Spis treści

1. Wstęp

1.1. Problematyka i zakres pracy

1.2. Cele pracy

1.3. Przegląd literatury w dziedzinie komputerowej analizy danych finansowych

2. Zarządzanie budżetem oraz predykcja wydatków przy użyciu znanych metod analizy danych

2.1. Wprowadzenie

Z zagadnieniem zarządzania budżetem domowym spotyka się każdy, kto dysponuje środkami pieniężnymi. Oznacza ono analizę wszystkich poniesionych kosztów oraz planowanie przyszłych wydatków. Prawidłowe zarządzanie budżetem pozwala na świadome i przemyślane wydawanie pieniędzy oraz ułatwia oszczędzanie.

Na zarządzanie budżetem istnieje wiele sposobów. Najprostszym i najstarszym z nich jest prowadzenie dziennika dochodów i wydatków. Usprawnieniem tego procesu są liczne aplikacje ułatwiające prowadzenie budżetu domowego przez wykorzystanie komputera lub smartfona. Udostępniają one szereg funkcjonalności, które sprawiają, że rejestracja przychodów i wydatków staje się łatwiejsza. Samo rejestrowanie kosztów nie pozwala jednak na ich planowanie, przez co zarówno tradycyjne podejście do zarządzania budżetem, jak i istniejące aplikacje do tego służące nie są w pełni funkcjonalne.

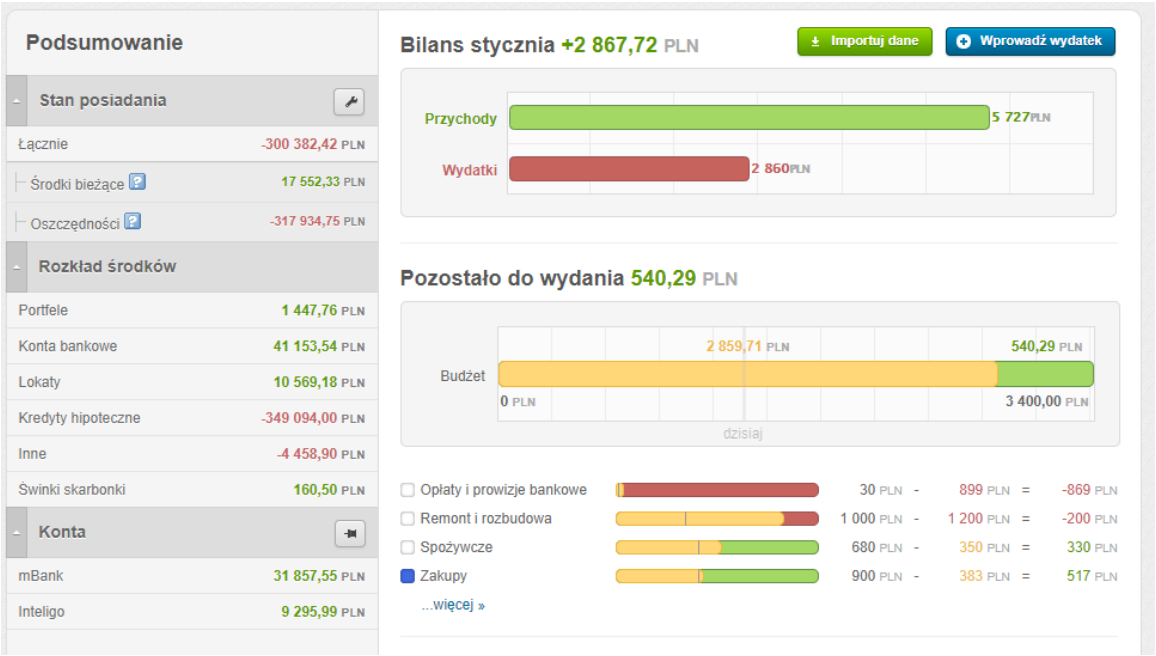
Powstała aplikacja na podstawie danych o przychodach i rozchodach wszystkich użytkowników pozwala na **predykcję**, czyli przewidywanie wydatków powiązanych z zakupem produktów lub usług, które niosą za sobą koszty utrzymania, np. samochodu.

2.2. Dostępne aplikacje do zarządzania budżetem

Na rynku jest dostępnych wiele aplikacji służących do zarządzania budżetem, zarówno płatnych, jak i darmowych. Poddane analizie zostały trzy aplikacje wykorzystywane w tym celu: **Kontomierz, YNAB** i **Spendee**.

Kontomierz (rys. 2.1) jest darmową aplikacją dostępną poprzez przeglądarkę internetową oraz aplikacje na systemy Android oraz iOS. Pobiera ona dane o wydatkach użytkownika

z historii transakcji na koncie bankowym. Aplikacja ta prezentuje użytkownikowi w przejrzysty sposób strukturę wydatków wraz z ich kategoriami, propozycje oszczędności oraz usług bankowych. Jest zintegrowana z systemami wielu polskich banków, dzięki czemu w szybki sposób umożliwia orientację w bieżącej sytuacji finansowej użytkownika. Jest to najpopularniejsza aplikacja do zarządzania budżetem na polskim rynku.

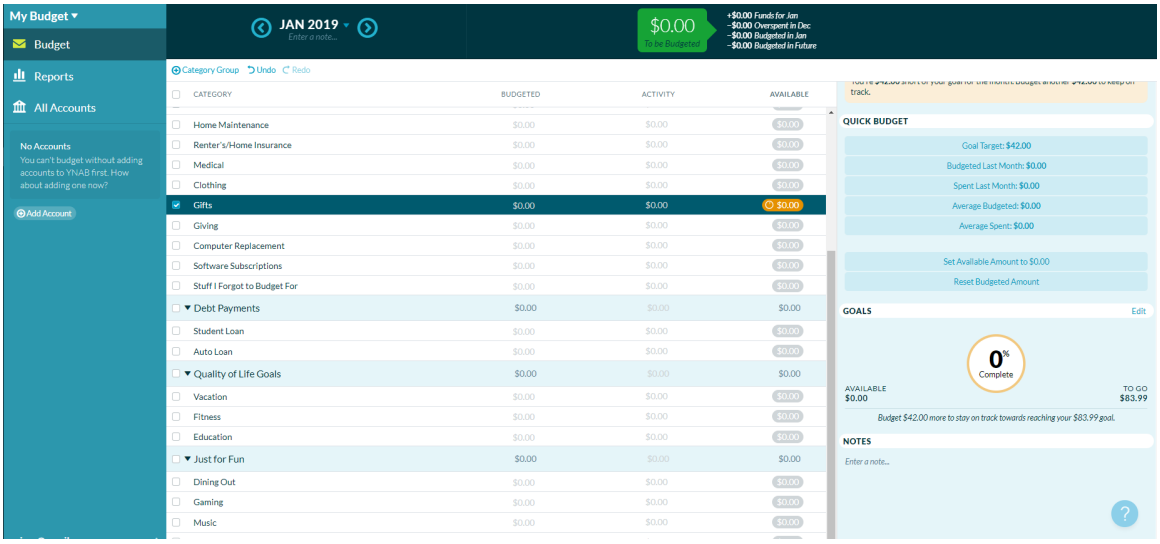


Rysunek 2.1. Interfejs aplikacji Kontomierz

YNAB - You Need A Budget (rys. 2.2) to aplikacja dostępna w języku angielskim, zarówno poprzez przeglądarkę internetową, jak i przez aplikacje mobilne, służąca do kompleksowej rejestracji i planowania budżetu. Podobnie jak Kontomierz, posiada możliwość importu danych z konta bankowego, jednak z racji tego, iż nie jest to polska aplikacja, nie ma możliwości importu danych z polskich banków.

YNAB jest jedną z bardziej rozbudowanych aplikacji dostępnych na rynku, aczkolwiek nie jest aplikacją darmową, dostępny jest jedynie miesięczny bezpłatny okres próbny. W skład jej funkcjonalności wchodzi automatyczna rejestracja wydatków i przychodów oraz kompleksowy system planowania budżetu. Wszystkie wolne środki jakie posiadamy przypisujemy do wybranych celów, którymi są planowane wydatki, opłaty stałe bądź oszczędności.

