Politechnika Łódzka
Instytut Informatyki

System planowania budżetu domowego z aspektem społecznościowym

Praca dyplomowa inżynierska

Wydział Fizyki Technicznej, Informatyki i Matematyki Stosowanej

Promotor: dr inż. Łukasz Chomątek

Dyplomant: Paweł Purgat

Nr albumu: 203975

Łódź, 2019

Instytut Informatyki
90-924 Łódź, ul. Wólczańska 215, budynek B9
tel. 042 631 27 97, 042 632 97 57, fax 042 630 34 14 email: office@ics.p.pl

Spis treści

1. Wstęp

1.1. Problematyka i zakres pracy

Niniejsza praca porusza problem zarządzania budżetem oraz predykcji wydatków, czyli ich prognozowania. Istnieje wiele rozwiązań służących do rejestracji przychodów i wydatków, zarówno w postaci systemów komputerowych, jak i tradycyjnych metod. Systemy, z których korzysta wielu użytkowników przechowują dane o dochodach i wydatkach użytkowników. Zbiór takich danych daje możliwość wykorzystania ich do modelowania zależności między nimi i wykonywania predykcji na ich podstawie.

Prognozowanie wydatków może w znaczącym stopniu pomóc w planowaniu budżetu oraz ułatwić zarządzanie dostępnymi środkami pieniężnymi.

W zakres niniejszej pracy wchodzi stworzenie systemu, który umożliwia rejestrację wydatków i przychodów użytkowników, a następnie na podstawie tak zebranych danych zapewnia prognozowanie przyszłych wydatków. System ten składa się z części serwerowej odpowiedzialnej za wykonywanie wszelkich operacji na zebranych danych oraz aplikacji klienckiej, która zapewnia interakcję systemu z użytkownikiem.

1.2. Cele pracy

Niniejsza praca powstała w celu:

- wyboru odpowiednich metod analizy danych, zastosowania ich w aplikacji służącej do predykcji wydatków oraz przeglądu istniejących aplikacji oferujących zbliżone funkcjonalności,
- stworzenia aplikacji mobilnej służącej do rejestrowania i planowania wydatków, wykorzystującej zebrane w ten sposób dane statystyczne do ich predykcji,
- określenia użyteczności stworzonej aplikacji, przydatności metod analizy danych do predykcji wydatków oraz porównania różnych modeli reprezentujących przetwarzane dane.

1.3. Przegląd literatury w dziedzinie komputerowej analizy danych

Beata Pułaska-Turyna, Statystyka dla ekonomistów. Wydanie III zmienione. - pozycja ta opisuje metody służące do analizy i modelowania struktury danych oraz ich szacowania i prognozowania.

Ryan Tibshirani, Nonparametric Regression - praca zawierająca opis metod regresji nieparametrycznej oraz wyznaczania jej estymatorów.

1.4. Opis zawartości pracy

W drugim rozdziale niniejszej pracy opisane zostały istniejące rozwiązania w dziedzinie zarządzania budżetem oraz metody analizy danych służące do predykcji. Trzeci rozdział zawiera opis technologii użytych w projekcie systemu ExpensePredictor. Dokumentacja techniczna systemu ExpensePredictor zawarta jest w rozdziale czwartym, a w rozdziale piątym znajduje się dokumentacja użytkownika tego systemu.

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. Zarządzanie budżetem oznacza analizę wszystkich poniesionych kosztów oraz planowanie przyszłych wydatków. Prawidłowe zarządzanie budżetem pozwala na 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.

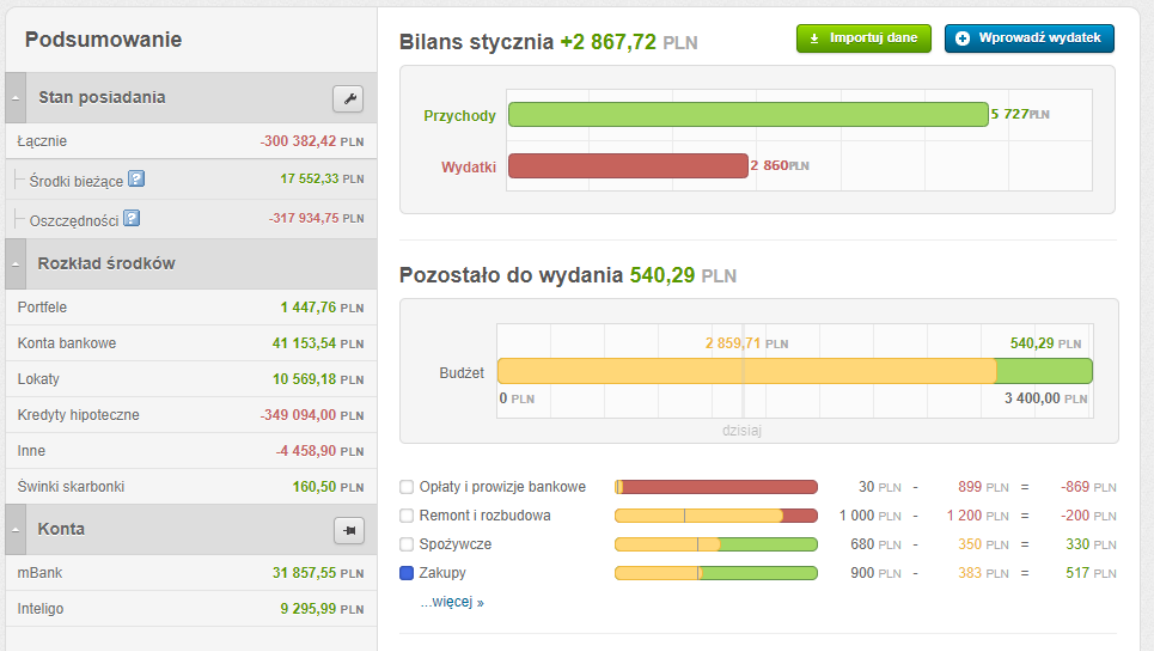
Powstała aplikacja na podstawie danych o przychodach i rozchodach wszystkich użytkowników pozwala na **predykcję**, czyli przewidywanie wydatków powiązanych z zakupem produktów lub usług, które niosą za sobą koszty utrzymania, na przykład samochodu.

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. Aplikacja ta pobiera dane o wydatkach

użytkownika z historii transakcji na koncie bankowym. Użytkownikowi prezentowana jest w przejrzysty sposób struktura 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.

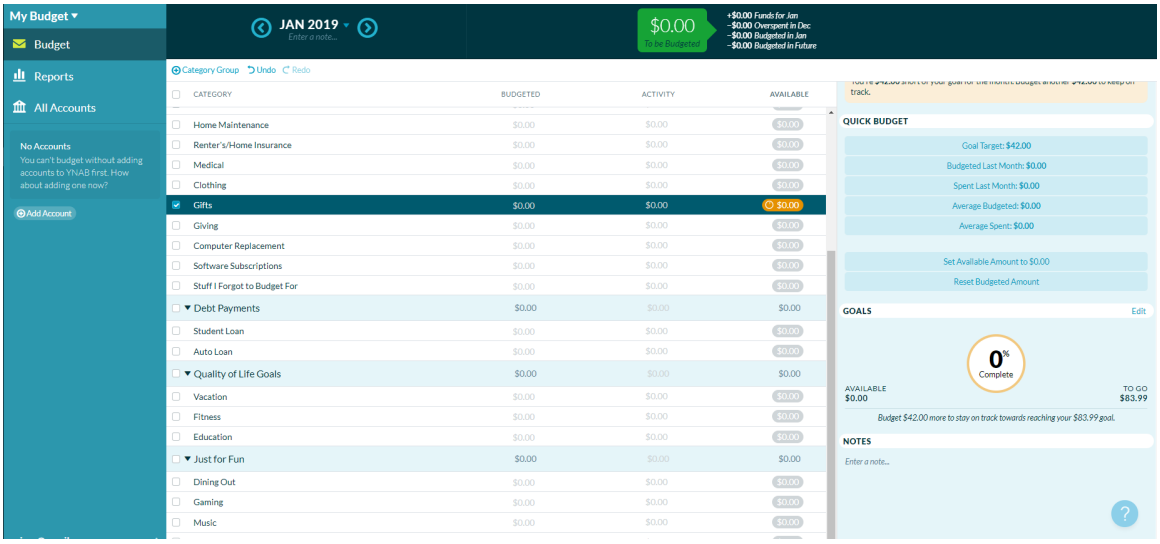


Rysunek 2.1: Interfejs aplikacji Kontomierz

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.

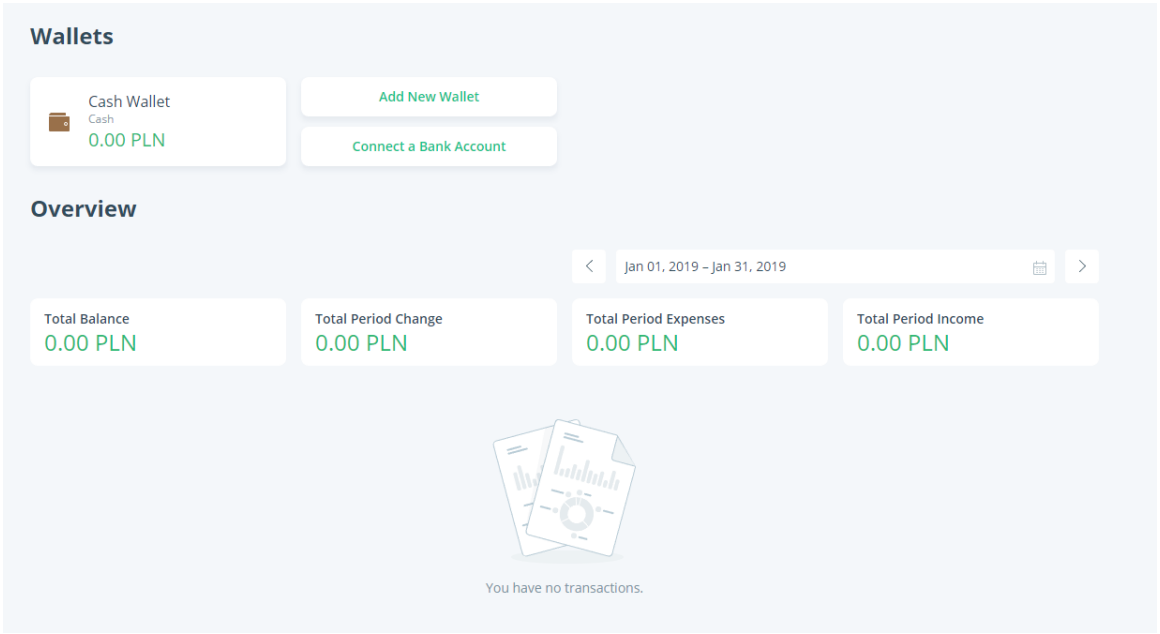
Spendee (Rys. 2.3) jest aplikacją anglojęzyczną służącą do zarządzania budżetem. Jest dostępna zarówno w formie aplikacji webowej, jak i mobilnej. Aplikacja posiada wersję podstawową oraz dwie wersje płatne różniące się dostępnymi funkcjonalnościami. Pakiet



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]

Zmienne zależne są to wartości, które przy pomocy gotowego modelu regresji będą prognozowane. Predyktory są zmiennymi niezależnym, których wartość wpływa na zmienną zależną. Na podstawie predyktorów wykonywana jest prognoza. Współczynniki regresji są wagami predyktorów, czyli wartościami, przez które mnożone są kolejne predyktory tak, że suma tych iloczynów tworzy prognozę. Reszty są różnicami faktycznej wartości zmiennej zależnej i wartości jej prognozy.

