
▼	1956		2023-06-26	24.06.2023	Płatność karta	-1,3	PLN	3421,33	Tytul: 000010495 5519139317654176893411
▼	1957		2023-06-26	24.06.2023	Płatność karta	-55,29	PLN	3422,63	Tytul: 000011610 0269971317515176612621
▼	1958		2023-06-26	26.06.2023	Przelew z rachunku	-500,0	PLN	3477,92	Rachunek odbiorcy: 96 1020 1967 0000 8302
▼	1959		2023-06-25	23.06.2023	Płatność karta	-69,46	PLN	3977,92	Tytul: 000013445 1526567317401115420976
▼	1960		2023-06-25	23.06.2023	Płatność karta	-91,7	PLN	4047,38	Tytul: 000013445 1526567317401110050976
▼	1961		2023-06-24	23.06.2023	Płatność karta	-54,68	PLN	4139,08	Tytul: 000014979 0518489317409353844614
▼	1962		2023-06-24	22.06.2023	Płatność karta	-99,98	PLN	4193,76	Tytul: 000019517 8230509317300000693450
▼	1963		2023-06-23	23.06.2023	Przelew z rachunku	-50,0	PLN	4293,74	Rachunek odbiorcy: 27 1020 1967 0000 8502
▼	1964		2023-06-23	21.06.2023	Płatność karta	-41,8	PLN	4343,74	Tytul: 000018465 9538382317212715093148
▼	1965		2023-06-23	21.06.2023	Płatność karta	-7,17	PLN	4385,54	Tytul: 000015792 052724231702688993493
▼	1966		2023-06-22	20.06.2023	Płatność karta	-31,49	PLN	4392,71	Tytul: 000011610 0269971317115077946818
▼	1967		2023-06-21	19.06.2023	Płatność karta	-62,9	PLN	4424,2	Tytul: 000015792 052724231702688993493
▼	1968		2023-06-21	19.06.2023	Płatność karta	-53,43	PLN	4487,1	Tytul: 000015792 052724231702688993493
▼	1969		2023-06-21	19.06.2023	Płatność karta	-19,5	PLN	4540,53	Tytul: 000011610 0269971317015039245647
▼	1970		2023-06-21	21.06.2023	Przelew z rachunku	-50,0	PLN	4560,03	Nazwa odbiorcy: PKO BP FINAT SP. Z O.O.
▼	1971		2023-06-20	17.06.2023	Płatność web - kod mobilny	-1211,99	PLN	4610,03	Tytul: 00000074314191798

Rysunek 6. Podstrona Your finances (1)