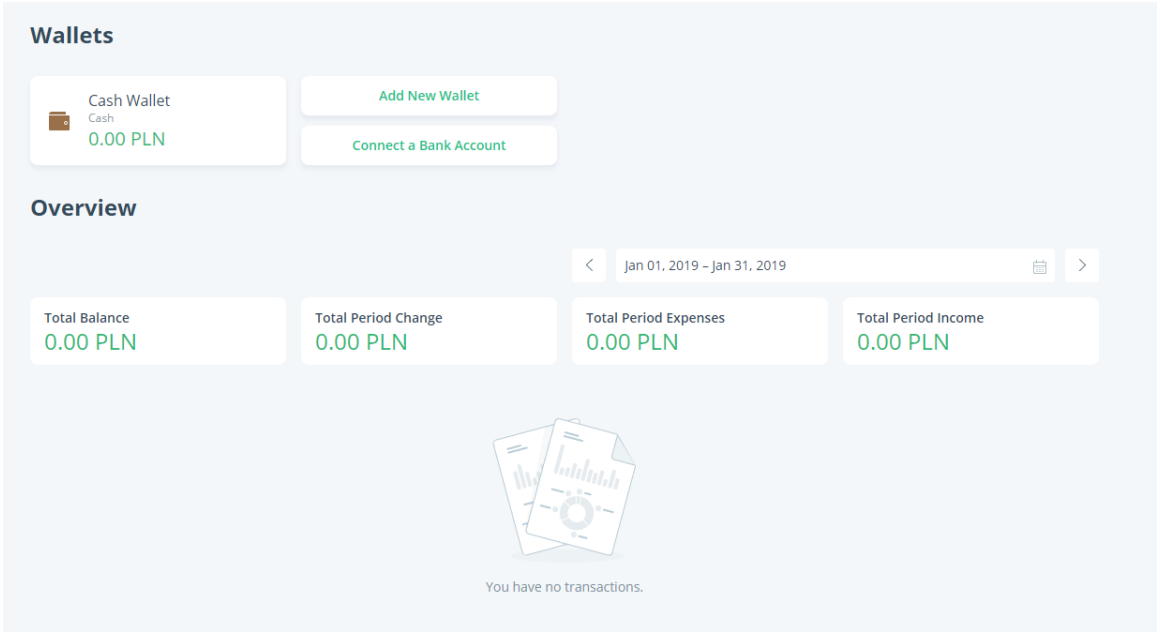
Spendee (rys. 2.3) jest aplikacją anglojęzyczną służącą do zarządzania budżetem. Jest dostępna zarówno w formie aplikacji webowej, jak i mobilnej. Aplikacja posiada wersję podstawową oraz dwie wersje płatne różniące się dostępnymi funkcjonalnościami. Pakiet



Rysunek 2.2. Interfejs aplikacji YNAB

podstawowy jest darmowy oraz oferuje możliwość wprowadzania i przeglądu wydatków oraz dochodów użytkownika.

Wersje płatne oferują znacznie więcej funkcjonalności. Aplikacja pozwala na automatyczny import danych z systemów bankowych oraz, w przeciwieństwie do YNAB, obsługuje banki z wielu krajów, w tym także z Polski. Wykupując wersję premium użytkownik dostaje również możliwość dzielenia wirtualnego portfela z innymi użytkownikami oraz automatyczną kategoryzację wydatków przy imporcie danych z systemu bankowego.



Rysunek 2.3. Interfejs aplikacji Spendee

2.3. Objaśnienie pojęć z dziedziny komputerowej analizy danych finansowych

Statystyka to nauka o metodach badań poświęconych liczbowo wyrażalnym właściwościom zbiorowości oraz wszelkie prace związane z gromadzeniem i opracowaniem masowych danych liczbowych.[1]

Analiza regresji jest statystyczną metodologią przewidywania wartości jednej lub więcej zmiennych odpowiedzi (zależnych) za pomocą zbioru predyktorów (czyli zmiennych niezależnych). Może być także użyta do oszacowania efektów jakie predyktory wywierają na odpowiedzi.[2]

Predykcja (prognozowanie) to przewidywanie jakie wartości przyjmie zmienna objaśniana przy zadanej wielkości zmiennej objaśniającej.[1]

2.4. Opis wybranych metod analizy danych

2.4.1. Wieloraka regresja liniowa

Niech $z_1, z_2, ..., z_k$ będą zbiorem k predyktorów, które potencjalnie wpływają na zmienną Y. Model regresji liniowej n-elementowej próbki:

$$Y_n = \beta_0 + \beta_1 z_{n1} + \beta_2 z_{n2} + \cdots + \beta_k z_{nk} + \epsilon_n$$

gdzie $\beta_i, i = 0, 1, ..., k$ są nieznanymi i ustalonymi współczynnikami regresji,
β_0 jest wyrazem wolnym,
ϵ jest błędem losowym.[2]

Dla zbioru n rekordów danych model regresji możemy zapisać w postaci macierzowej:

$$Y = X\beta + \epsilon$$

gdzie

$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$ - macierz zmiennych zależnych,

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1k} \\ 1 & x_{21} & x_{22} & \dots & x_{2k} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{n1} & x_{n2} & \dots & x_{nk} \end{bmatrix}$$ - macierz predyktorów,

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_n \end{bmatrix}$$ - macierz współczynników regresji,

$$\epsilon = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$ - macierz reszt. [3]

2.4.2. Regresja nieparametryczna

Regresja nieparametryczna, w odróżnieniu od regresji parametrycznej (np. 2.4.1), nie uwzględnia ścisłego matematycznego modelu regresji, ale próbuje oszacować postać zależności między predyktorami X oraz zmienną zależną Y. Dzięki tej cesze regresja nieparametryczna znajduje zastosowanie w przypadkach, w których postać zależności między predyktorami a zmienną zależną nie jest ściśle określona.[4] Przykładem regresji nieparametrycznej jest metoda **k najmniejszych kwadratów**. Polega ona na przybliżeniu wartości zmiennej zależnej jako średniej wartości k najbliższych rekordów znanych danych.

$$f(x) = \sum_{i=1}^{n} w_i(x) y_i$$

gdzie:

n - liczba próbek,

$$w_i(x) = \begin{cases} 1/k \; jeżeli \; x_i \; jest \; jednym \; z \; k \; najbliższych \; do \; x \; rekordów, \\ 0 \; w \; przeciwnym \; wypadku, \end{cases}$$

y_i - wartość i-tej zmiennej zależnej.[5]

Odległość rekordów definiuje się za pomocą metryk, np. metryki euklidesowej. Określa się ją jako:

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

, gdzie:

x, y - dwa zbiory danych, x_i, y_i - i-ta cecha zbioru x lub y.[6]

2.4.3. Metoda najmniejszych kwadratów

Metoda najmniejszych kwadratów jest metodą wyznaczenia współczynników regresji parametrycznej polegającą na znalezieniu takich ich wartości, aby zminimalizować rozrzut między wartościami rzeczywistymi i odpowiadającymi im wartościami teoretycznymi zmiennej, czyli zminimalizować sumę kwadratów reszt.[1]

W najlepszym wypadku różnica między wartościami rzeczywistymi i teoretycznymi wynosi 0, zatem reszta ϵ dla każdego rekordu danych jest równa 0. W takim wypadku model regresji można zapisać następująco:

$$Y = X\beta$$

Przekształcając powyższe równanie otrzymujemy:

$$Y - X\beta = 0$$

$$X^T(Y - X\beta) = 0$$

$$X^TY - X^TX\beta = 0$$

$$X^TX\beta = X^TY$$

Stąd macierz współczynników regresji wyznaczamy w następujący sposób:

$$\beta = (X^TX)^{-1}X^TY$$

gdzie

X - macierz predyktorów,

Y - macierz zmiennych zależnych.[3]

Przedstawione operacje na macierzach X^T i X^{-1} to odpowiednio *transpozycja macierzy* oraz *odwrócenie macierzy*.

2.5. Wykorzystanie metod analizy danych do predykcji wydatków

Predykcja wydatków w stworzonej aplikacji wiąże się z tzw. **wydatkami głównymi**. Są to wydatki, które generują dodatkowe, występujące przez kilka następnych miesięcy, koszta. Użytkownik, po wprowadzeniu kategorii oraz wielkości wydatku głównego, na podstawie danych o wydatkach wszystkich użytkowników, dostaje predykcję kosztów związanych ze wspomnianym wydatkiem w miesiącach po nim następujących.

W niniejszej pracy użyta została parametryczna regresja wieloraka (2.4.1) wraz z metodą najmniejszych kwadratów (2.4.3). Metody te zostały wybrane, gdyż zbiór danych, na których dokonujemy predykcji jest ściśle ustrukturyzowany oraz możliwe jest określenie matematycznego modelu opisującego zależności w nim występujące.

Model regresji skonstruowany na potrzeby predykcji posiada trzy predyktory: miesięczny dochód użytkownika, wielkość wydatku dla którego wykonywana jest predykcja oraz suma wydatków z prognozowanej kategorii. Współczynniki regresji wyznaczane są przy pomocy metody najmniejszych kwadratów opisanej w rozdziale 2.4.3. Dla prognozy na każdy kolejny miesiąc tworzony jest model regresji, koszta w każdym miesiącu obliczane są przy użyciu innych współczynników regresji.

Dostępne na rynku aplikacje do zarządzania budżetem zapewniają kompleksowe rozwiązania służące do rejestracji wydatków i dochodów oraz planowania budżetu. Ich funkcjonalność może być wzbogacona o aspekt prognozowania wydatków. Mógłby on być szczególnie przydatny przy planowaniu przyszłych kosztów oraz ich integracji z istniejącym budżetem.