2.4.2. Regresja nieparametryczna

Regresja nieparametryczna, w odróżnieniu od regresji parametrycznej (na przykład 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, na przykład 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:

$$\beta = (X^T X)^{-1} X^T Y$$

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 aplikacji: części serwerowej (ang. *backend*) oraz aplikacji klienckiej (ang. *frontend*).
Obecnie istnieje na rynku wiele technologii umożliwiających realizację części serwerowej, jak na przykład Spring (Java), ASP .NET (C#), ASP .NET Core (C#). Technologie te różnią się wymaganiami oraz dostępnymi narzędziami.
Interfejs użytkownika został zrealizowany w formie aplikacji mobilnej. Aplikacje te można podzielić na trzy kategorie:

- Natywne - zbudowane dla konkretnej platformy i napisane w języku dla niej odpowiednim, na przykład 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 na przykład 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 jest platformą ułatwiającą tworzenie aplikacji z użyciem języka Java. 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 jest darmową, otwartoźródłową platformą programistyczną służącą do wytwarzania różnych typów aplikacji. Platforma .NET jest dostępna dla języków C#, F# oraz 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 darmowym i otwartoźródłowym, wieloplatformowym frameworkiem używanym do tworzenia wszelkiego rodzaju aplikacji 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] Pozwala na definiowanie niezależnych od systemu operacyjnego interfejsów użytkownika w języku XAML.

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.

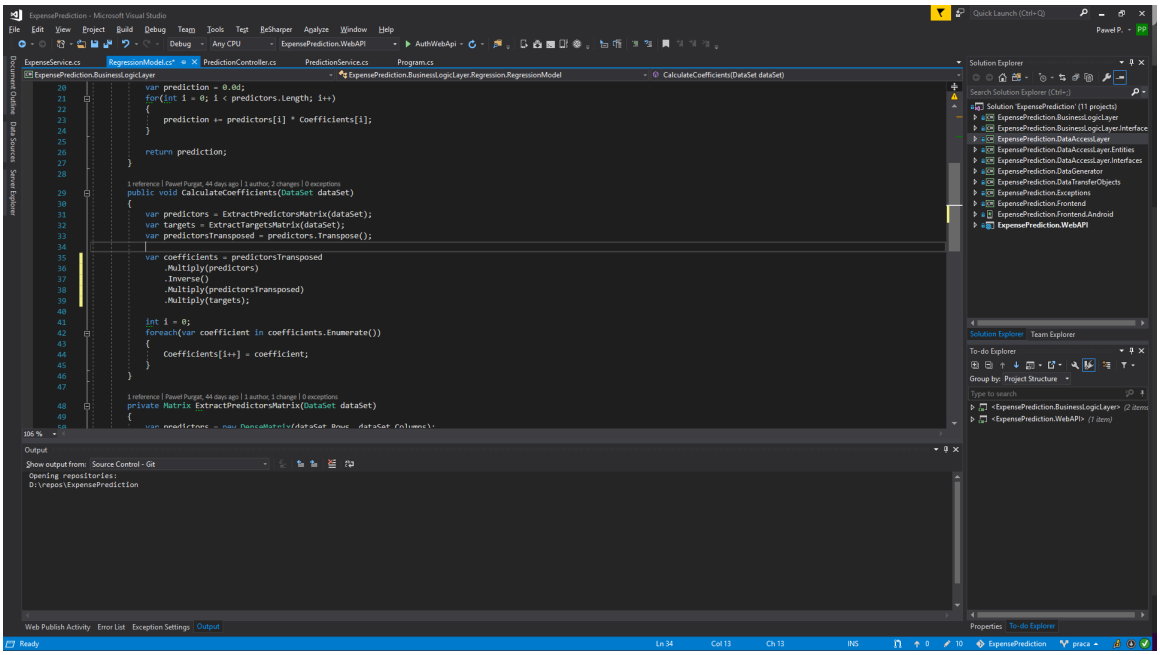
System wykonany na potrzeby niniejszej pracy stworzony został przy użyciu platform ASP .NET Core (backend) oraz Xamarin Forms (frontend). Obie te platformy wykorzystują język C# oraz dysponują zestawem narzędzi zapewnionym przez .NET Standard. Umożliwia to współdzielenie części kodu między tymi dwiema platformami oraz usprawnia proces wytwarzania oprogramowania.

Spring jest sprawdzoną, powszechnie używaną platformą do tworzenia aplikacji internetowych. ASP .NET Core jest nowszą i nie aż tak dobrze przetestowanym frameworkiem, jak Spring, ale zawiera narzędzia, których brak w Springu. Bardzo przydatnym narzędziami dostępnymi w .NET Core jest LINQ, który pozwala na konstruowanie zapytań wykonywanych na danych różnych formatów oraz wsparcie programowania asynchronicznego przez biblioteki wchodzące w skład tej platformy. Dodatkową zaletą frameworków z rodziny .NET jest ścisła integracja

ze środowiskiem Microsoft Visual Studio. 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. 3.1) 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]

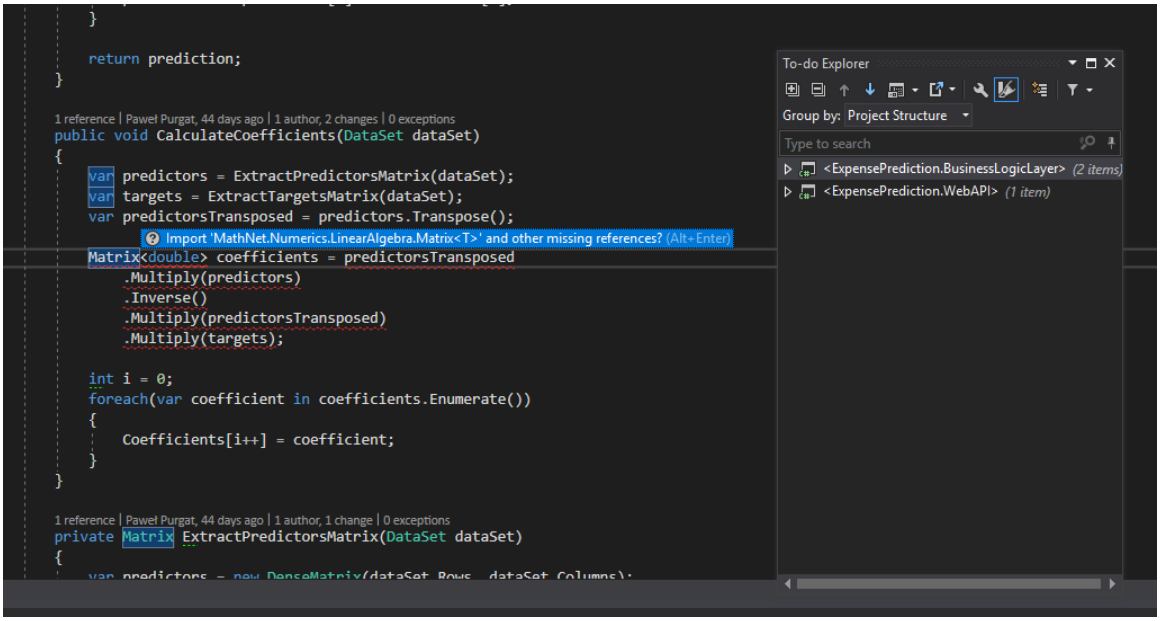


Rysunek 3.1: Interfejs środowiska Microsoft Visual Studio

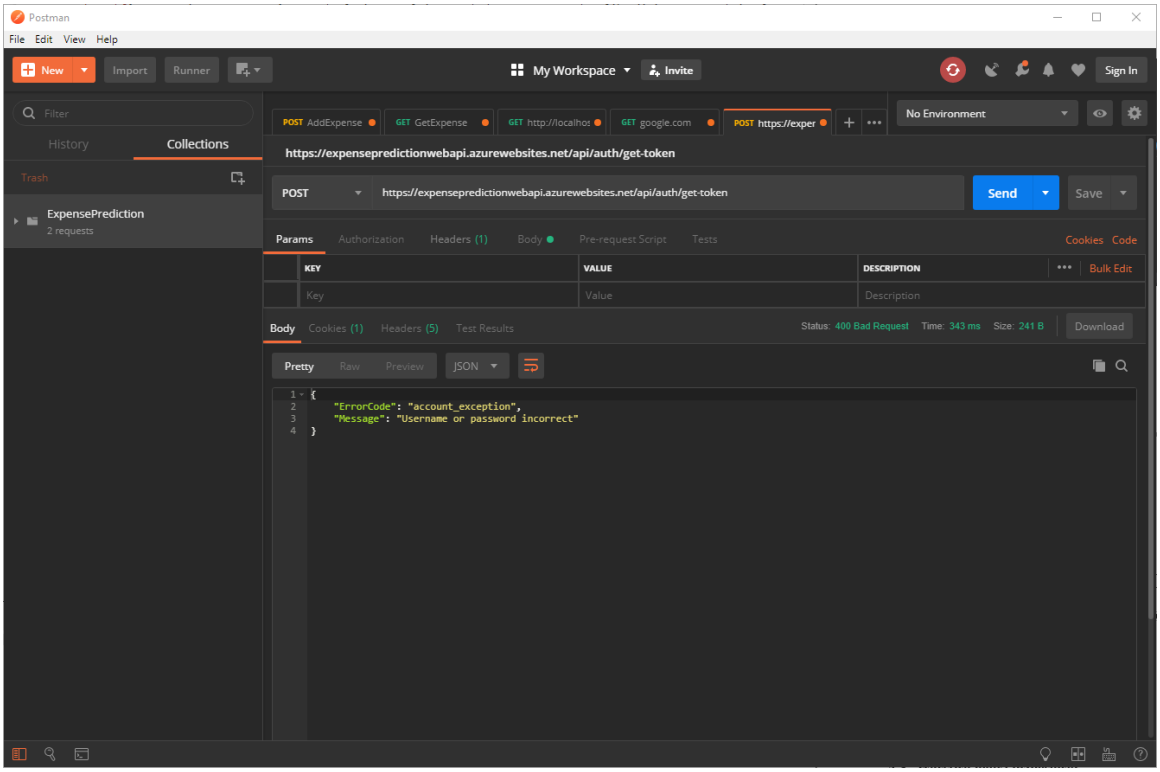
ReSharper (Rys. 3.2) jest narzędziem rozszerzającym środowisko Microsoft Visual Studio autorstwa JestBrains. Usprawnia ono proces wytwarzania oprogramowania poprzez automatyzację wielu procesów związanych z pisaniem wysokiej jakości kodu. Zapewnia ciągłą analizę kodu, co ułatwia wykrycie wielu błędów.[17]

