

Dokumentacja projektu

Przedmiot:

Wzorce projektowe i architektura aplikacji

Tytuł projektu:

Aplikacja do wyszukiwania cen akcji

Prowadzący: Wykonawca:

dr Marek Jaszuk Wojciech Marek

Nr albumu: 57108

Rzeszów, 2020

# **Opis założeń projektu.**

Celem projektu jest stworzenie aplikacji umożliwiającej użytkownikowi wyszukiwanie akcji spółek giełdowych oraz ich cen. Dodatkowo aplikacja powinna umożliwiać wyświetlanie szczegółów notowań poszczególnych akcji w dwóch przedziałach czasowych, tj. godzinowym i dziennym. Aplikacja udostępniać ma także możliwość utworzenia przez użytkownika konta, w którym to po zalogowaniu będzie istniała możliwość zapisania ulubionych spółek do prywatnej listy ulubionych aktywów. Projekt ma zostać wykonany w oparciu o technologię Blazor Server, dostępną w frameworku   
.Net Core 3.1.

# **Specyfikacja wymagań.**

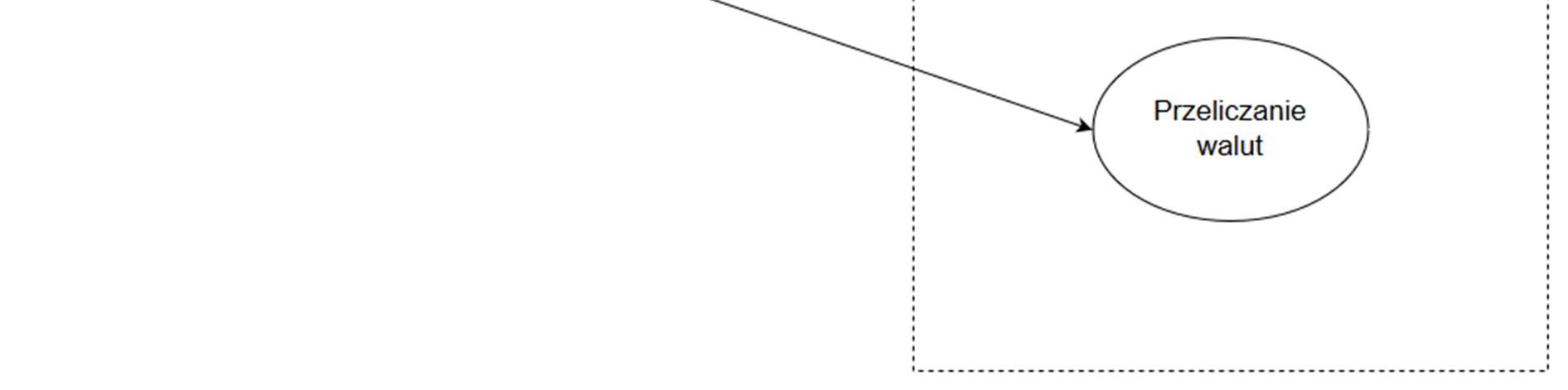
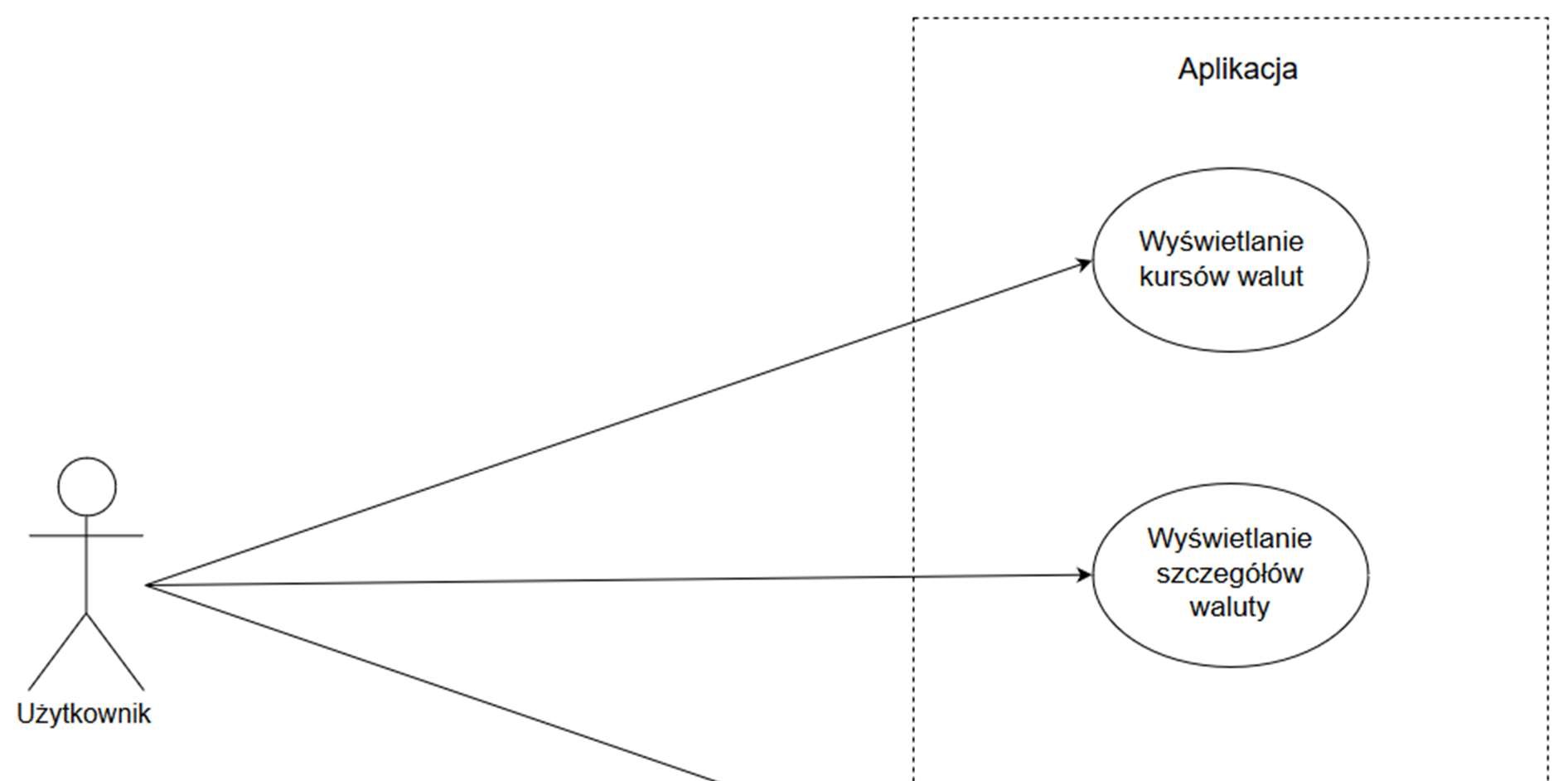
Lista wymagań funkcjonalnych:

* aplikacja ma pobierać aktualną listę akcji spółek giełdowych wraz z ich ostatnią zanotowaną ceną,
* aplikacja ma umożliwiać odczyt aktualnych cen akcji,
* aplikacja ma umożliwiać wyświetlanie szczegółów ceny w przedziale dziennym i godzinowym,
* aplikacja ma udostępniać możliwość zapisania spółki jako ulubionej na prywatnej liście użytkownika,
* aplikacja powinna zapobiegać nieuprawnionemu dostępowi do zabezpieczonej części aplikacji.

Lista wymagań niefunkcjonalnych:

* aplikacja ma oferować możliwość utworzenia konta użytkownika.
* aplikacja ma być aplikacją webową, możliwą do uruchomienia na każdej nowoczesnej przeglądarce internetowej.
* interfejs graficzny składać się ma z bocznego menu nawigacyjnego oraz części głównej aplikacji,
* aplikacja do swojego działania wymagać będzie zainstalowanego na maszynie klienckiej przeglądarki internetowej,
* wymagane jest połączenie z Internetem.

* 1. Diagram przypadków użycia.



* 1. Harmonogram realizacji projektu.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Lp. | Zadanie | Rozpoczęcie | Czas trwania | Zakończenie | Stan |
| zad. 1 | Zapoznanie z językiem C# oraz środowiskiem Visual Studio. | 01.04.2018 | 7 | 07.04.2018 | Zakończony |
| zad. 2 | Określenie funkcji pisanego programu, wraz z próbami implementacji danych funkcji. | 08.04.2018 | 5 | 12.04.2018 | Zakończony |
| zad. 3 | Projektowanie interfejsu graficznego aplikacji. | 13.04.2018 | 10 | 22.04.2018 | Zakończony |
| zad. 4 | Logika biznesowa aplikacji. | 23.04.2018 | 20 | 12.05.2018 | Zakończony |
| zad. 5 | Testy działania. | 13.05.2018 | 3 | 15.05.2018 | Zakończony |
| zad. 6 | Poprawki. | 16.05.2018 | 3 | 18.05.2018 | Zakończony |
| zad. 7 | Stworzenie dokumentacji końcowej. | 19.05.2018 | 5 | 23.05.2018 | Zakończony |

01.04.2018

06.04.2018

11.04.2018

16.04.2018

21.04.2018

26.04.2018

01.05.2018

06.05.2018

11.05.2018

16.05.2018

21.05.2018

zad. 1

zad. 2

zad. 3

zad. 4

zad. 5

zad. 6

zad. 7

* 1. Opis techniczny projektu.