3. Technologie użyte w projekcie

System stworzony na potrzeby niniejszej pracy składa się z dwóch komponentów: warstwy logiki (ang. *backend*) oraz warstwy interfejsu użytkownika (ang. *frontend*).
Obecnie istnieje na rynku wiele technologii umożliwiających realizację warstwy backendu, jak np. Spring (Java), ASP .NET (C#), ASP .NET Core (C#). Technologie te różnią się wymaganiami oraz dostępnymi narzędziami.
Warstwa frontendu została zrealizowana w formie aplikacji mobilnej. Aplikacje te można podzielić na trzy kategorie:

- Natywne - zbudowane dla konkretnej platformy i napisane w języku dla niej odpowiednim, np. Swift dla iOS lub Kotlin dla Androida. Tak wykonane aplikacje cechują się szybkością oraz dostępem do funkcji urządzenia, takich jak akcelerometr czy czytnik linii papilarnych. Aplikacja natywna jest powiązana z konkretnym systemem operacyjnym, przez co proces tworzenia takowej dla różnych platform wiąże się z pisaniem odrębnych aplikacji.
- Webowe - dostęp do nich odbywa się poprzez przeglądarkę internetową. Nie mają dostępu do urządzenia na takim poziomie, jak aplikacje natywne, są jednak niezależne od systemu operacyjnego, przez co mniej kosztowne w produkcji. Popularne technologie tworzenia aplikacji webowych to np. Ruby on Rails (Ruby), Angular (TypeScript).
- Hybrydowe - połączenie aplikacji natywnej i webowej, posiadają zalety obu kategorii, nie są jednak tak szybkie, jak natywne. Ich interfejs stworzony jest w formie aplikacji webowej i jest interpretowany przez natywną aplikację dla konkretnego systemu. Pozwala to na tworzenie aplikacji, do których nie jest wymagana przeglądarka internetowa, posiadających dostęp do urządzenia na poziomie aplikacji natywnej. Aplikacje hybrydowe tworzone są przy pomocy takich technologii jak Xamarin Forms (C#) i React Native (JavaScript).

3.1. Języki programowania

Język **Java** jest współbieżnym, opartym na klasach i zorientowanym obiektowo językiem ogólnego zastosowania. Jest zaprojektowany tak, aby być prostym i sprawdzonym językiem.

Zapewnia automatyczne zarządzanie pamięcią przy użyciu odśmiecacza (ang. *garbage collector*). Kod napisany w Javie jest kompilowany do tzw. kodu bajtowego Javy (ang. *Java bytecode*), który jest interpretowany przez maszynę wirtualną Javy (ang. *Java Virtual Machine, JVM*), dzięki czemu jest wieloplatformowy (ang. *cross-platform*).[7]

```
class Program {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

Listing 3.1. Kod w języku Java wyświetlający napis "Hello, World".

C# jest prostym, nowoczesnym i zorientowanym obiektowo językiem programowania, wywodzącym się z języka C. Posiada on szereg funkcjonalności usprawniających proces wytwarzania oprogramowania, takimi jak: odśmiecanie (ang. *garbage collection*), bezpieczeństwo typologiczne (ang. *type safety*) czy obsługa wyjątków (ang. *exception handling*).[8]

```
using System;

public static class Program
{
    public static void Main()
    {
        Console.WriteLine("Hello, World!");
    }
}
```

Listing 3.2. Kod w języku C# wyświetlający napis "Hello, World".

JavaScript jest interpretowanym językiem programowania bez ścisłej kontroli typów posiadającego możliwości języka zorientowanego obiektowo. Syntaktycznie jest podobny do języków C lub C++.[9] Nadzbiorem języka JavaScript jest **TypeScript**, który pozwala na typowanie statyczne.

```
document.write('Hello, world!')
```

Listing 3.3. Kod w języku JavaScript wyświetlający napis "Hello, World".

Ruby jest językiem całkowicie zorientowanym obiektowo, co oznacza że każda wartość jest obiektem. Jest dynamicznym językiem z bogatym zasobem bibliotek. Podobny jest do takich języków jak Lisp, Smalltalk czy Perl.[10]

```
puts 'Hello, World!'
```

Listing 3.4. Kod w języku Ruby wyświetlający napis "Hello, World".

Swift to język programowania ogólnego użytku stworzony przez firmę Apple Inc. Jest zaprojektowany jako następca języków wywodzących się od C (C, C++, Objective-C) o porównywalnej wydajności. Jest to główny język tworzenia aplikacji mobilnych na system iOS.[11]

```
print("Hello, World")
```

Listing 3.5. Kod w języku Swift wyświetlający napis "Hello, World".

Kotlin jest statycznie typowanym, wieloplatformowym i darmowym językiem programowania ogólnego użytku rozwijanym przez JetBrains.[12] Kotlin jest oficjalnie wspierany przez Google jako język tworzenia aplikacji mobilnych na system Android.[13]