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. 3.4) jest narzędziem służącym do automatycznej dokumentacji kodu oraz zapewniającym konfigurowalny graficzny interfejs obrazujący stworzone API. Z poziomu

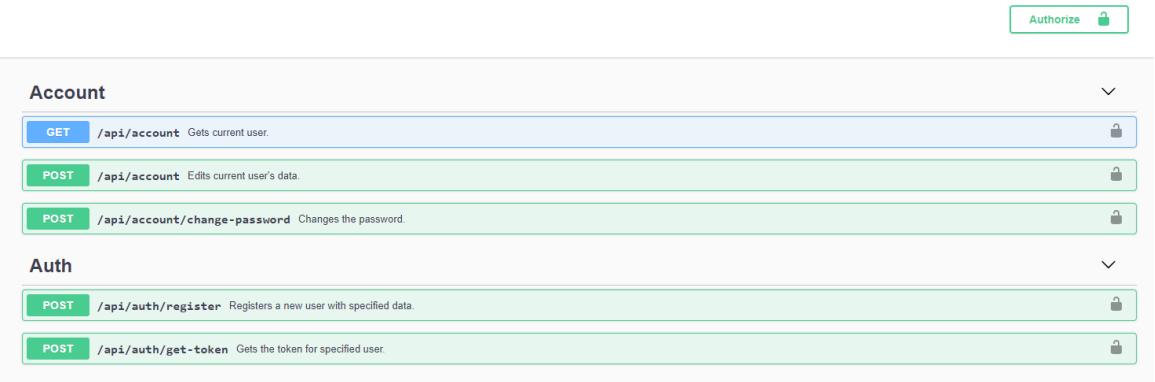


Rysunek 3.2: Przykładowe funkcjonalności rozszerzenia ReSharper



Rysunek 3.3: Interfejs aplikacji Postman

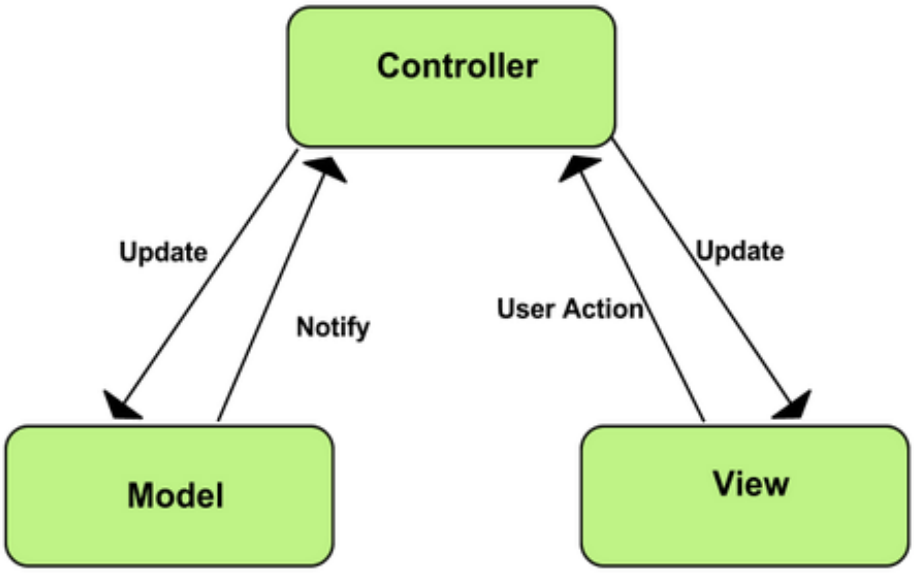
wygenerowanego interfejsu możliwe jest proste konstruowanie zapytań zgodnych ze stworzoną aplikacją.[19]



Rysunek 3.4: 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]

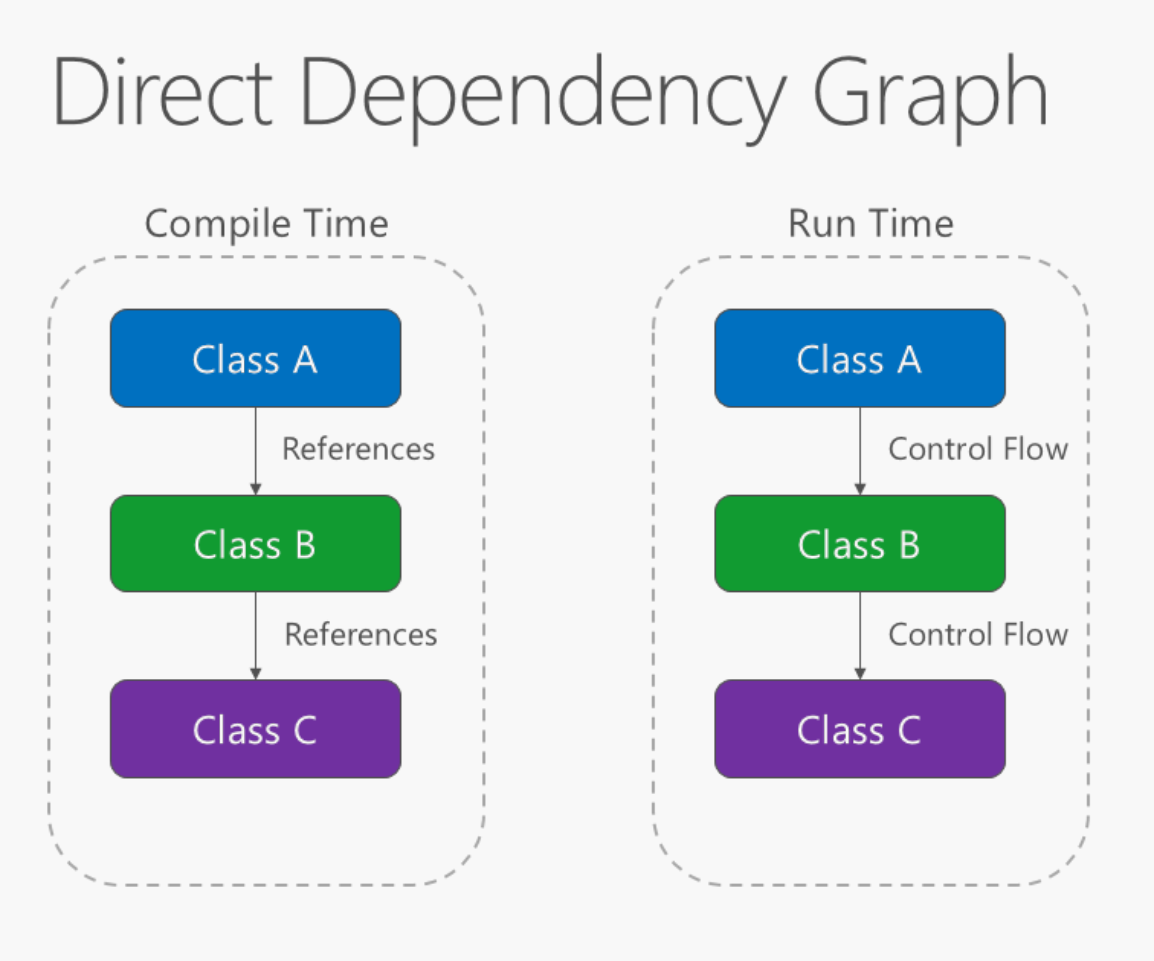
MVC (Model-View-Controller) (Rys. 3.5) to wzorzec projektowy polegający na podzieleniu projektu aplikacji na warstwy modelu danych, widoku, czyli interfejsu użytkownika oraz kontrolera, który reaguje na akcje wykonane przez użytkownika w warstwie widoku oraz odczytuje i modyfikuje dane z warstwy modelu.[21]



Rysunek 3.5: 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. 3.6). Odwrócenie sterowania polega na skonstruowaniu zależności w

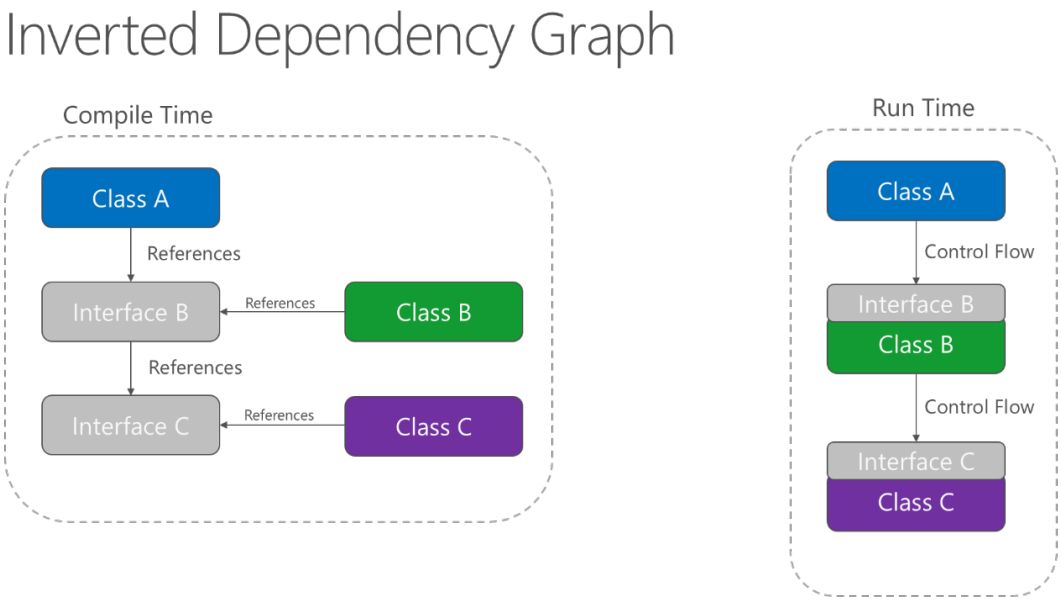


Rysunek 3.6: Diagram przedstawiający zależność bezpośrednią (https://docs.microsoft.com/en-us/dotnet/standard/ modern-web-apps-azure-architecture/media/image4-1.png)

taki sposób, aby szczegółowa implementacja zależała od abstrakcyjnego komponentu (Rys. 3.7).[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]



Rysunek 3.7: Diagram przedstawiający zależność odwróconą (https://docs.microsoft.com/en-us/dotnet/standard/ modern-web-apps-azure-architecture/media/image4-2.png)

Language Integrated Query (LINQ) to zbiór technologii których zadaniem jest zintegrowanie możliwości wykonywania zapytań na danych z językiem C#. Zapewnia on jednolitą składnię tworzenia zapytań dla wielu źródeł danych, jak na przykład 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.