Projekt został zrealizowany w technologii Windows Presentation Foundation (WPF), natomiast logika działania aplikacji została napisana w języku C#.

Głównym celem podczas realizacji projektu było całkowite oddzielenie warstwy odpowiedzialnej za wyświetlenie interfejsu graficznego od warstwy danych. Zastosowanie znalazł więc wzorzec architektoniczny „Model-View-ViewModel”. Wzorzec ten, w przeciwieństwie do wzorców MVC oraz MVP, najlepiej nadaje się do implementacji w aplikacjach wykorzystujących „binding” danych.

Aby móc zaimplementować wzorzec MVVM, w tym projekcie posłużono się frameworkiem „MVVM Light Library”. Biblioteka ta dostarcza kilka gotowych funkcji, które w znaczący sposób ułatwiają proces wymiany danych pomiędzy warstwami: View oraz ViewModel.

Dzięki zastosowaniu tego wzorca, „code-behind” Widoku nie posiada praktycznie żadnych metod z nim powiązanych. Wszystkie wartości, które widok potrzebuje, są pobierane z publicznych właściwości w warstwie ViewModel. To ta warstwa jest odpowiedzialna za obróbkę i przygotowanie danych do wyświetlenia, które zostały zmapowanie przez klasę usług znajdującą się w warstwie Modelu.

Dane dotyczące aktualnego kursu walut pobierane są ze strony Narodowego Banku Polskiego. NBP udostępnia wygodne API w formacie JSON lub XML. W tym projekcie postawiono na format JSON, z którego dane można bezproblemowo zmapować na obiekt danych podobny do tych z mapowań ORM’ów dla baz danych. Do wykonania mapowania użyto kolejnego frameworka, bardzo popularnego wśród użytkowników .Net: „Newtonsoft.Json”.