```
package hello

fun main() {
    println("Hello World")
}
```

Listing 3.6. Kod w języku Kotlin wyświetlający napis "Hello, World".

3.2. Platformy programistyczne (frameworki)

Spring Framework (rys. 3.1) jest platformą ułatwiającą tworzenie aplikacji z użyciem technologii Java Enterprise Edition. Spring składa się z wielu modułów, u podstawy posiadających rozbudowane mechanizmy konfiguracji i wstrzykiwania zależności (ang. *dependency injection*). Zapewnia wsparcie dla różnych architektur aplikacji.[14] Dzięki możliwości uruchomienia przy pomocy JVM jest technologią wieloplatformową.

.NET (rys. 3.2) jest darmową, otwartoźródłową platformą programistyczną służącą do wytwarzania różnych typów aplikacji. Platforma .NET jest dostępna dla języków C#, F# oraz

Rysunek 3.1. Logo Spring Framework (https://spring.io/img/spring-by-pivotal.png)

Visual Basic. Zaletą platformy .NET jest bogaty zasób bibliotek dostępnych dla wszystkich frameworków wchodzących w jej skład:

- **.NET Standard** jest wspólnym dla wszystkich platform .NET zestawem **interfejsów programowania aplikacji (ang. *application programming interface, API*)** i bibliotek. Ułatwia to pracę z różnymi frameworkami .NET poprzez zastosowanie tych samych narzędzi.
- **.NET Core** jest wieloplatformowym frameworkiem używanym do tworzenia stron internetowych, serwerów lub aplikacji konsolowych na systemy Windows, Linux i macOS.
- **.NET Framework** jest implementacją platformy .NET wymagającą do uruchomienia systemu Windows i przystosowaną do niego. Dzięki temu wzbogacona jest o biblioteki specyficzne dla tego systemu, jak na przykład narzędzia pozwalające na dostęp do rejestru systemu Windows.
- **Xamarin Forms** implementuje platformę .NET i pozwala na tworzenie aplikacji mobilnych na systemy Android, iOS oraz Windows Phone współdzielących w dużym stopniu swój kod oraz zachowujących wygląd aplikacji natywnych dla każdego systemu.[15]

Rysunek 3.2. Logo .NET (https://docs.microsoft.com/en-us/dotnet/images/hub/net.svg)

ASP .NET i ASP .NET Core to technologie pozwalające na budowę nowoczesnych aplikacji internetowych i usług sieciowych. Bazują one odpowiednio na .NET Framework

i .NET Core, co dyktuje ich dostępność na różnych systemach operacyjnych oraz dostępne biblioteki.

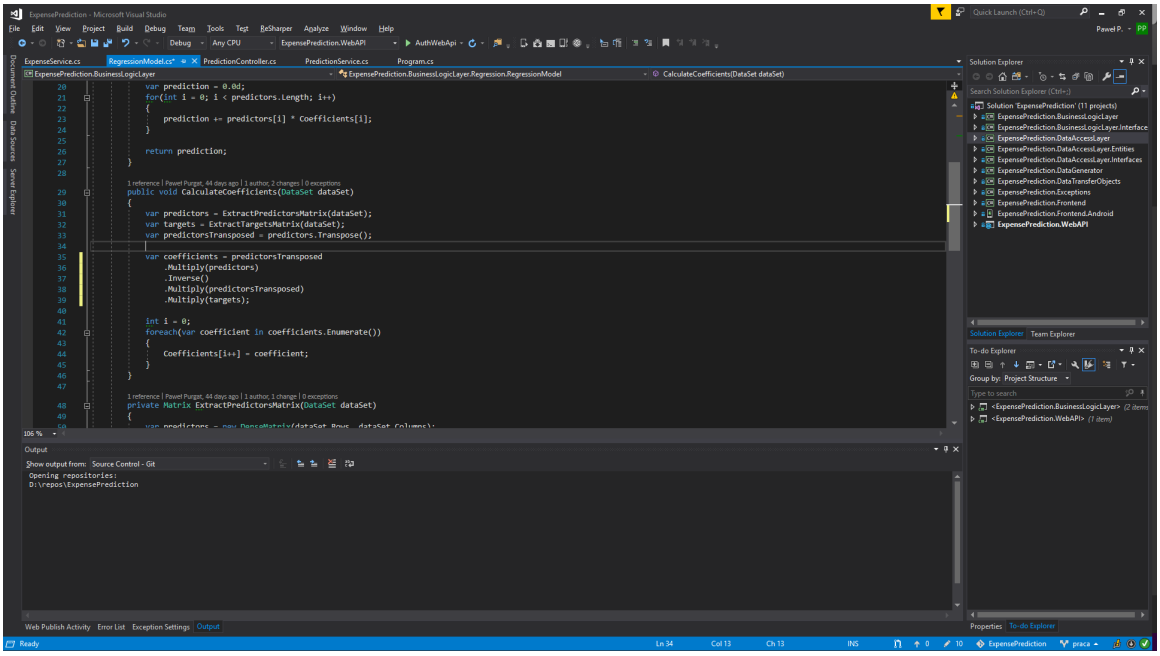
Rysunek 3.3. Logo ASP .NET (https://secure.gravatar.com/avatar/46c8189d84092927e2a78b63c37e7734)

System wykonany na potrzeby niniejszej pracy stworzony został przy użyciu platform ASP .NET Core (backend) oraz Xamarin Forms (frontend). Obie te platformy wykorzystują język C# oraz dysponują zestawem narzędzi zapewnionym przez .NET Standard. Umożliwia to współdzielenie części kodu między tymi dwiema platformami oraz usprawnia proces wytwarzania oprogramowania.
Zastosowanie Xamarin Forms w stworzeniu warstwy interfejsu użytkownika umożliwia łatwe rozszerzenie aplikacji aby możliwe było jej uruchomienie na systemach mobilnych innych niż Android.

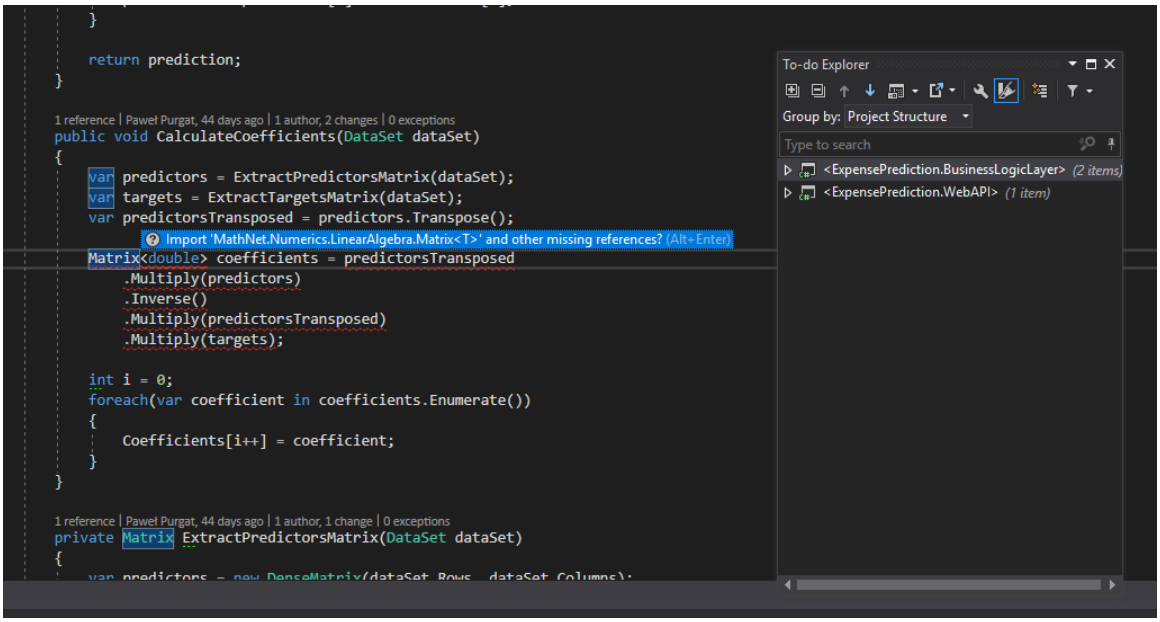
3.3. Narzędzia i wzorce projektowe

Microsoft Visual Studio (rys. 3.4) jest to profesjonalne środowisko programistyczne (ang. *integrated development environment, IDE*) stworzone przez firmę Microsoft Corporation, służące do tworzenia aplikacji desktopowych, mobilnych oraz sieciowych. Dzięki dostępności zaawansowanych narzędzi usprawnia proces programowania.[16]