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 na przykład 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)

```
Matrix firstMatrix = GetFirstMatrix();
Matrix secondMatrix = GetSecondMatrix();
```

```
Matrix thirdMatrix = GetThirdMatrix();

var resultMatrix = firstMatrix
.Multiply(secondMatrix)
.Add(thirdMatrix);
```

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]

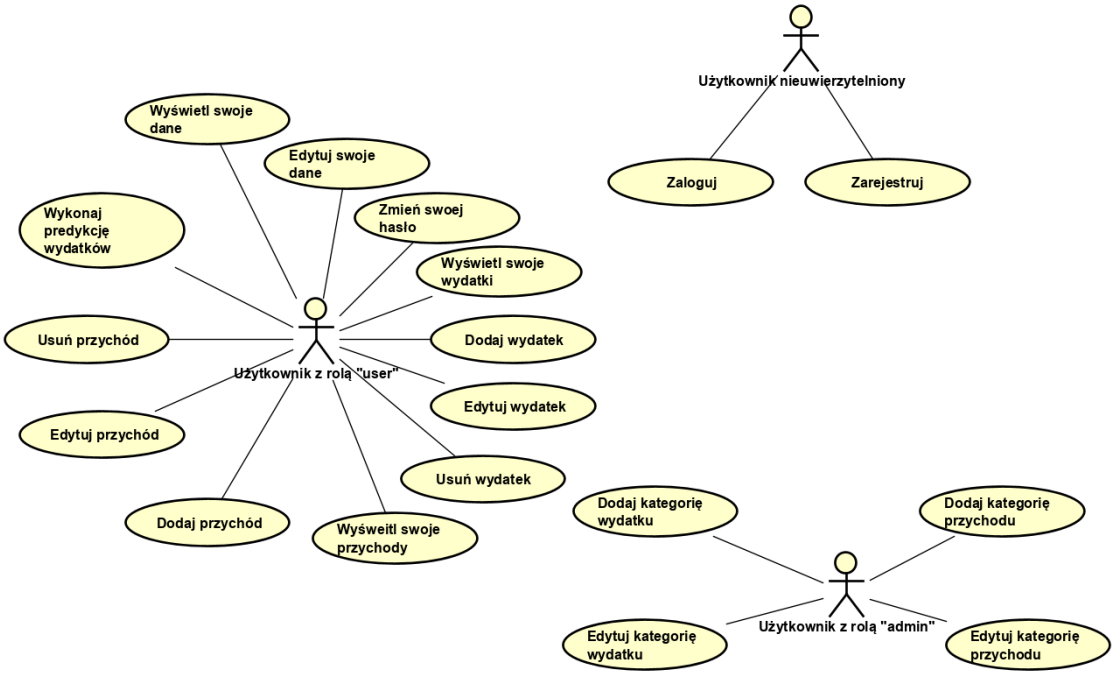
4. Dokumentacja techniczna aplikacji ExpensePredictor

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, są przedstawione na diagramie przypadków użycia (Rys. 4.1) oraz opisane poniżej:

- **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,
- **dodanie kategorii wydatków lub przychodów** - dodanie przez administratora kategorii, do której użytkownicy mogą przypisywać swoje wydatki lub przychody,
- **edycja kategorii wydatków lub przychodów** - zmiana przez administratora opisu istniejącej kategorii wydatków lub przychodów.



Rysunek 4.1: Diagram przypadków użycia

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,

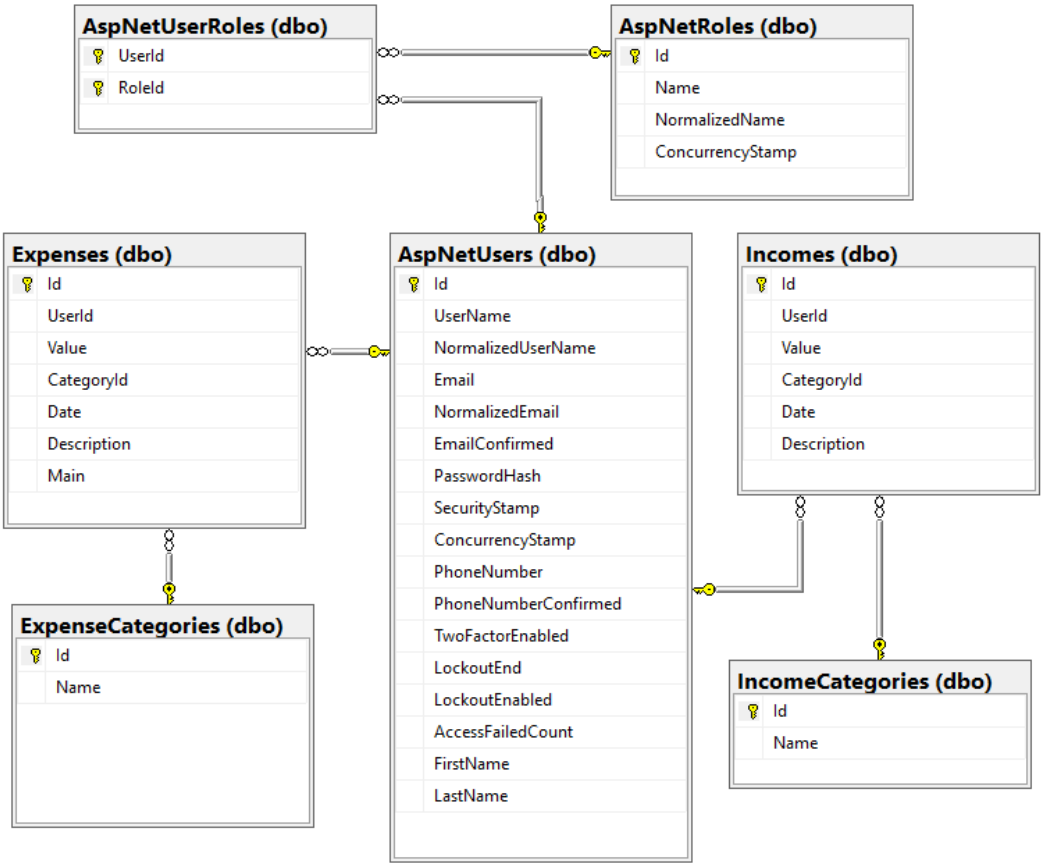
- **dostęp do systemu dla uwierzytelnionych użytkowników** - możliwość skorzystania z funkcjonalności systemu (z kilkoma wyjątkami) jest dostępna jedynie dla uwierzytelnionych użytkowników,
- **dostęp do funkcjonalności ograniczony rolami** - dostęp do funkcjonalności jest ograniczony poprzez role przypisane do konta użytkownika, na przykład 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

Warstwa modelu danych składa się z bazy danych, klas encyjnych oraz klas zapewniających podstawowe operacje na danych.

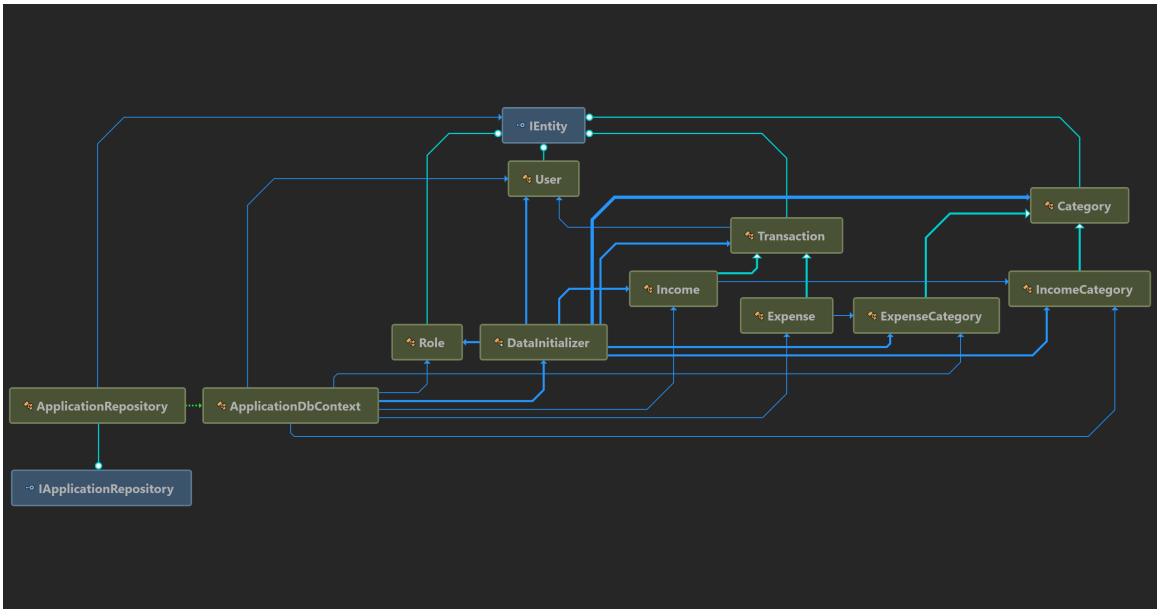
Dane w bazie danych (Rys. 4.2) podzielone są na siedem tabel:

- **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.

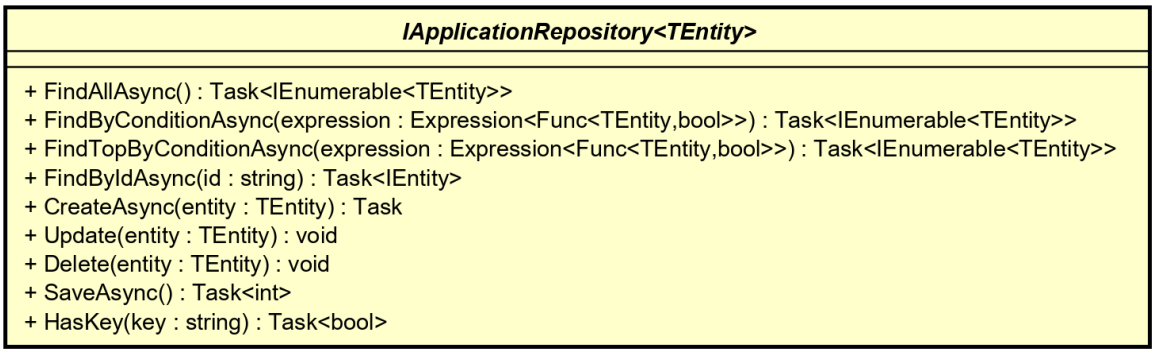


Rysunek 4.2: Diagram bazy danych

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. 4.3. 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. 4.4). 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 4.3: Diagram zależności typów warstwy modelu danych