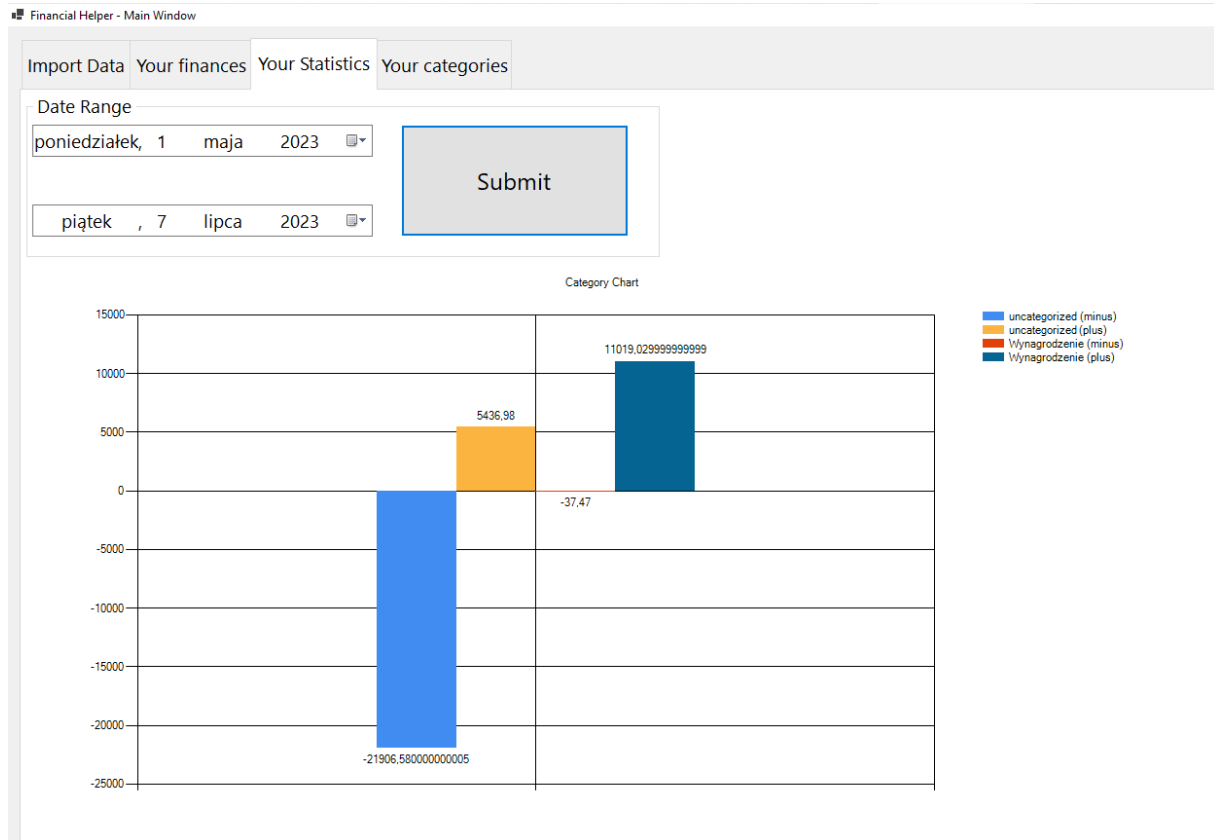
Financial Helper - Main Window

Import Data Your finances Your Statistics Your categories

Select Category	Id	Category	OperationDate
▼	1939	Wynagrodzenie	2023-07-03
▼	1940		2023-07-03
▼	1941		2023-07-02
▼	1942		2023-07-02
▼	1943		2023-07-02
▼	1944		2023-07-02

Rysunek 7. Podstrona Your finances (2)

Na podstronie Your statistics użytkownik może wybrać zakres dat od i do i po wybraniu opcji „Submit” zostanie wyświetlona statystyka wydatków i przychodu z wybranego zakresu. Przykładowy wygląd tej strony pokazany jest na **Rysunek 8**.



Rysunek 8. Podstrona Your Statistics

Na podstronie Your Categories użytkownik może przejrzeć swoje wcześniej utworzone kategorie, utworzyć nowe, a także zmodyfikować lub usunąć już istniejące. Przy zaznaczeniu kategorii w DataGridView zostanie wyświetlony dodatkowy formularz w którym można zmodyfikować dane zaznaczonej kategorii lub przy wybraniu opcji „Delete” usunąć zaznaczoną kategorię. Przykładowy wygląd tej podstrony pokazany jest na *Rysunek 9*.

Financial Helper - Main Window

Import Data | Your finances | Your Statistics | **Your categories**

Add New Category

Modify Category

	Id	Name	Commentary
▶	8	Wynagrodzenie	Wynagrodzenie

Rysunek 9. Podstrona Your Categories

6. Wnioski

Aplikacja została utworzona w technologii WinForms, ponieważ w takiej technologii było proponowane realizowanie wszystkich laboratoriów. Jednakże w trakcie realizowania tego projektu jak i na końcu nie porzuciło wyczucie, że lepiej byłoby realizować ten projekt w innej technologii, np., WPF, MAUI lub jako aplikację internetową ASP.NET + Angular/React itd.

W trakcie realizacji tego projektu napotkano zostało bardzo dużo problemów (widoczne po ilości linków w sekcji 7), na przykład takich jak brak dostępu kontrolki Chart w WinForms .NET 7, ale cele, które zostały zaplanowane zostały zrealizowane i można śmiało powiedzieć, że to narzędzie będzie wykorzystywane we własnym zakresie.

Także można dodać że aplikacja, ma przyszłość i może być rozwijana o różne nowe funkcjonalności.

7. Bibliografia

1. Źródła pomocnicze

- a. Intro to the CsvHelper Library for C#
[<https://www.youtube.com/watch?v=z3BwMlcGdhg>], dostęp: 07.07.2023
- b. How to Read Data From a CSV File in C# [<https://code-maze.com/csharp-read-data-from-csv-file/>], dostęp: 07.07.2023
- c. How to customize the tab page header size in WinForms TabControlAdv?
[<https://support.syncfusion.com/kb/article/9538/how-to-customize-the-tab-page-header-size-in-winforms-tabcontroladv>], dostęp: 07.07.2023
- d. C# OpenFileDialog [<https://www.c-sharpcorner.com/UploadFile/mahesh/openfiledialog-in-C-Sharp/>], dostęp: 07.07.2023
- e. Add a ComboBox and CheckBox Into the DataGridView in C#
[<https://www.c-sharpcorner.com/UploadFile/9f4ff8/add-combobox-and-checkbox-into-the-datagridview-in-C-Sharp/>], dostęp: 07.07.2023
- f. Styling WinForms DataGridView
[<https://www.youtube.com/watch?v=Y4WV1tQBW5I>], dostęp: 07.07.2023
- g. Chart Control in Windows Forms Application [<https://www.c-sharpcorner.com/UploadFile/1e050f/chart-control-in-windows-form-application/>],
dostęp: 07.07.2023
- h. WinForms Data Visualization [<https://github.com/kirsan31/winforms-datavisualization>], dostęp: 07.07.2023
- i. How to Secure Passwords with BCrypt.NET [<https://code-maze.com/dot-net-secure-passwords-bcrypt/>], dostęp: 07.07.2023