ReSharper (rys. 3.5) jest narzędziem rozszerzającym środowisko Microsoft Visual Studio autorstwa JestBrains. Usprawnia ono proces wytwarzania oprogramowania poprzez automatyzację wielu procesów związanych z pisaniem wysokiej jakości kodu. Zapewnia ciągłą analizę kodu, co ułatwia wykrycie wielu błędów.[17]



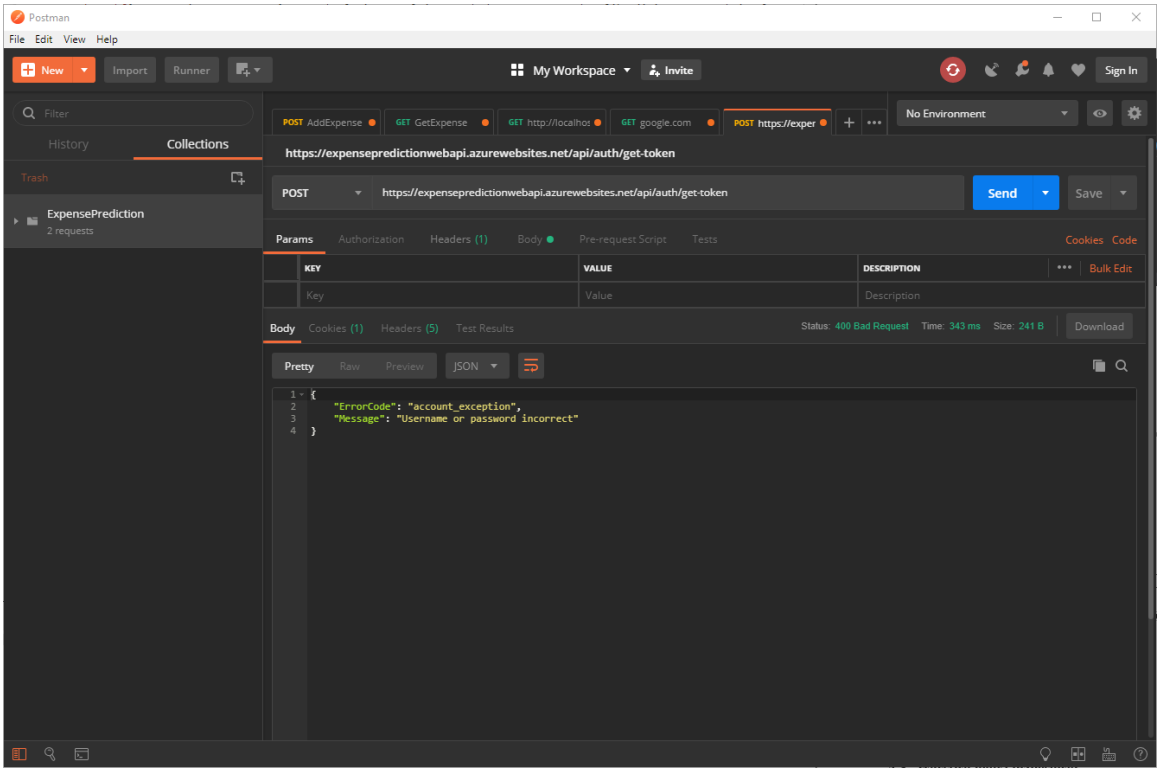
Rysunek 3.4. Interfejs środowiska Microsoft Visual Studio



Rysunek 3.5. Przykładowe funkcjonalności rozszerzenia ReSharper

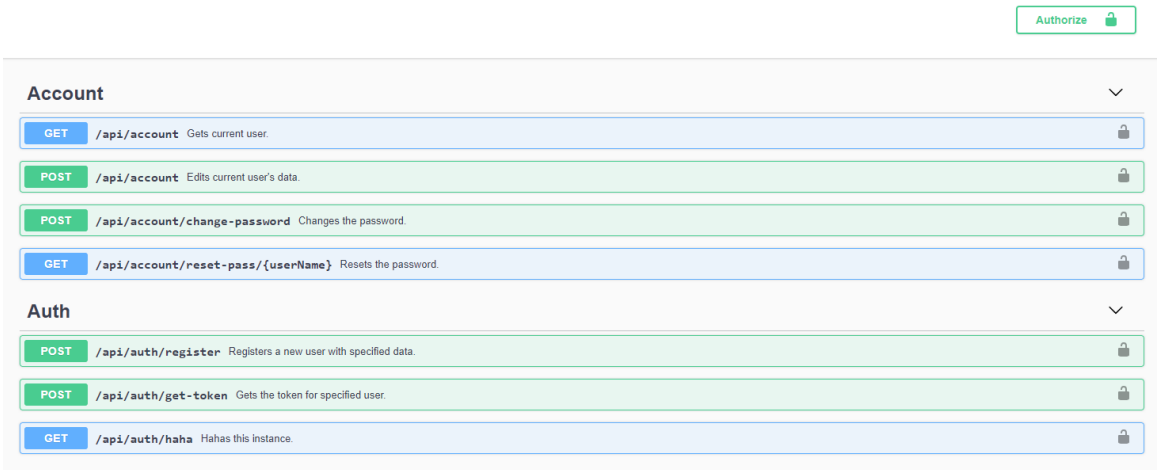
Postman to aplikacja zapewniająca interfejs użytkownika służący do tworzenia i wysyłania zapytań do aplikacji internetowej. Zawiera ona kompleksowe narzędzia umożliwiające przegląd i analizę zapytań HTTP. [18]

Swagger (rys. 3.7) jest narzędziem służącym do automatycznej dokumentacji kodu oraz zapewniającym konfigurowalny graficzny interfejs obrazujący stworzone API. Z poziomu



Rysunek 3.6. Interfejs aplikacji Postman

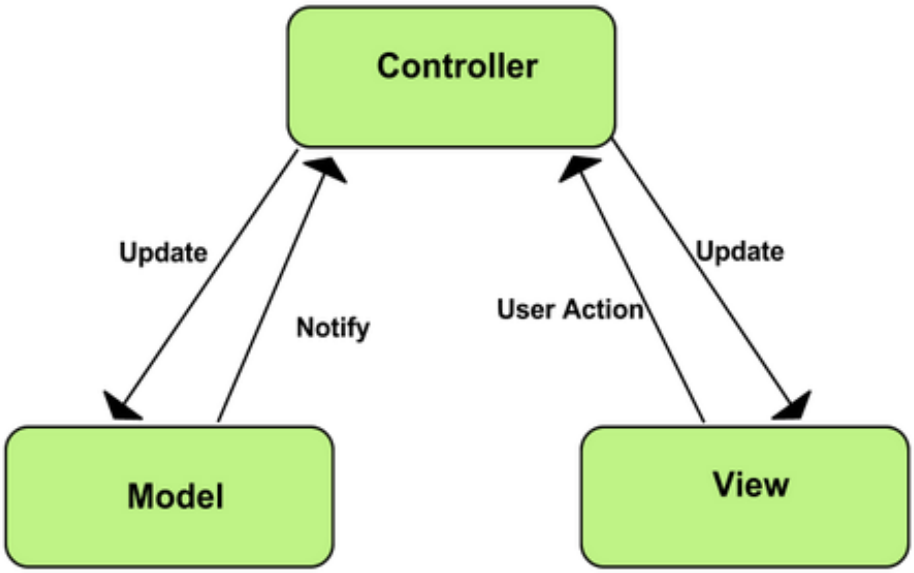
wygenerowanego interfejsu możliwe jest proste konstruowanie zapytań zgodnych ze stworzoną aplikacją.[19]



Rysunek 3.7. Interfejs wygenerowany przy użyciu narzędzia Swagger

REST (Representational State Transfer) jest stylem projektowania systemów. Nie jest on standardem, lecz zbiorem reguł, takich jak bezstanowość, relacja klient - serwer oraz jednolity interfejs.[20]

MVC (Model-View-Controller) (rys. 3.8) to wzorzec projektowy polegający na podzieleniu projektu aplikacji na warstwy modelu danych, widoku, czyli interfejsu użytkownika oraz kontrolera, który reaguje na akcje wykonane przez użytkownika w warstwie widoku oraz odczytuje i modyfikuje dane z warstwy modelu.[21]



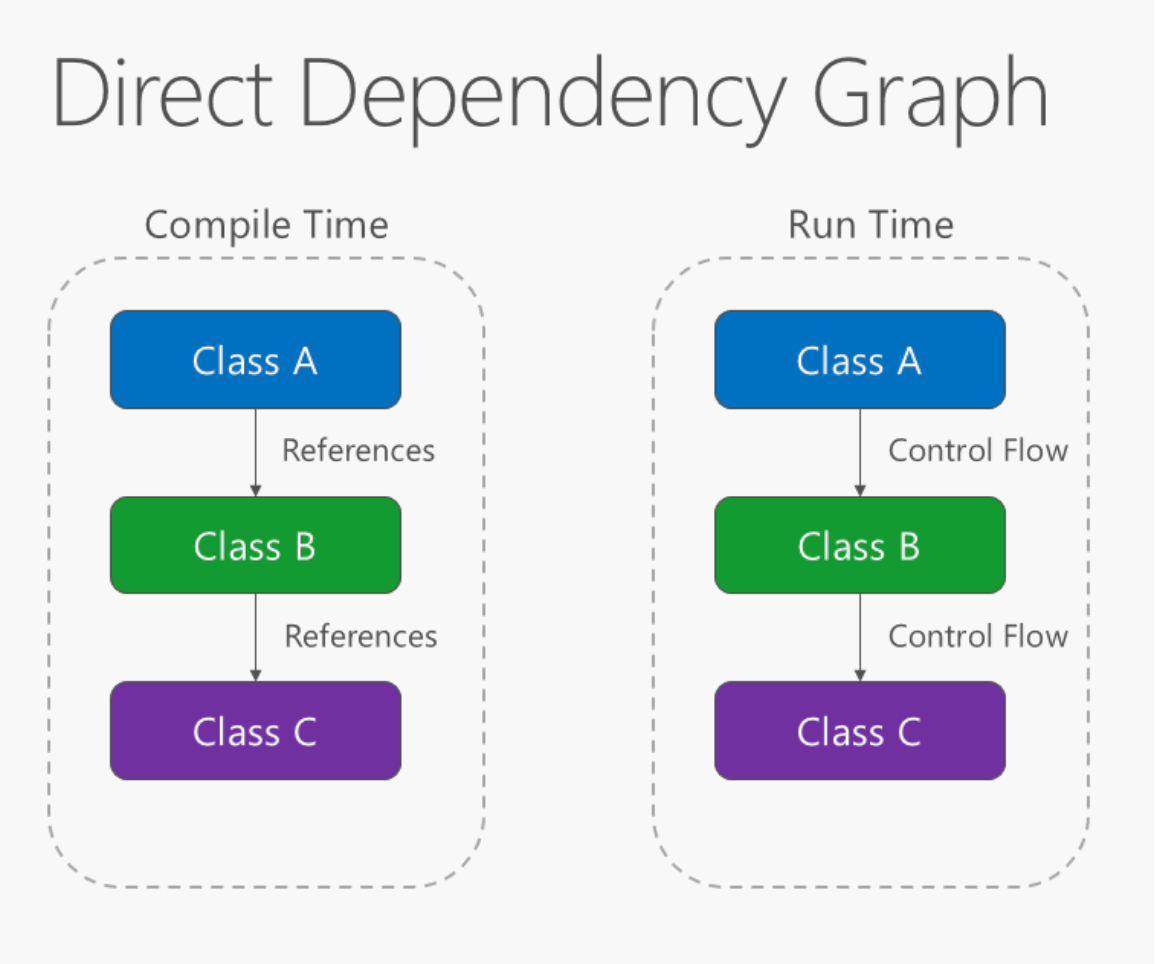
Rysunek 3.8. Diagram przedstawiający wzorzec MVC (https://developer.chrome.com/static/images/mvc.png)

Odwrócenie sterowania (ang. *inversion of control*) jest techniką pozwalającą na skonstruowanie zależności w aplikacji zgodnie z kierunkiem abstrakcji, zamiast implementacji. W wielu aplikacjach zależności między klasami skonstruowane są na podstawie kolejności ich wykonania (rys. 3.9). Odwrócenie sterowania polega na skonstruowaniu zależności w taki sposób, aby szczegółowa implementacja zależała od abstrakcyjnego komponentu (rys. 3.10).[22]

Wstrzykiwanie zależności (ang. *dependency injection*) jest wzorcem projektowym, który pozwala na osiągnięcie odwrócenia sterowania pomiędzy klasą a jej zależnościami poprzez przeniesienie odpowiedzialności za wybór implementacji abstrakcyjnego elementu na zewnętrzny kontener.[23]

Entity Framework Core jest narzędziem realizującym mapowanie relacyjno-obiektowe (ang. *object-relational mapping*) i oferującym możliwość odczytu i modyfikacji danych znajdujących się w bazie danych za pomocą klas reprezentujących encje w tej bazie.[24]

Language Integrated Query (LINQ) to zbiór technologii których zadaniem jest zintegrowanie możliwości wykonywania zapytań na danych z językiem C#. Zapewnia on



Rysunek 3.9. Diagram przedstawiający zależność bezpośrednią (https://docs.microsoft.com/en-us/dotnet/standard/ modern-web-apps-azure-architecture/media/image4-1.png)

jednolitą składnię tworzenia zapytań dla wielu źródeł danych, jak np. SQL, XML.[25]

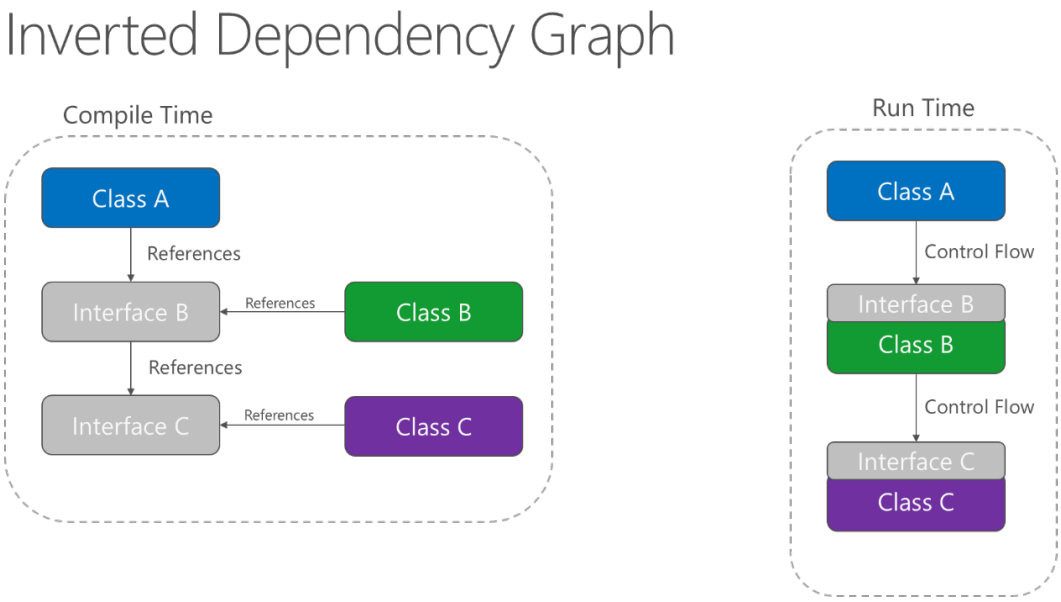
Przyjmując, że kolekcja *Users* i bazodanowa tabela o tej samej nazwie odnoszą się do tego samego zbioru danych, zapytanie LINQ (listing 3.7) oraz zapytanie SQL (listing 3.8) zwróciłyby taki sam wynik.