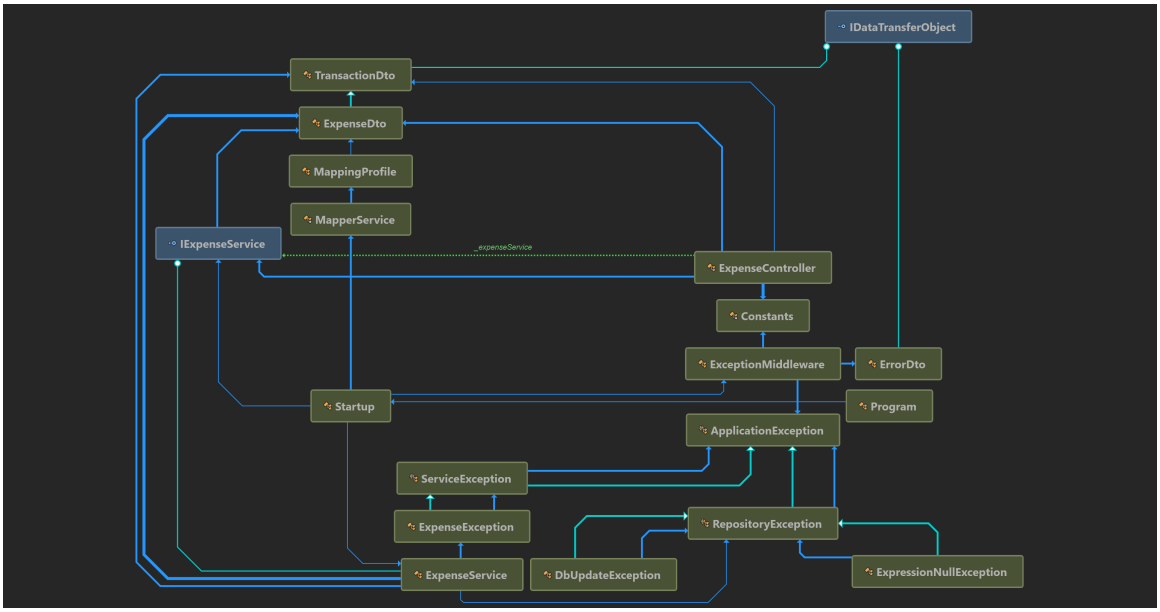
Rysunek 4.4: Diagram interfejsu `IApplicationRepository`

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. Na Rys. 4.5 przedstawiony jest diagram klas modułu wydatków, pozostałe moduły posiadają analogiczną strukturę.

Wszystkie operacje poza logowaniem oraz rejestracją wymagają od użytkownika podania tokenu uwierzytelniającego. Następnie przebiega weryfikacja tokenu, czyli sprawdzenie podpisu cyfrowego w celu stwierdzenia, czy nie został on zmieniony bądź wygenerowany w



Rysunek 4.5: Diagram zależności typów warstwy logiki biznesowej dla modułu wydatków.

inny sposób niż przy procesie logowania lub rejestracji oraz zweryfikowanie, czy użytkownik posiada role wymagane do wykonania danej operacji.

W celu rejestracji, po otrzymaniu danych o nowym koncie od użytkownika, przy pomocy zapewnionej przez ASP .NET Core klasy `UserManager` tworzony jest nowe konto. Po utworzeniu użytkownika przypisywane są do niego role początkowe. Jeżeli proces rejestracji przebiegł pomyślnie, generowany jest token oraz zwracany użytkownikowi, przez co po rejestracji logowanie przebiega automatycznie.

Logowanie istniejącego użytkownika polega na wygenerowaniu dla niego tokenu uwierzytelniającego go w systemie. Po otrzymaniu danych w postaci loginu oraz hasła, są one weryfikowane. Przy pomyślnej weryfikacji odczytywane są z bazy danych wszystkie role przypisane do użytkownika i wraz z innymi danymi o użytkowniku zapisywane tokenie. Wygenerowany token jest podpisywany cyfrowo, co pozwala na jego weryfikację przy próbie uwierzytelnienia i odrzucenie w razie wykrycia jakiejkolwiek ingerencji w jego zawartość.

Zmiana hasła użytkownika wymaga podania starego oraz nowego hasła. Po weryfikacji poprawności dotychczasowego hasła generowany jest skrót nowego hasła, który zastępuje istniejący skrót. Oprócz zmiany hasła użytkownik może także zmienić swoje dane osobowe.

Dodawanie i edycja kategorii są dostępne jedynie dla użytkownika posiadającego rolę "admin". Operacje te sprowadzają się do zmodyfikowania bądź stworzenia rekordu w bazie

danych zawierającego opis danej kategorii. Raz stworzonej kategorii nie można usunąć z powodu powiązanych z nią wydatków bądź dochodów użytkowników.

Dodawanie wydatku lub przychodu użytkownika jest procesem, który dodaje nowy rekord z wydatkiem lub przychodem powiązanego z użytkownikiem pobranym z tokenu uwierzytelniającego. Przy dodawaniu nowego wydatku, w zależności od jego wielkości w porównaniu z miesięcznymi przychodami użytkownika, jest ustawiana flaga oznaczająca, że wydatek jest wydatkiem głównym.

W momencie uruchomienia części serwerowej systemu wyznaczane są współczynniki modelu regresji dla każdej kategorii wydatków oraz co określony czas są one aktualizowane. Dla każdego modelu dane o wydatkach i przychodach są ekstrahowane i zapisywane w postaci macierzy predyktorów i wartości zależnych. Następnie, według metody najmniejszych kwadratów opisanej w rozdziale 2.4.3 obliczane są współczynniki modelu regresji. Operacje te przedstawione są na listingu 4.1. W ten sposób obliczone dla każdej kategorii wydatków współczynniki zapisywane są następnie w statycznej kolekcji.