2. Napotkane problem i ich rozwiązania

- a. DataGridView AutoFit and Fill [<https://stackoverflow.com/questions/18666582/datagridview-autofit-and-fill>], dostęp: 07.07.2023
- b. Autosize Form containing TabControl [<https://stackoverflow.com/questions/16498366/autosize-form-containing-tabcontrol>], dostęp: 07.07.2023
- c. How can you make the form maximize to any computer screen in a Windows Forms application? [<https://stackoverflow.com/questions/2955061/how-can-you-make-the-form-maximize-to-any-computer-screen-in-a-windows-forms-app>], dostęp: 07.07.2023

- d. Windows – 1252 is not supported encoding name [<https://stackoverflow.com/questions/32471058/windows-1252-is-not-supported-encoding-name>], dostęp: 07.07.2023
- e. How can I filter a DataGridView? [<https://stackoverflow.com/questions/21845016/how-can-i-filter-a-datagridview>], dostęp: 07.07.2023
- f. How to enable DataGridView sorting when user clicks on the column header [<https://stackoverflow.com/questions/5553100/how-to-enable-datagridview-sorting-when-user-clicks-on-the-column-header>], dostęp: 07.07.2023
- g. Convert IList<T> to BindingList<T> [<https://stackoverflow.com/questions/14953461/convert-ilstt-to-bindinglistt>], dostęp: 07.07.2023
- h. How to Concat String in LINQ SQL Where clause? [<https://stackoverflow.com/questions/34302814/how-to-concat-string-in-linq-sql-where-clause>], dostęp: 07.07.2023
- i. DataGridView capturing user row selection [<https://stackoverflow.com/questions/1027360/datagridview-capturing-user-row-selection>], dostęp: 07.07.2023
- j. Cannot set values from a DataGridViewComboBoxColumn to a bound DataGridView [<https://stackoverflow.com/questions/68366296/cannot-set-values-from-a-datagridviewcomboboxcolumn-to-a-bound-datagridview>], dostęp: 07.07.2023
- k. Add items to combobox in an already existing datagridview comboBox column [<https://stackoverflow.com/questions/41612330/add-items-to-combobox-in-an-already-existing-datagridview-combobox-column>], dostęp: 07.07.2023
- l. DataGridView – Use DataPropertyName to show child element property [<https://stackoverflow.com/questions/14046830/datagridview-use-datapropertyname-to-show-child-element-property>], dostęp: 07.07.2023
- m. Entity Framework – Group By Sum [<https://stackoverflow.com/questions/32759955/entity-framework-group-by-sum>], dostęp: 07.07.2023
- n. Creating a chart in C# WinForms .NET 6 [<https://stackoverflow.com/questions/72251237/creating-a-chart-in-c-sharp-winforms-net-6>], dostęp: 07.07.2023

8. Spis ilustracji

Rysunek 1. Diagram klas kontekstu bazy danych, wersja podstawowa	6
Rysunek 2. Diagram klas kontekstu bazy danych, wersja rozszerzona	6
Rysunek 3. Okno Login Form.....	22
Rysunek 4. Podstrona Import Data (1).....	23
Rysunek 5. Podstrona Import Data (2).....	23
Rysunek 6. Podstrona Your finances (1).....	24
Rysunek 7. Podstrona Your finances (2).....	24
Rysunek 8. Podstrona Your Statistics	25
Rysunek 9. Podstrona Your Categories.....	26

9. Spis snippet'ów

Snippet 1. Polecenia Install-Package	3
Snippet 2. Kontekst bazy danych	4
Snippet 3. Encja "BankData"	4
Snippet 4. Encja "Category"	5
Snippet 5. Encja "User"	5
Snippet 6. Interfejs IBankDataService	7
Snippet 7. Interfejs ICategoryService	7
Snippet 8. Interfejs ICSVService	7
Snippet 9. Interfejs UserService.....	7
Snippet 10. Serwis BankDataService.....	8
Snippet 11. Serwis CategoryService	10
Snippet 12. Serwis CSVService	11
Snippet 13. Serwis UserService	12
Snippet 14. Model BankData	14
Snippet 15. Model CategoryChartData	15
Snippet 16. Login Form	15
Snippet 17. Main Form - Import page.....	17
Snippet 18. Main Form - Your Finances Page	18
Snippet 19. Main Form - Categories Page	19

Snippet 20. Main Form - Your Statistics Page.....	20
---	----