```
Users
    .Where(u => u.Name == "NameA")
    .Select(u => u.SecondName);
```

Listing 3.7. Zapytanie LINQ.

```
SELECT SecondName
FROM Users
WHERE FirstName = 'NameA'
```

Listing 3.8. Zapytanie SQL.



Rysunek 3.10. Diagram przedstawiający zależność odwróconą (https://docs.microsoft.com/en-us/dotnet/standard/ modern-web-apps-azure-architecture/media/image4-2.png)

3.4. Biblioteki

Dla środowiska .NET istnieje wiele zewnętrznych bibliotek. Dostępne są one poprzez system zarządzania pakietami NuGet. Narzędzie to pozwala na tworzenie oraz używanie bibliotek znajdujących się w repozytorium. W większości przypadków pakiety pobierane są z globalnego repozytorium dostępnego pod adresem `https://www.nuget.org`, jednakże istnieje możliwość tworzenia własnych repozytoriów i konfiguracji klienta NuGet do współpracy z nimi. W niniejszym podrozdziale opisane zostały jedynie biblioteki niewchodzące w skład frameworków Xamarin czy ASP .NET Core.

AutoMapper jest biblioteką umożliwiającą proste mapowanie jednego obiektu na drugi.[26] Biblioteka ta pozwala na wzajemną konwersję dwóch typów poprzez przypisanie polom jednego obiektu wartości pól drugiego o takich samych nazwach. Taki zabieg jest konieczny np. przy mapowaniu obiektu klasy encyjnej na obiekt transferu danych (ang. *data transfer object, DTO*).

Jeżeli pola obu typów mają takie same nazwy i nie wymagane jest żadne mapowanie między polami o różnych nazwach, definicja mapy sprowadza się do zdefiniowania typów, między jakimi ma powstać mapa (listing 3.9).

```
mapperConfiguration.CreateMap<User, UserDto>();
```

```
mapperConfiguration.CreateMap<Expense, ExpenseDto>();
```

Listing 3.9. Definiowanie mapowania między dwoma typami.

Przy tak zdefiniowanej mapie proces mapowania obiektów przebiega w sposób przedstawiony na listingu 3.10.

```
User user = GetUser();
UserDto userDto = mapper.Map<UserDto>(user);
```

Listing 3.10. Mapowanie jednego obiektu na drugi.

MathNet.Numerics jest biblioteką dostarczającą metody i algorytmy służące do wykonywania obliczeń i operacji numerycznych, m.in. algebry liniowej, modeli prawdopodobieństwa, interpolacji czy zagadnień optymalizacji.[27] W niniejszym projekcie biblioteka ta została wykorzystana w zakresie wykonywania operacji na macierzach (listing 3.11)

```
var predictors = ExtractPredictorsMatrix(dataSet);
var targets = ExtractTargetsMatrix(dataSet);
var predictorsTransposed = predictors.Transpose();