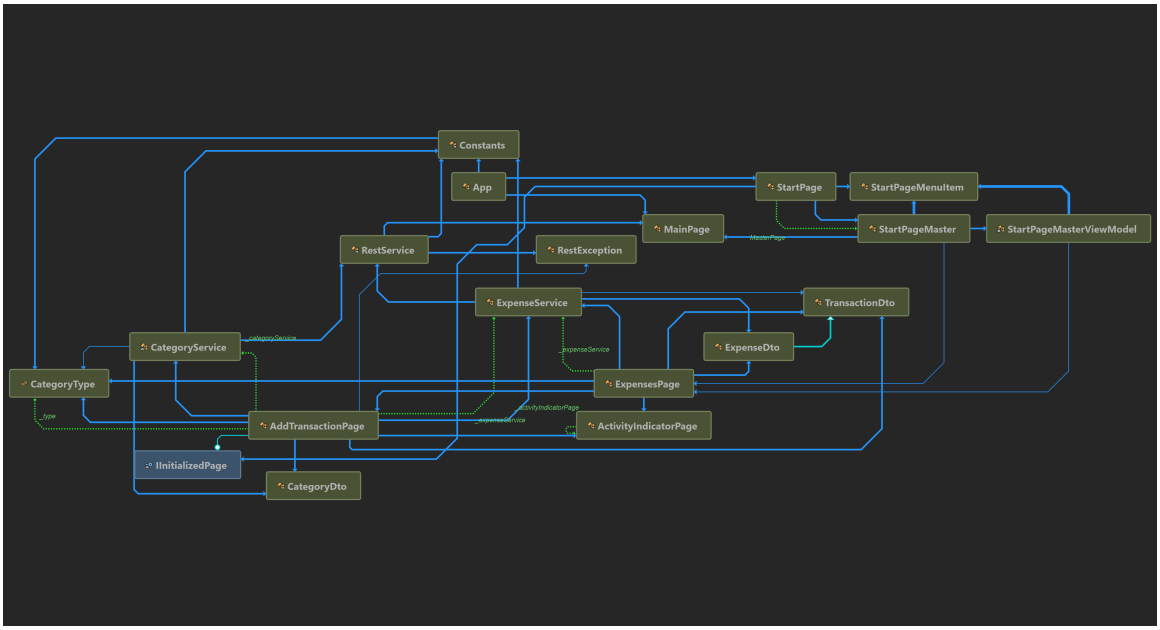
```
var predictors = ExtractPredictorsMatrix(dataSet);
var targets = ExtractTargetsMatrix(dataSet);
var predictorsTransposed = predictors.Transpose();

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

Listing 4.1: Operacje wyznaczania współczynników modelu regresji.

Proces predykcji wydatków na określony cel polega na pobraniu od użytkownika kosztu wydatku głównego oraz jego kategorii, a następnie na podstawie jego miesięcznych przychodów, wydatków z wybranej kategorii oraz podanego kosztu wydatku przy użyciu wyznaczonych współczynników modelu regresji wyznaczane są koszta związane z określonym wydatkiem.

4.5. Warstwa interfejsu użytkownika - aplikacja mobilna

Warstwa interfejsu użytkownika zrealizowana jest w postaci aplikacji mobilnej. Składa się ona ze stron wyświetlanych na ekranie urządzenia. Diagram klas wchodzących w skład części aplikacji odpowiadającej za obsługę wydatków przedstawiony jest na Rys. 4.6. Analogicznie ustrukturyzowane są moduły aplikacji odpowiadające za pozostałe funkcjonalności systemu.



Rysunek 4.6: Diagram zależności typów warstwy logiki biznesowej dla modułu wydatków.

Aplikacja obsługuje wszystkie akcje użytkownika wykonywane na stronach aplikacji przy użyciu serwisów, których zadaniem jest odczytanie danych wprowadzonych przez użytkownika i skonstruowanie zapytania do części serwerowej. Każde wysyłane zapytanie oprócz zapytań dotyczących rejestracji i logowania posiada dołączony nagłówek `Authorization`, który zawiera token uwierzytelniający wygenerowany w części serwerowej aplikacji.

Dzięki zastosowaniu do komunikacji z częścią serwerową metod asynchronicznych interfejs użytkownika jest responsywny i podczas, gdy serwer przetwarza zapytanie wyświetlany jest stosowny wskaźnik oczekiwania.

5. Dokumentacja użytkownika aplikacji ExpensePredictor

W niniejszym rozdziale opisany został interfejs użytkownika oraz funkcjonalność systemu ExpensePredictor. Diagram wszystkich przypadków użycia przedstawiony jest na Rys. 4.1.

5.1. Moduł obsługi konta użytkownika

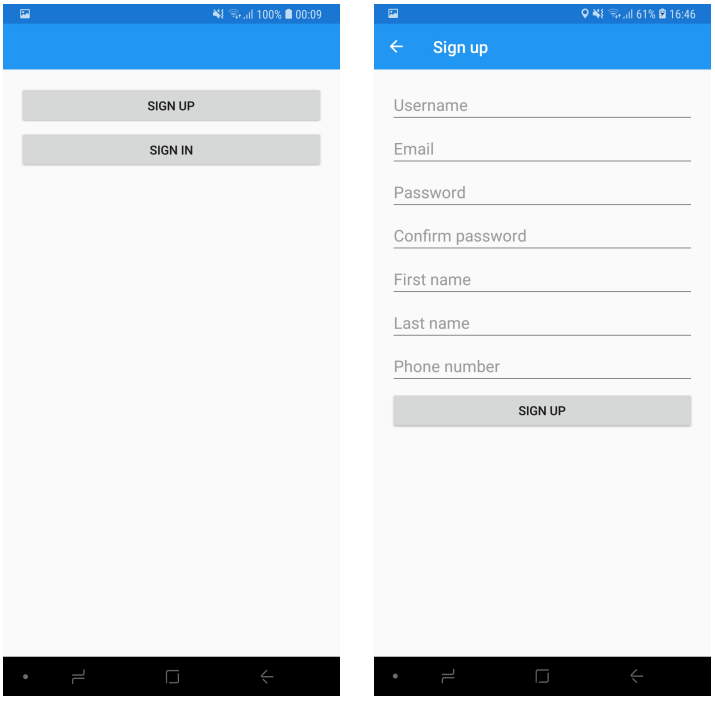
Moduł obsługi konta użytkownika swoim zakresem obejmuje pięć przypadków użycia:

- zarejestruj,
- zaloguj,
- wyświetl swoje dane,
- edytuj swoje dane,
- zmień swoje hasło.

Rejestracja (przypadek użycia "zarejestruj") to proces tworzenia nowego konta użytkownika. Jest jedną z dwóch funkcjonalności niewymagających uwierzytelnienia i dostępnych przy pierwszym uruchomieniu aplikacji. Po wybraniu na ekranie startowym (Rys. 5.1a) przycisku "SIGN UP" otwarta zostaje strona rejestracji (Rys. 5.1b). Po poprawnym wypełnieniu pól i naciśnięciu przycisku "SIGN UP" konto zostaje utworzone.

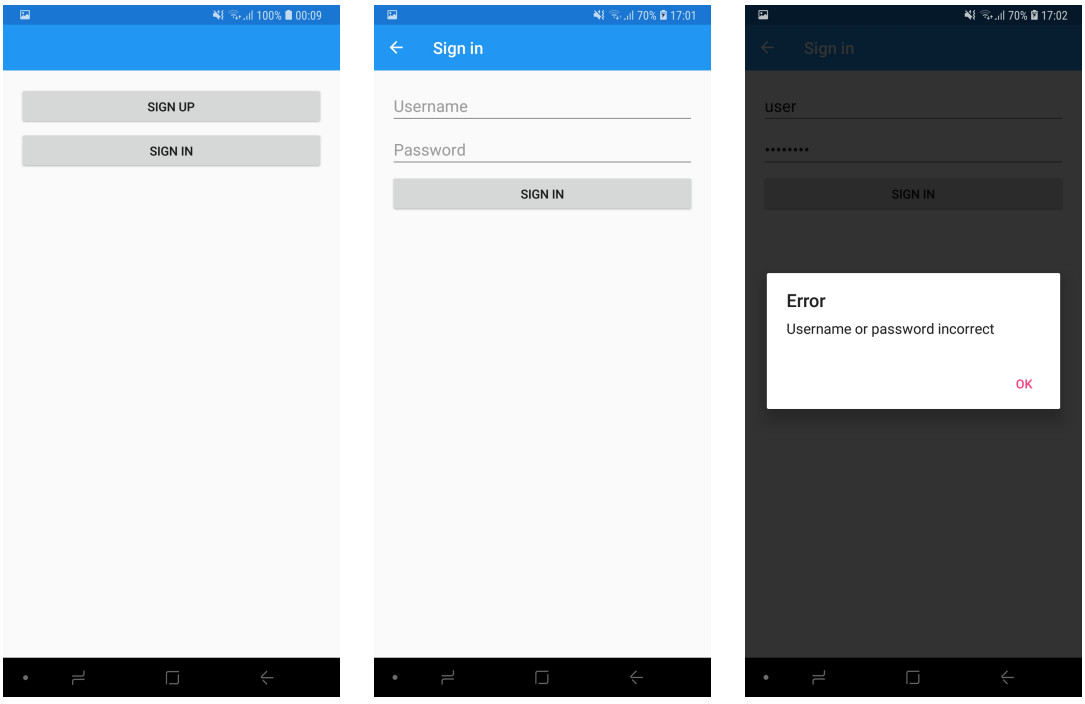
Logowanie (przypadek użycia "zaloguj") jest procesem generacji tokenu uwierzytelniającego. Po wybraniu na ekranie startowym (Rys. 5.2a) przycisku "SIGN IN" wyświetla się strona logowania (Rys. 5.2b). Po wypełnieniu formularza i wciśnięciu przycisku "SIGN IN", w przypadku błędnego hasła wyświetla się komunikat (Rys. 5.2c), w przeciwnym wypadku proces kończy się powodzeniem.

Wyświetlanie danych użytkownika (przypadek użycia "wyświetl swoje dane") - pobranie danych użytkownika i wyświetlenie ich użytkownikowi. Po uwierzytelnieniu się w aplikacji i wybraniu z menu bocznego strony "User details" (Rys. 5.3a) użytkownikowi prezentowane są jego dane (Rys 5.3b).



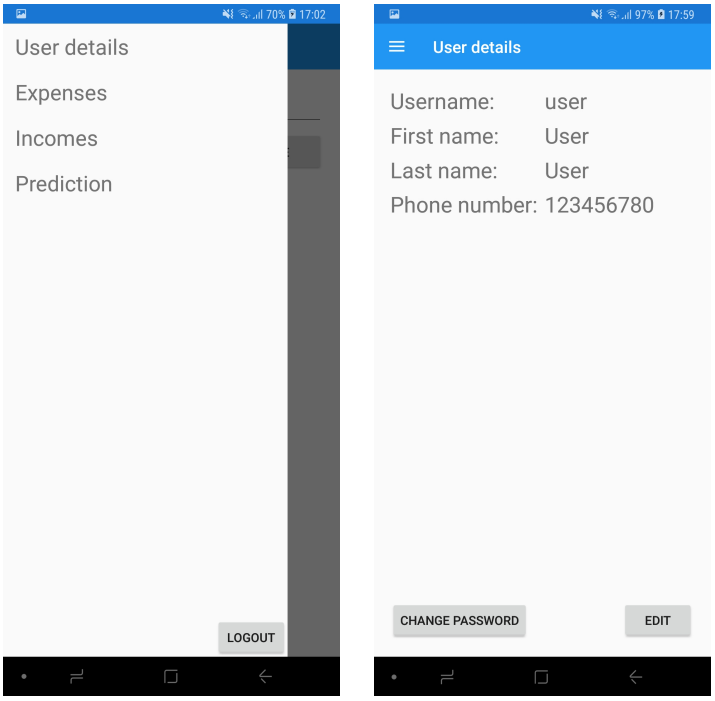
(a) Ekran startowy. (b) Strona rejestracji.

Rysunek 5.1: Zrzuty ekranu procesu rejestracji.



(a) Ekran startowy. (b) Strona logowania. (c) Błędne hasło.

Rysunek 5.2: Zrzuty ekranu procesu logowania.



(a) Menu boczne. (b) Dane użytkownika.

Rysunek 5.3: Zrzuty ekranu procesu wyświetlania danych użytkownika.

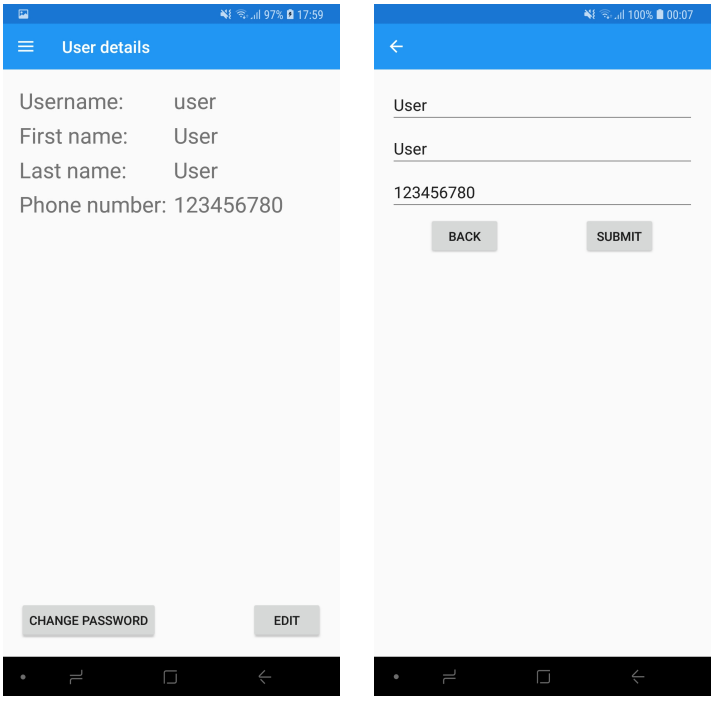
Edycja danych użytkownika (przypadek użycia "edytuj swoje dane") - zmiana imienia, nazwiska bądź numeru telefonu użytkownika. Po przejściu na ekran danych użytkownika i wybraniu przycisku "EDIT" (Rys. 5.4a) otwierany jest formularz edycji (Rys. 5.4b). Wciśnięcie przycisku "SUBMIT" edytuje dane użytkownika, przycisk "BACK" odrzuca wszelkie wprowadzone zmiany.

Zmiana hasła (przypadek użycia "zmień swoje hasło) - ustawienie nowego hasła użytkownika. Po wybraniu przycisku "CHANGE PASSWORD" na ekranie danych użytkownika (Rys. 5.5a) wyświetlany jest formularz zmiany hasła (Rys. 5.5b). Poprawne wypełnienie formularza i naciśnięcie przycisku "SUBMIT" skutkuje zmianą hasła lub, jeżeli nie spełnia ono polityki haseł, wyświetlany jest komunikat o błędzie (Rys. 5.5c). Wciśnięcie przycisku "BACK" powoduje powrót do strony danych użytkownika.

5.2. Moduł rejestracji przychodów

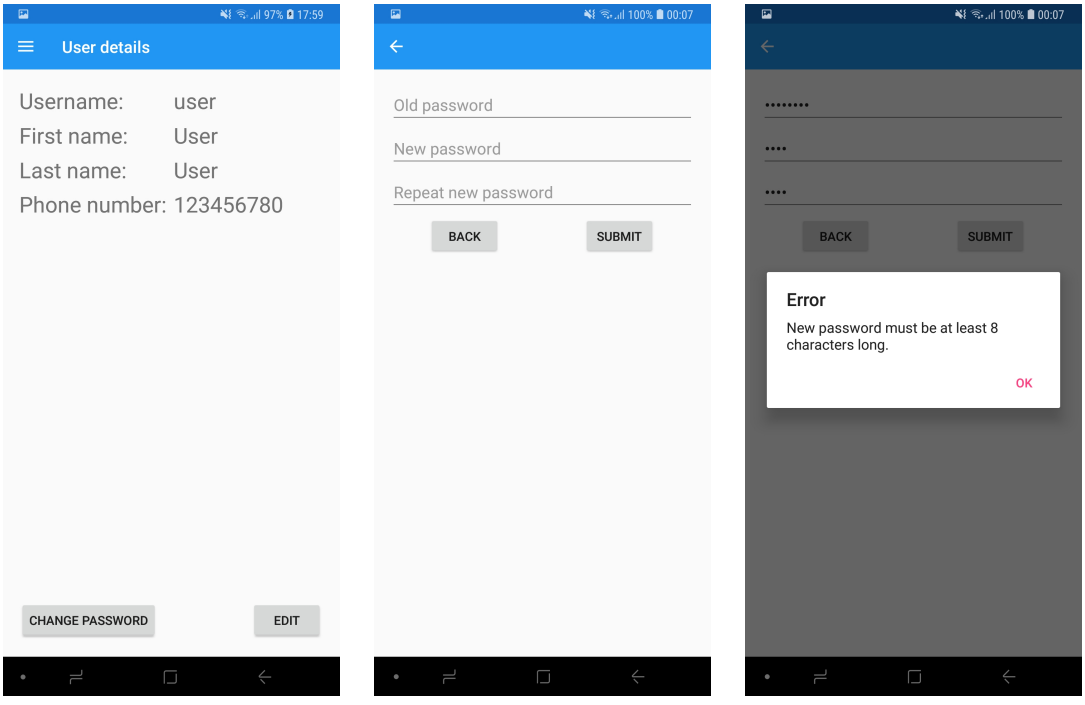
Moduł rejestracji przychodów składa się z czterech przypadków użycia:

- wyświetl swoje przychody,
- dodaj przychód,



(a) Dane użytkownika. (b) Edycja danych użytkownika.

Rysunek 5.4: Zrzuty ekranu procesu edycji danych użytkownika.