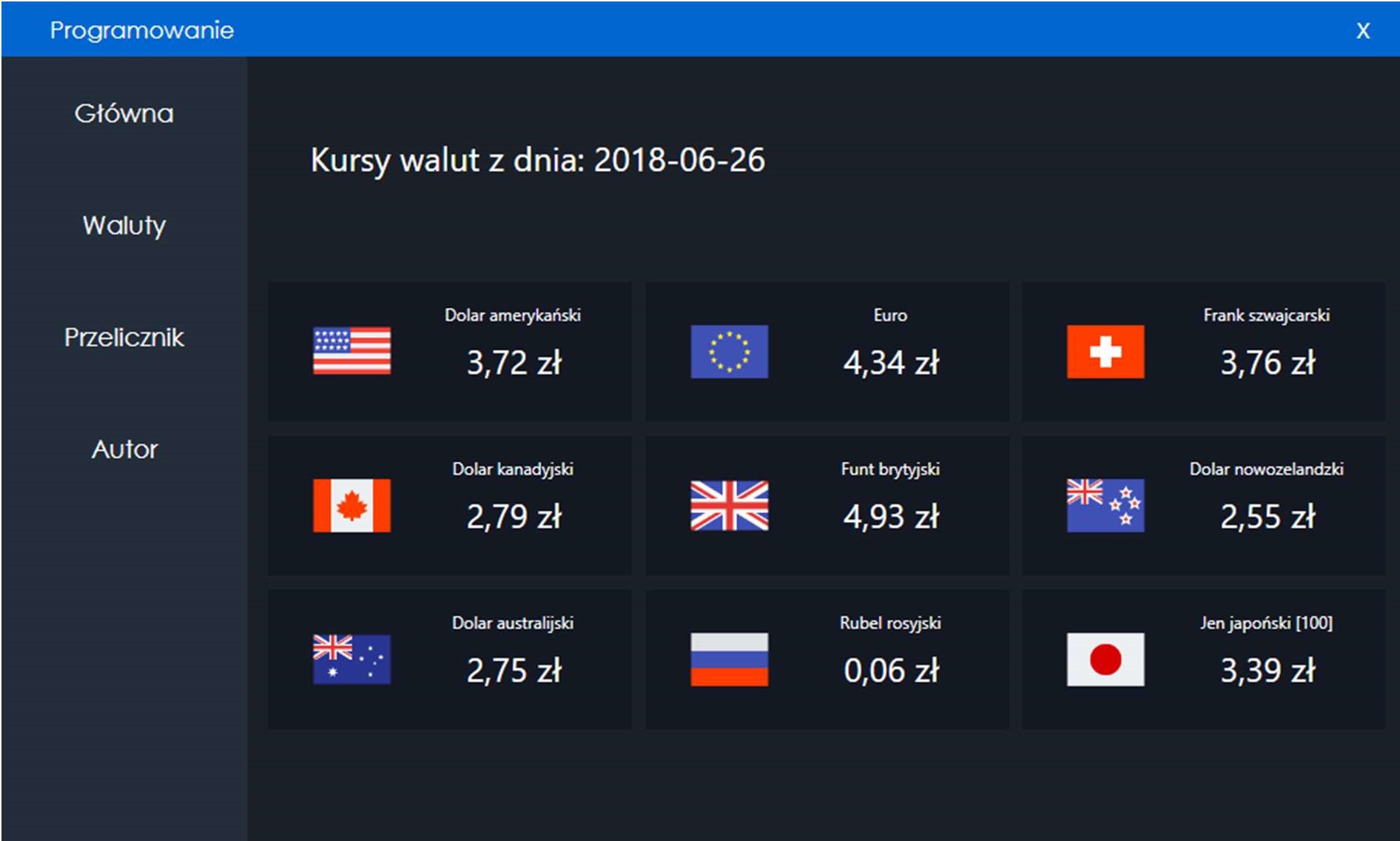
Interfejs graficzny programu to pojedyncze okno stworzone w stylu „Modern UI”. Większym nakładem pracy, jednak celowo zrezygnowano z wyglądu typowo „okienkowego” znanego z systemów Windows. Okno zostało podzielone na 3 części:

* pasek tytułu,
* przyciski nawigacyjne,
* „grid” z zawartością danej funkcjonalności.

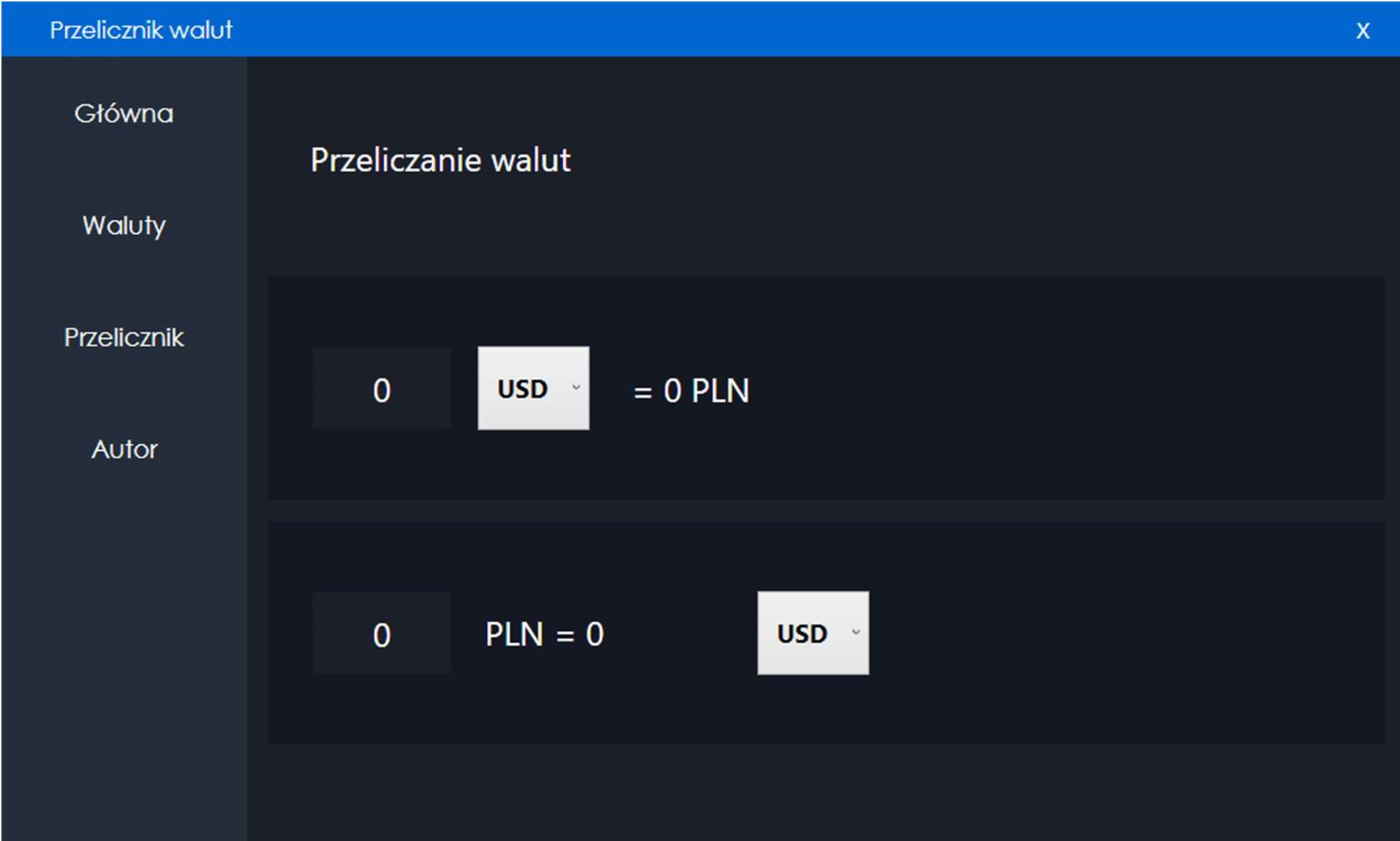
Przyciski nawigacyjne oraz przycisk „X” to zwykłe Button’y dostarczane przez środowisko, lecz został zmieniony ich styl i zachowanie z domyślnego na niestandardowy.