var coefficients = predictorsTransposed
.Multiply(predictors)
.Inverse()
.Multiply(predictorsTransposed)
.Multiply(targets);
```

Listing 3.11. Operacje na macierzach przy użyciu biblioteki MathNet.Numerics.

Newtonsoft.Json to biblioteka służącą do obsługi danych w formacie JSON. Jest to wydajne narzędzie pozwalające między innymi na serializację i deserializację obiektów, stosowanie LINQ do ekstrakcji danych z formatu JSON oraz konwersję między JSON a XML.[28] Biblioteka ta jest najczęściej pobieranym pakietem z globalnego repozytorium NuGet.[29]

4. Dokumentacja techniczna aplikacji do zarządzania budżetem i predykcji wydatków

4.1. Opis wymagań funkcjonalnych aplikacji

Wymagania funkcjonalne, które stworzony system ma spełniać uwzględniają czynności z zakresu zarządzania własnymi danymi, rejestracji przychodów i wydatków oraz predykcji wydatków:

- **rejestracja** - utworzenie nowego konta przez niezalogowanego użytkownika,
- **logowanie** - proces generowania tokenu pozwalającego na dostęp do aplikacji ,
- **wyświetlanie danych użytkownika** - dostęp do podstawowych danych użytkownika, takich jak imię, nazwisko lub numer telefonu,
- **edycja danych użytkownika** - edycja podstawowych danych użytkownika,
- **zmiana hasła użytkownika** - zmiana hasła aktualnie zalogowanego użytkownika,
- **wyświetlenie listy wydatków** - dostęp do listy wszystkich wydatków aktualnie zalogowanego użytkownika z określonego okresu,
- **wyświetlenie danych wydatku** - dostęp do szczegółów wybranego wydatku,
- **dodanie wydatku** - dodanie nowego wydatku dla aktualnie zalogowanego użytkownika,
- **edycja wydatku** - zmiana kwoty, daty, kategorii i opisu istniejącego wydatku,
- **usunięcie wydatku** - usunięcie wybranego wydatku z listy wydatków aktualnie zalogowanego użytkownika,
- **wyświetlenie listy przychodów** - dostęp do listy wszystkich przychodów aktualnie zalogowanego użytkownika z określonego okresu,
- **wyświetlenie danych przychodu** - dostęp do szczegółów wybranego przychodu,
- **dodanie przychodu** - dodanie nowego przychodu dla aktualnie zalogowanego użytkownika,
- **edycja przychodu** - zmiana kwoty, daty, kategorii i opisu istniejącego przychodu,

- **usunięcie przychodu** - usunięcie wybranego przychodu z listy przychodów aktualnie zalogowanego użytkownika,
- **predykcja wydatków** - wykonanie prognozy wielkości wydatków powiązanych z wydatkiem głównym na nadchodzące miesiące.

4.2. Opis wymagań niefunkcjonalnych aplikacji

Wymagania niefunkcjonalne uwzględniają ograniczenia systemu z zakresu bezpieczeństwa lub optymalizacji:

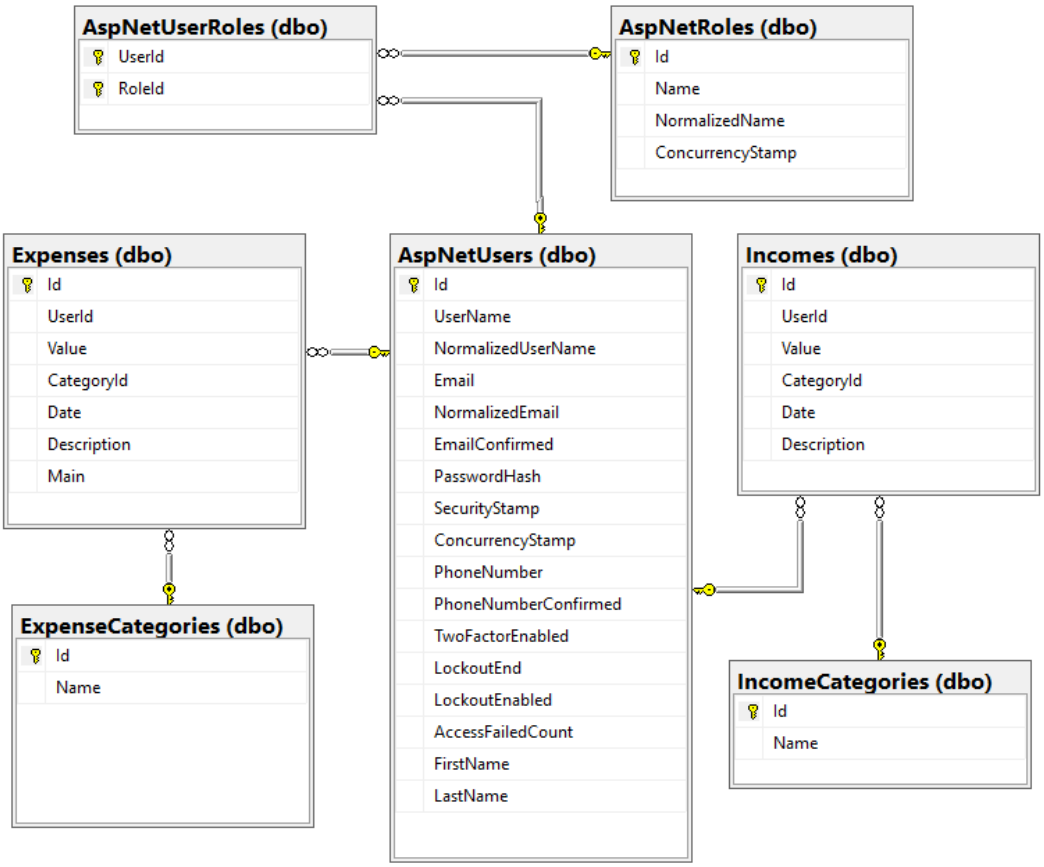
- **bezpieczne hasło** - nałożenie ograniczeń, które ciąg znaków musi spełniać, aby mógł być użyty jako hasło: minimalna długość to osiem znaków, ciąg musi zawierać co najmniej jedną małą i dużą literę, liczbę oraz znak specjalny,
- **dostęp do systemu dla uwierzytelnionych użytkowników** - możliwość skorzystania z funkcjonalności systemu (z kilkoma wyjątkami) jest dostępna jedynie dla uwierzytelnionych użytkowników,
- **dostęp do funkcjonalności ograniczony rolami** - dostęp do funkcjonalności jest ograniczony poprzez role przypisane do konta użytkownika, np. edycję kategorii wykonać może jedynie użytkownik, który posiada rolę "admin",
- **użycie tokenu** - wykorzystanie do uwierzytelniania i autoryzacji tokenu, który przechowuje informacje na temat tożsamości użytkownika oraz jego ról po stronie aplikacji klienckiej,
- **przechowywanie hasła w postaci jego skrótu** - wykorzystanie funkcji skrótu (ang. *hash function*), aby nie przechowywać hasła w bazie danych w postaci jawnej,
- **tworzenie modelu regresji przy starcie aplikacji** - wyznaczenie współczynników regresji przed skorzystaniem z funkcjonalności predykcji przez użytkownika, gdyż operacja ta jest czasochłonna przy dużej ilości danych,
- **obsługa błędów** - wyświetlenie użytkownikowi komunikatu z opisem błędu w razie jego wystąpienia.

4.3. Warstwa modelu danych

Na warstwę modelu danych składa się baza danych, klasy encyjne oraz klasy zapewniające podstawowe operacje na danych.

Dane w bazie danych (Rys. 4.1) podzielone są na siedem tabel:

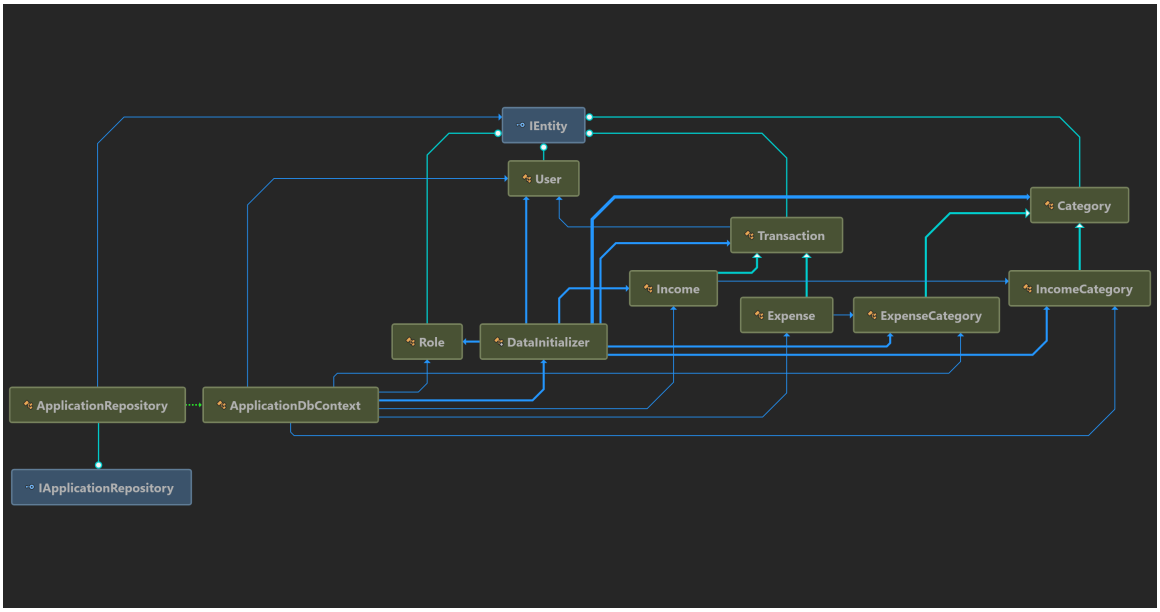
- **AspNetUsers** - zawiera dane o użytkownikach. Struktura tabeli wynika z mapowania wbudowanych w ASP.NET Core struktur służących do zarządzania użytkownikami.
- **AspNetRoles** - tabela zawierająca zdefiniowane w systemie role użytkowników.
- **AspNetUserRoles** - tabela definiująca powiązania między użytkownikami a ich rolami.
- **Incomes** - tabela zawierająca przychody wszystkich użytkowników. Jeden rekord zawiera takie dane jak użytkownik, do którego dany przychód należy, kwota przychodu, jego kategoria, data i opis.
- **IncomeCategories** - tabela z definicjami kategorii przychodów.
- **Expenses** - tabela zawierająca wydatki użytkowników. Zawiera ona koszt, kategorię, datę oraz opis wydatku, użytkownika, do którego należy oraz flagę określającą czy jest to główny wydatek.
- **ExpenseCategories** - tabela definiująca kategorie wydatków.



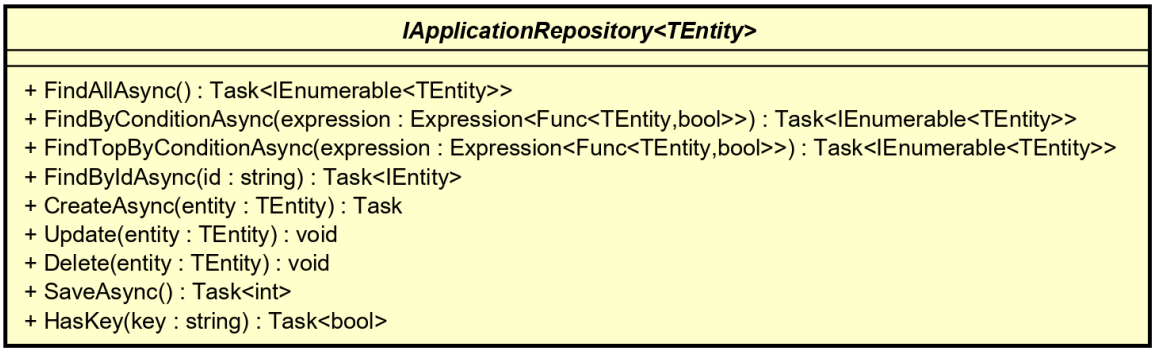
Rysunek 4.1. Diagram bazy danych

Dostęp do danych w bazie danych z poziomu części serwerowej odbywa się przy pomocy klas encyjnych i mapowania relacyjno-obiektowego oraz klas zapewniających podstawowe

operacje na danych. Diagram typów tej warstwy znajduje się na Rys. 4.2. Wszystkie klasy encyjne implementują interfejs `IEntity` oraz odwzorowują strukturę bazy danych, a metody dostępu do danych dla pozostałych warstw aplikacji zapewnione są przez generyczny interfejs `IApplicationRepository<TEntity>`, gdzie typ `TEntity` obrazuje klasę encyjną (Rys. 4.3). Dodatkowo w warstwie modelu danych znajduje się klasa `DataInitializer`, która zawiera początkowe dane, jakie baza danych musi zawierać przy uruchomieniu aplikacji. Dane te są umieszczane w bazie podczas migracji, czyli procesu tworzącego strukturę bazy danych na podstawie istniejących klas encyjnych.



Rysunek 4.2. Diagram zależności typów warstwy modelu danych