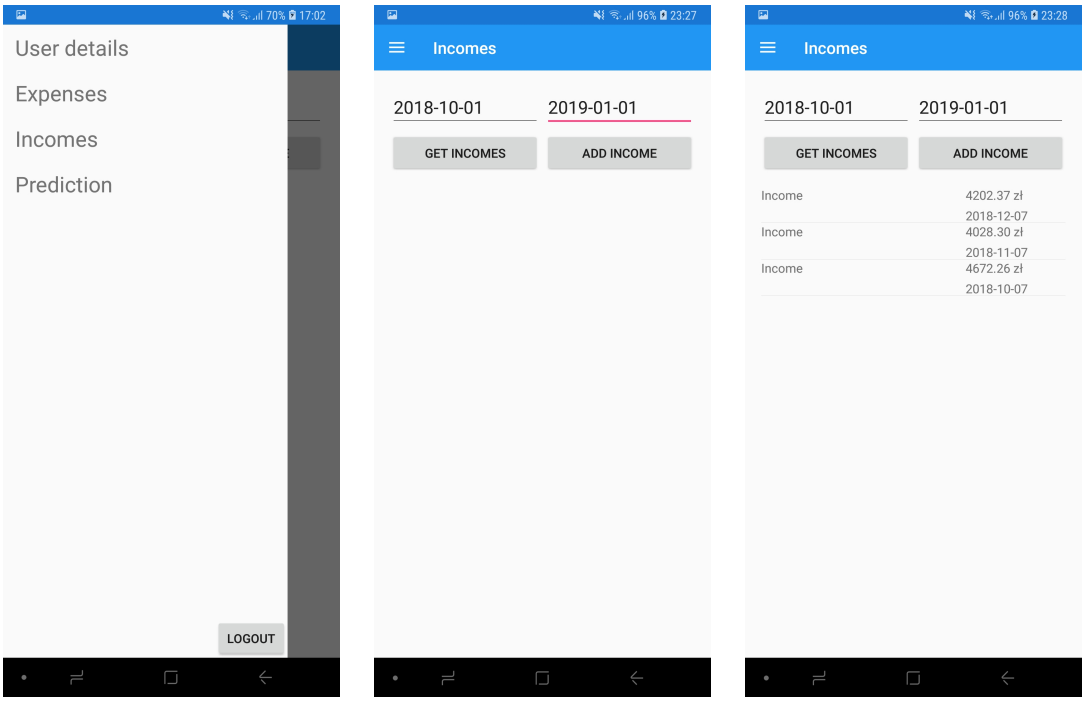


(a) Dane użytkownika. (b) Formularz zmiany hasła. (c) Komunikat o błędzie.

Rysunek 5.5: Zrzuty ekranu procesu zmiany hasła.

- edytuj przychód,
- usuń przychód.

Wyświetlanie przychodów (przypadek użycia "wyświetl swoje przychody") - pobranie i zaprezentowanie użytkownikowi listy jego przychodów z danego okresu. Po wybraniu z menu bocznego (Rys. 5.6a) strony "Incomes" (Rys. 5.6b), określeniu daty początkowej i końcowej oraz naciśnięciu przycisku "GET INCOMES" wyświetlana jest lista przychodów z wybranego okresu (Rys. 5.6c).

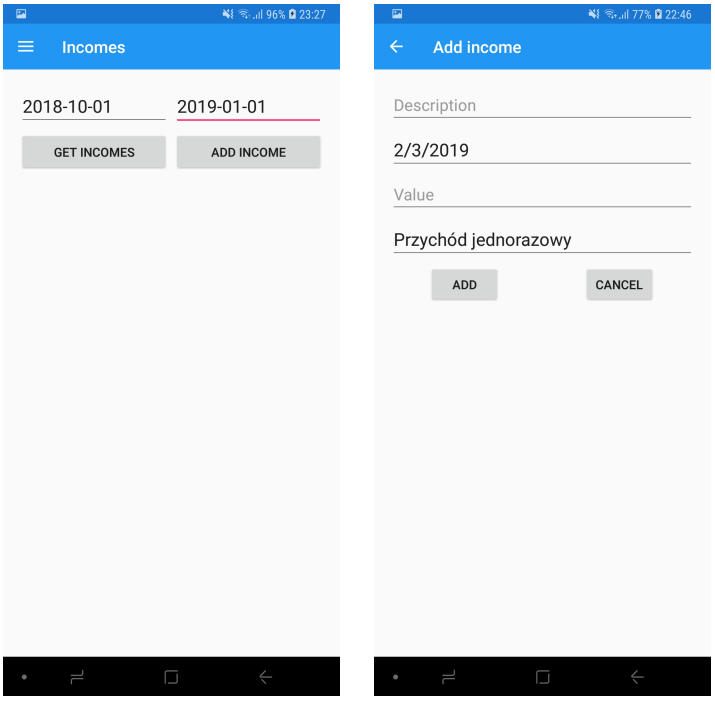


(a) Menu boczne. (b) Strona z przychodami. (c) Lista przychodów.

Rysunek 5.6: Zrzuty ekranu procesu wyświetlania przychodów.

Dodawanie przychodu (przypadek użycia "dodaj przychód") - dodanie wpisu z przychodem uwierzytelnionego użytkownika. Wybranie przycisku "ADD INCOME" na stronie przychodów (Rys. **??**) powoduje wyświetlenie formularza dodawania przychodu (Rys. 5.7b). Po wypełnieniu formularza przy pomocy przycisku "ADD" przychód jest dodawany, a przycisk "BACK" skutkuje powrotem do strony przychodów.

Edycja przychodu (przypadek użycia "edytuj przychód") - zmiana wartości, opisu lub kategorii istniejącego przychodu uwierzytelnionego użytkownika. W widoku listy przychodów (Rys. 5.8a), naciśnięcie konkretnego przychodu powoduje wyświetlenie menu z opcjami (Rys. 5.8b). Po naciśnięciu "Edit" wyświetla się formularz z edycją przychodu (Rys. 5.8c), który,



(a) Strona przychodów.

(b) Formularz dodawania przychodu.

Rysunek 5.7: Zrzuty ekranu procesu dodawania przychodu.

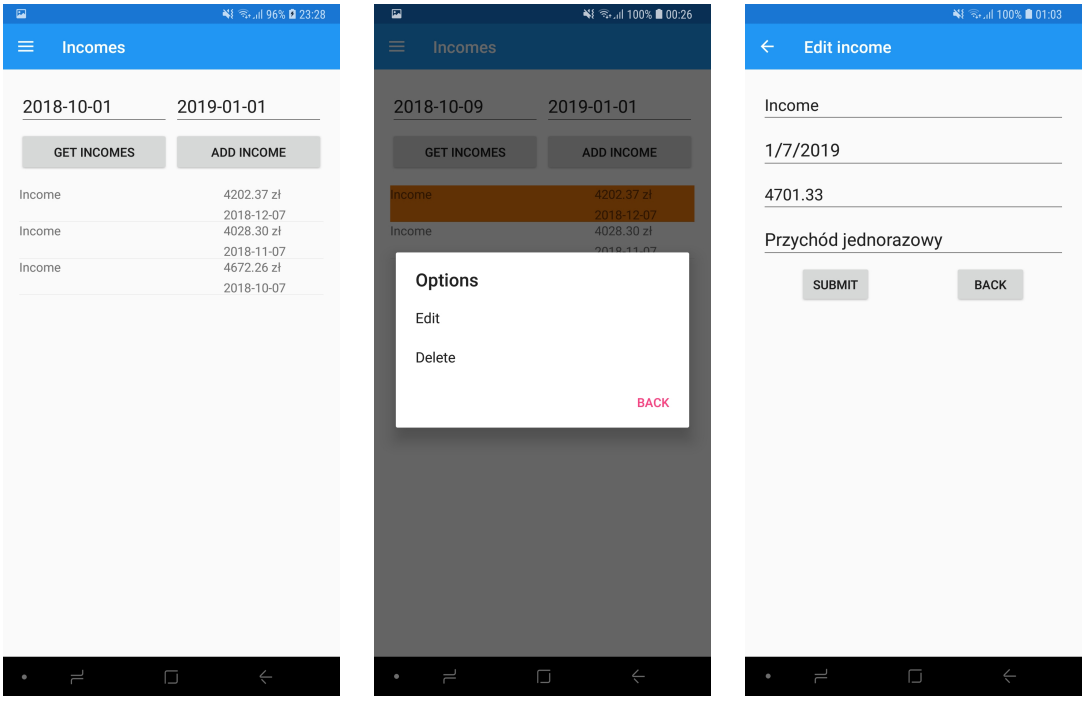
po wypełnieniu, można wysłać przyciskiem "SUBMIT", co powoduje zapisanie zmian, lub powrócić do listy przychodów przyciskiem "BACK".

Usuwanie przychodu (przypadek użycia "usuń przychód") - usunięcie istniejącego przychodu należącego do aktualnie uwierzytelnionego użytkownika. Po naciśnięciu przychodu na liście przychodów (Rys. 5.9a), wyświetlane jest menu (Rys. 5.9b). Aby usunąć przychód należy w menu wybrać opcję "Delete" oraz potwierdzić chęć wykonania operacji (Rys. 5.9c).

5.3. Moduł rejestracji wydatków

W zakres modułu rejestracji wydatków wchodzą następujące przypadki użycia:

- wyświetl swoje wydatki,
- dodaj wydatek,
- edytuj wydatek,
- usuń wydatek,
- wykonaj predykcję wydatków.



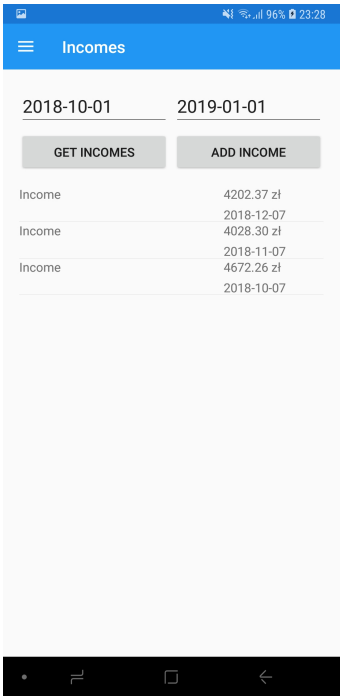
(a) Lista przychodów. (b) Menu przychodu. (c) Formularz edycji przychodu.

Rysunek 5.8: Zrzuty ekranu procesu edycji przychodu.

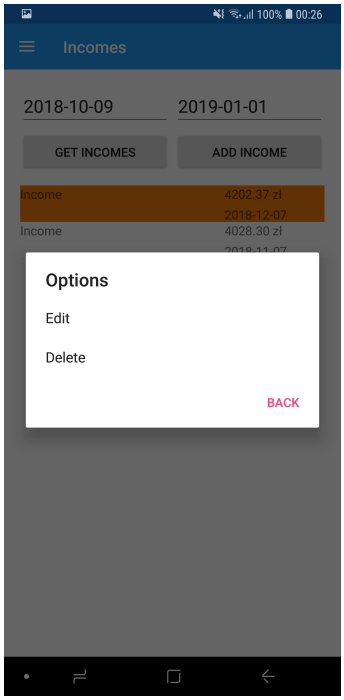
Operacje wyświetlania, dodawania, edycji oraz usuwania wydatków wykonuje się analogicznie do ich odpowiedników z modułu rejestracji przychodów.

Predykcja wydatków (przypadek użycia "wykonaj predykcję wydatków") jest to wykonanie prognozy wydatków powiązanych z wydatkiem głównym na nadchodzące miesiące. Po wybraniu z menu bocznego (Rys. 5.10a) strony "Prediction" (Rys. 5.10b), uzupełnieniu wartości, daty oraz kategorii wydatku i naciśnięcie przycisku "PREDICTION" użytkownik otrzymuje prognozę wydatków na trzy nadchodzące miesiące (Rys. 5.10c).

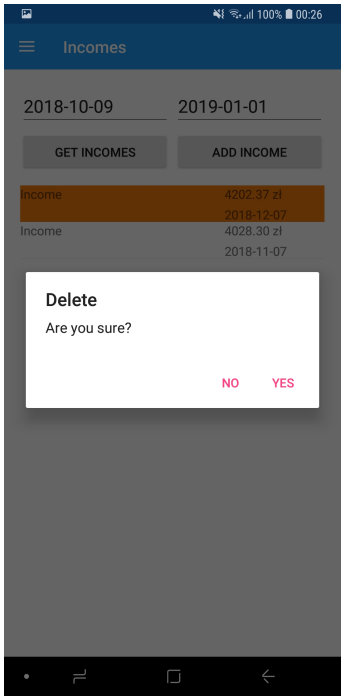
Przedstawione na Rys. 5.10b wartości prognozowanych kosztów obliczone są na podstawie wielkości wydatku wprowadzonego przez użytkownika oraz znajdujących się w bazie wygenerowanych przykładowych danych o przychodach i wydatkach istniejących użytkowników. Dane te obrazują wydatki, które powiązane są z wprowadzonym przez użytkownika wydatkiem głównym. Na potrzeby testów systemu ExpensePredictor wygenerowanych zostało około 1000 rekordów z danymi o przychodach i wydatkach. W przypadku wdrożenia systemu dane pochodzić będą od użytkowników i będą odwzorowywać faktyczne zależności między danymi.



(a) Lista przychodów.



(b) Menu przychodu.

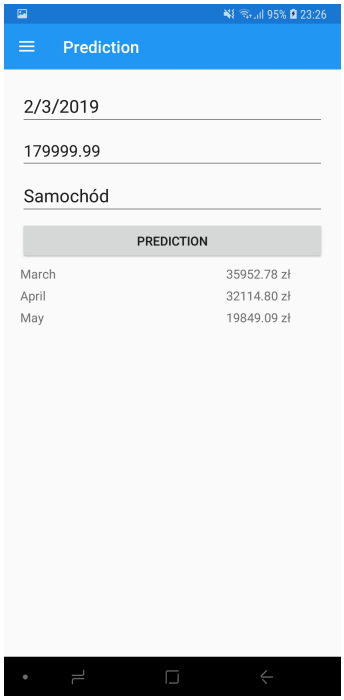


(c) Okno potwierdzające chęć usunięcia przychodu.

Rysunek 5.9: Zrzuty ekranu procesu usuwania przychodu.

(a) Menu boczne.

(b) Strona predykcji.



(c) Wyniki predykcji.

Rysunek 5.10: Zrzuty ekranu procesu wykonywania predykcji wydatków.

6. Podsumowanie

6.1. Wnioski

Stworzony system ExpensePredictor jest systemem służącym do zarządzania budżetem. Na rynku istnieje wiele produktów o takim przeznaczeniu. System ExpensePredictor wyróżnia się tym, że oferuje funkcjonalność unikatową, którą jest predykcja wydatków, czyli przewidywanie wartości przyszłych wydatków użytkownika.

Predykcja danych opiera się na metodach regresji parametrycznej lub nieparametrycznej. Regresja parametryczna jest w stanie stworzyć matematyczny model danych i na jego podstawie wykonywać prognozę, lecz wymaga ściśle ustalonej struktury danych i możliwości matematycznego opisu zależności między nimi. Regresja nieparametryczna sprawdza się lepiej w przypadkach, gdy zależności między danymi nie są ściśle określone i nie jest możliwe stworzenie matematycznego modelu je opisującego.