Dzięki zastosowaniu nietypowego stylu okna, twórca interfejsu graficznego ma większe możliwości dostosowania go dla danej grupy odbiorców. Warto zauważyć, że nietypowy i kolorowy interfejs graficzny bardziej przyciąga wzrok użytkownika, ewentualnego klienta, niż domyślny jasnoszary interfejs okienkowy.

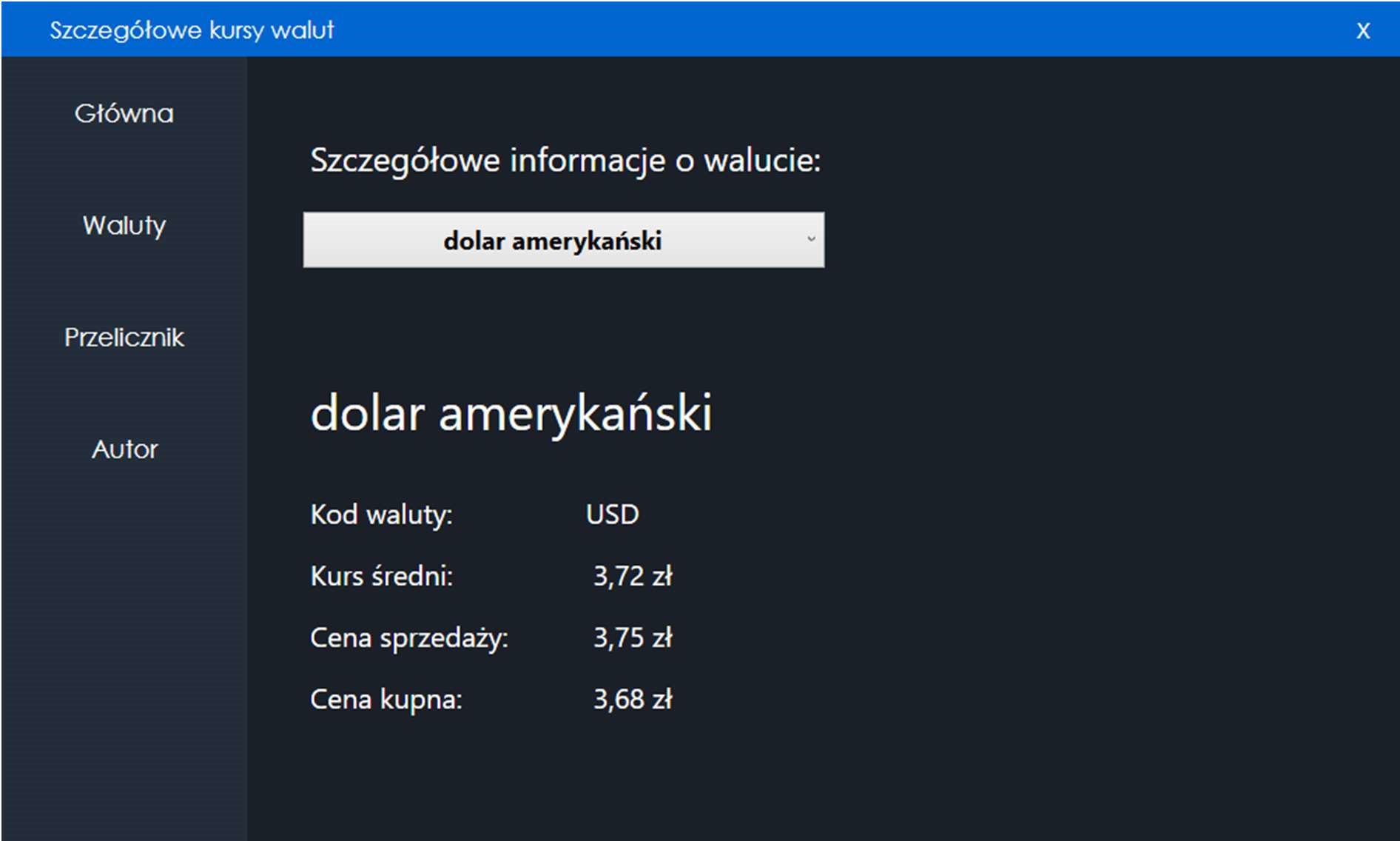
1. Prezentacja warstwy użytkowej projektu.