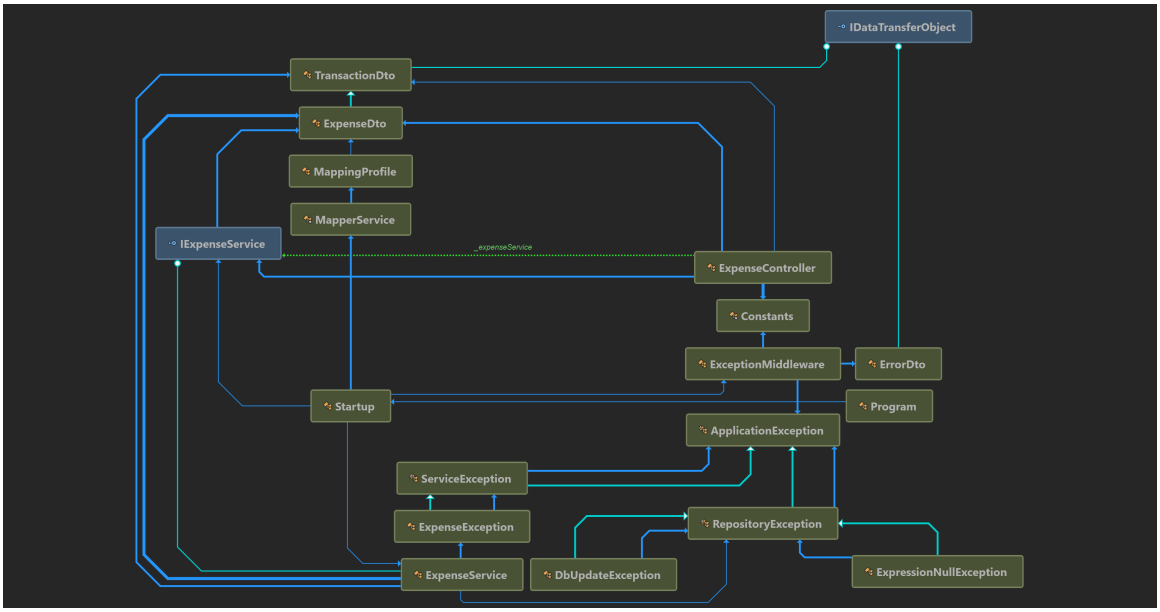
Rysunek 4.3. Diagram interfejsu `IApplicationRepository`

4.4. Warstwa logiki biznesowej

Na warstwę logiki biznesowej składają się klasy realizujące logikę przypadków użycia, typy definiujące DTO i wyjątki aplikacyjne oraz klasy kontrolerów, pozwalających na dostęp do części serwerowej z poziomu aplikacji klienckiej.

W warstwie tej można wydzielić cztery moduły: moduł danych użytkownika, moduł wydatków, moduł przychodów oraz moduł predykcji. Moduły te mają podobną strukturę klas.

Na Rys. 4.4 przedstawiony jest diagram zależności typów modułu wydatków. Klasa `ExpenseController` odpowiedzialna jest za obsługę zapytań aplikacji klienckiej. Po otrzymaniu zapytania, klasa `ExpenseService` implementująca interfejs `IExpenseService` przy pomocy repozytorium znajdującego się w warstwie modelu danych obsługuje zapytanie użytkownika oraz zwraca jego wynik przy użyciu typu implementującego `IDataTransferObject`. Mapowanie klasy encyjnej na typ DAO jest zdefiniowane w klasie `MappingProfile`.



Rysunek 4.4. Diagram zależności typów warstwy logiki biznesowej dla modułu wydatków.

4.5. Warstwa interfejsu użytkownika - aplikacja mobilna

5. Dokumentacja użytkownika aplikacji do zarządzania budżetem i predykcji wydatków

5.1. Moduł rejestracji wydatków i przychodów

5.2. Moduł predykcji wydatków

6. Podsumowanie

6.1. Wnioski

6.2. Perspektywy rozwoju

Bibliografia

[1] B. Pułaska-Turyna, *Statystyka dla ekonomistów. Wydanie III zmienione.* Difin SA, 2011. (Cytowanie na stronach 7 i 9.)

[2] M. Lebiedź, A. Weinstok, E. Wolska, and K. Zyskowska, "Wielowymiarowy model regresji liniowej." `http://www.mif.pg.gda.pl/homepages/kdz/StatystykaII/Rozdzial7b.pdf`, 2014. [dostęp 04.01.2019]. (Cytowanie na stronie 7.)

[3] M. Bremer, *Math 261A*. SAN JOSÉ STATE UNIVERSITY, 2012. (Cytowanie na stronach 8 i 9.)

[4] N. E. Helwig, "Nonparametric regression." `http://www.stat.cmu.edu/~larry/=sml/nonpar.pdf`, 2015. (Cytowanie na stronie 8.)

[5] R. Tibshirani and L. Wasserman, "Introduction to nonparametric regression." `http://users.stat.umn.edu/~helwig/notes/npreg-Notes.pdf`, 2017. (Cytowanie na stronie 8.)

[6] M. Rachelski, "Metryka | Informatyka MIMUW." `http://smurf.mimuw.edu.pl/node/220`, 2010. (Cytowanie na stronie 9.)

[7] J. Gosling, B. Joy, G. Steele, G. Bracha, and A. Buckley, *The Java® Language Specification. Java SE 8 Edition.* Oracle America, Inc, 2015. (Cytowanie na stronie 12.)

[8] B. Wagner, M. Wenzel, L. Latham, and P. Onderka, "A tour of C# - C# guide." `https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/index`, 2016. [dostęp 15.01.2019]. (Cytowanie na stronie 12.)

[9] D. Flanagan, *Javascript: The Definitive Guide. Fifth Edition.* O'Reilly Media, Inc., 2006. (Cytowanie na stronie 12.)

[10] D. Flanagan and Y. Matsimoto, *The Ruby Programming Language: Everything You Need to Know*. O'Reilly Media, Inc., 2008. (Cytowanie na stronie 13.)

[11] Apple Inc., "The Swift programming language." `https://docs.swift.org/swift-book/`, 2018. [dostęp 27.01.2019]. (Cytowanie na stronie 13.)

[12] JetBrains s.r.o., "Kotlin reference." `https://kotlinlang.org/docs/reference/faq.html`, 2018. [dostęp 27.01.2019]. (Cytowanie na stronie 13.)

[13] M. Shafirov, "Kotlin on android. Now official." `https://blog.jetbrains.com/kotlin/2017/05/kotlin-on-android-now-official/`, 2017. [dostęp 27.01.2019]. (Cytowanie na stronie 13.)

[14] R. Stoyanchev, "Spring framework overview." https://docs.spring.io/spring-framework/docs/current/spring-framework-reference/overview.html, 2019. [dostęp 27.01.2019]. (Cytowanie na stronie 13.)

[15] Microsoft Corporation, "What is .NET?." https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet, 2019. [dostęp 27.01.2019]. (Cytowanie na stronie 14.)

[16] Microsoft Corporation, "Visual studio ide, code editor, vsts, & app center." https://visualstudio.microsoft.com/. [dostęp 30.01.2019]. (Cytowanie na stronie 15.)

[17] JetBrains s.r.o., "Resharper: The visual studio extension for .NET developers by JetBrains." https://www.jetbrains.com/resharper/. [dostęp 30.01.2019]. (Cytowanie na stronie 15.)

[18] Postman, Inc., "Postman | API development environment." https://www.getpostman.com/. [dostęp 30.01.2019]. (Cytowanie na stronie 16.)

[19] SmartBear Software, "The best APIs are built with Swagger Tools | Swagger." https://swagger.io/. [dostęp 30.01.2019]. (Cytowanie na stronie 17.)

[20] Pivotal Software, Inc., "Understanding REST." https://spring.io/understanding/REST. [dostęp 30.01.2019]. (Cytowanie na stronie 17.)

[21] Google LLC, "MVC architecture." https://developer.chrome.com/apps/app_frameworks. [dostęp 30.01.2019]. (Cytowanie na stronie 18.)

[22] Microsoft Corporation, "Architectural principles." https://docs.microsoft.com/en-us/dotnet/standard/modern-web-apps-azure-architecture/architectural-principles#dependency-inversion. [dostęp 30.01.2019]. (Cytowanie na stronie 18.)

[23] S. Smith, S. Addie, and L. Latham, "Dependency injection in ASP.NET Core." https://docs.microsoft.com/en-us/aspnet/core/fundamentals/dependency-injection?view=aspnetcore-2.2. [dostęp 30.01.2019]. (Cytowanie na stronie 18.)

[24] Microsoft Corporation, "Entity Framework Core." https://docs.microsoft.com/en-us/ef/core/. [dostęp 30.01.2019]. (Cytowanie na stronie 18.)

[25] Microsoft Corporation, "Language Integrated Query (LINQ)." https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/. [dostęp 30.01.2019]. (Cytowanie na stronie 19.)

[26] J. Bogard, "AutoMapper." https://automapper.org/. [dostęp 01.02.2019]. (Cytowanie na stronie 20.)

[27] C. Rüegg, “Math.NET Numerics.” https://numerics.mathdotnet.com/. [dostęp 01.02.2019]. (Cytowanie na stronie 21.)

[28] Newtonsoft, “Json.NET - Newtonsoft.” https://www.newtonsoft.com/json. [dostęp 01.02.2019]. (Cytowanie na stronie 21.)

[29] Microsoft Corporation, “NuGet Gallery | Statistics.” https://www.nuget.org/stats. [dostęp 01.02.2019]. (Cytowanie na stronie 21.)

Spis rysunków

Spis listingów