Kategoryzacja wydatków pozwala uwzględnić różnice w zależnościach między danymi o wydatkach z różnych kategorii. Stworzenie modelu regresji dla konkretnej kategorii danych pozwala na dokładniejsze modelowanie zależności, a co za tym idzie trafniejsze prognozy.

Dzięki uwzględnieniu danych wszystkich użytkowników systemu w procesie tworzenia modelu regresji możliwa jest analiza większego przekroju danych statystycznych, co przekłada się na bardziej dopasowany do rzeczywistości model regresji.

6.2. Perspektywy rozwoju

Po zebraniu i analizie stosunkowo dużego zbioru danych użytkowników systemu możliwa będzie modyfikacja istniejącej bądź stworzenie własnej metody analizy danych, która lepiej odwzoruje zależności między zebranymi danymi i pozwoli na jeszcze trafniejsze prognozy.

W celu ułatwienia i usprawnienia procesu korzystania z systemu, będzie w przyszłości możliwa integracja z systemem e-paragonów. System ten, według Ministerstwa Cyfryzacji, ma zostać wprowadzony w Polsce i do końca 2020 roku ma w całości zastąpić fizyczne

paragony.[30] Możliwa jest również integracja z istniejącymi systemami bankowymi i rejestracja wydatków i przychodów na podstawie danych udostępnionych przez te systemy.

Integrując system ExpensePredictor z zewnętrznymi systemami dostarczającymi dane o wydatkach i przychodach możliwe będzie opracowanie i zastosowanie algorytmu pozwalającego na automatyczną kategoryzację wydatków. Usprawni to proces korzystania z systemu przez użytkownika.

Bibliografia

[1] B. Pułaska-Turyna, *Statystyka dla ekonomistów. Wydanie III zmienione.* Difin SA, 2011. (Cytowanie na stronach 9 i 11.)

[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 9.)

[3] M. Bremer, *Math 261A*. SAN JOSÉ STATE UNIVERSITY, 2012. (Cytowanie na stronach 10 i 12.)

[4] N. E. Helwig, "Nonparametric regression." `http://www.stat.cmu.edu/~larry/=sml/nonpar.pdf`, 2015. (Cytowanie na stronie 10.)

[5] R. Tibshirani and L. Wasserman, "Introduction to nonparametric regression." `http://users.stat.umn.edu/~helwig/notes/npreg-Notes.pdf`, 2017. (Cytowanie na stronie 11.)

[6] M. Rachelski, "Metryka | Informatyka MIMUW." `http://smurf.mimuw.edu.pl/node/220`, 2010. (Cytowanie na stronie 11.)

[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 14.)

[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 14.)

[9] D. Flanagan, *Javascript: The Definitive Guide. Fifth Edition.* O'Reilly Media, Inc., 2006. (Cytowanie na stronie 14.)

[10] D. Flanagan and Y. Matsimoto, *The Ruby Programming Language: Everything You Need to Know*. O'Reilly Media, Inc., 2008. (Cytowanie na stronie 15.)

[11] Apple Inc., "The Swift programming language." `https://docs.swift.org/swift-book/`, 2018. [dostęp 27.01.2019]. (Cytowanie na stronie 15.)

[12] JetBrains s.r.o., "Kotlin reference." `https://kotlinlang.org/docs/reference/faq.html`, 2018. [dostęp 27.01.2019]. (Cytowanie na stronie 15.)

[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 15.)

[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 15.)

[15] Microsoft Corporation, "What is .NET?." https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet, 2019. [dostęp 27.01.2019]. (Cytowanie na stronie 16.)

[16] Microsoft Corporation, "Visual studio ide, code editor, vsts, & app center." https://visualstudio.microsoft.com/. [dostęp 30.01.2019]. (Cytowanie na stronie 17.)

[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 17.)

[18] Postman, Inc., "Postman | API development environment." https://www.getpostman.com/. [dostęp 30.01.2019]. (Cytowanie na stronie 17.)

[19] SmartBear Software, "The best APIs are built with Swagger Tools | Swagger." https://swagger.io/. [dostęp 30.01.2019]. (Cytowanie na stronie 18.)

[20] Pivotal Software, Inc., "Understanding REST." https://spring.io/understanding/REST. [dostęp 30.01.2019]. (Cytowanie na stronie 19.)

[21] Google LLC, "MVC architecture." https://developer.chrome.com/apps/app_frameworks. [dostęp 30.01.2019]. (Cytowanie na stronie 19.)

[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 20.)

[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 20.)

[24] Microsoft Corporation, "Entity Framework Core." https://docs.microsoft.com/en-us/ef/core/. [dostęp 30.01.2019]. (Cytowanie na stronie 20.)

[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 21.)

[26] J. Bogard, "AutoMapper." https://automapper.org/. [dostęp 01.02.2019]. (Cytowanie na stronie 22.)

[27] C. Rüegg, “Math.NET Numerics.” https://numerics.mathdotnet.com/. [dostęp 01.02.2019]. (Cytowanie na stronie 22.)

[28] Newtonsoft, “Json.NET - Newtonsoft.” https://www.newtonsoft.com/json. [dostęp 01.02.2019]. (Cytowanie na stronie 23.)

[29] Microsoft Corporation, “NuGet Gallery | Statistics.” https://www.nuget.org/stats. [dostęp 01.02.2019]. (Cytowanie na stronie 23.)

[30] P. Ratyński, “Strumień e-Faktura i e-Paragon.” https://www.gov.pl/web/cyfryzacja/strumien-e-faktura-i-e-paragon, 2017. (Cytowanie na stronie 41.)

Spis rysunków

Spis listingów