Rys. Strona główna programu.



Rys. Strona przelicznika walut.



Rys. Strona danych szczegółowych o danej walucie

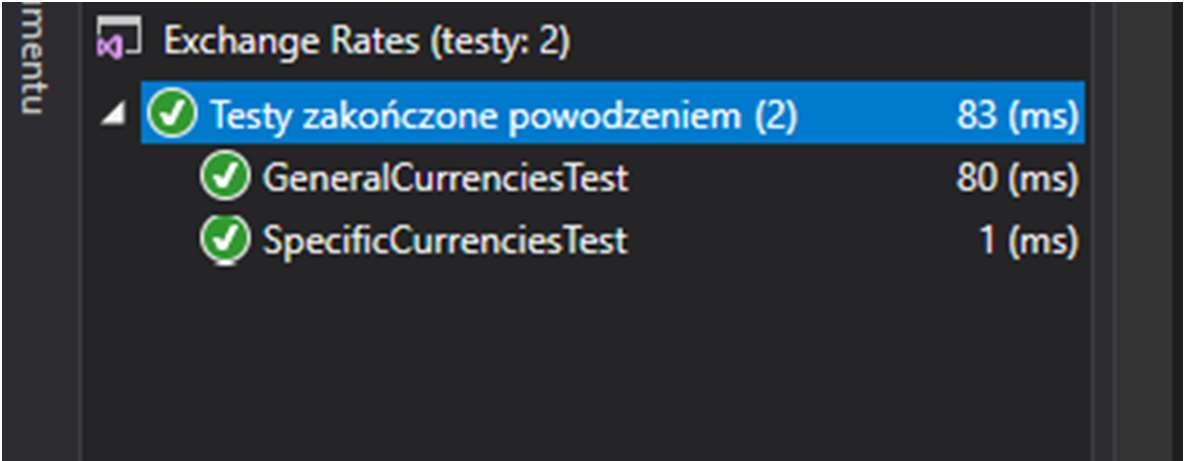
1. Raporty z testów.

Podział aplikacji na Widok, Model i warstwę pośredniczącą WidokModel, spowodował że wszystkie dane są przetwarzane przez warstwę ViewModel. Metody występujące w tej warstwie działają na zmiennych wewnętrznych klasy, jednocześnie nie zwracają żadnych danych, które można by testować.

Jedynymi metodami możliwymi do testowania są te pobierające dane z serwisów zewnętrznych, czyli:

* GetGeneralCurrencies,
* GetSpecificCurrencies

Poprawnie działająca metoda zwraca za pomocą „callback” zmapowany obiekt danych oraz pusty obiekt błędów (null). Jeśli natomiast występuje błąd w tej metodzie, obiekt danych jest pusty, natomiast obiekt błędów posiada zawartość z informacją o problemie jaki wystąpił.



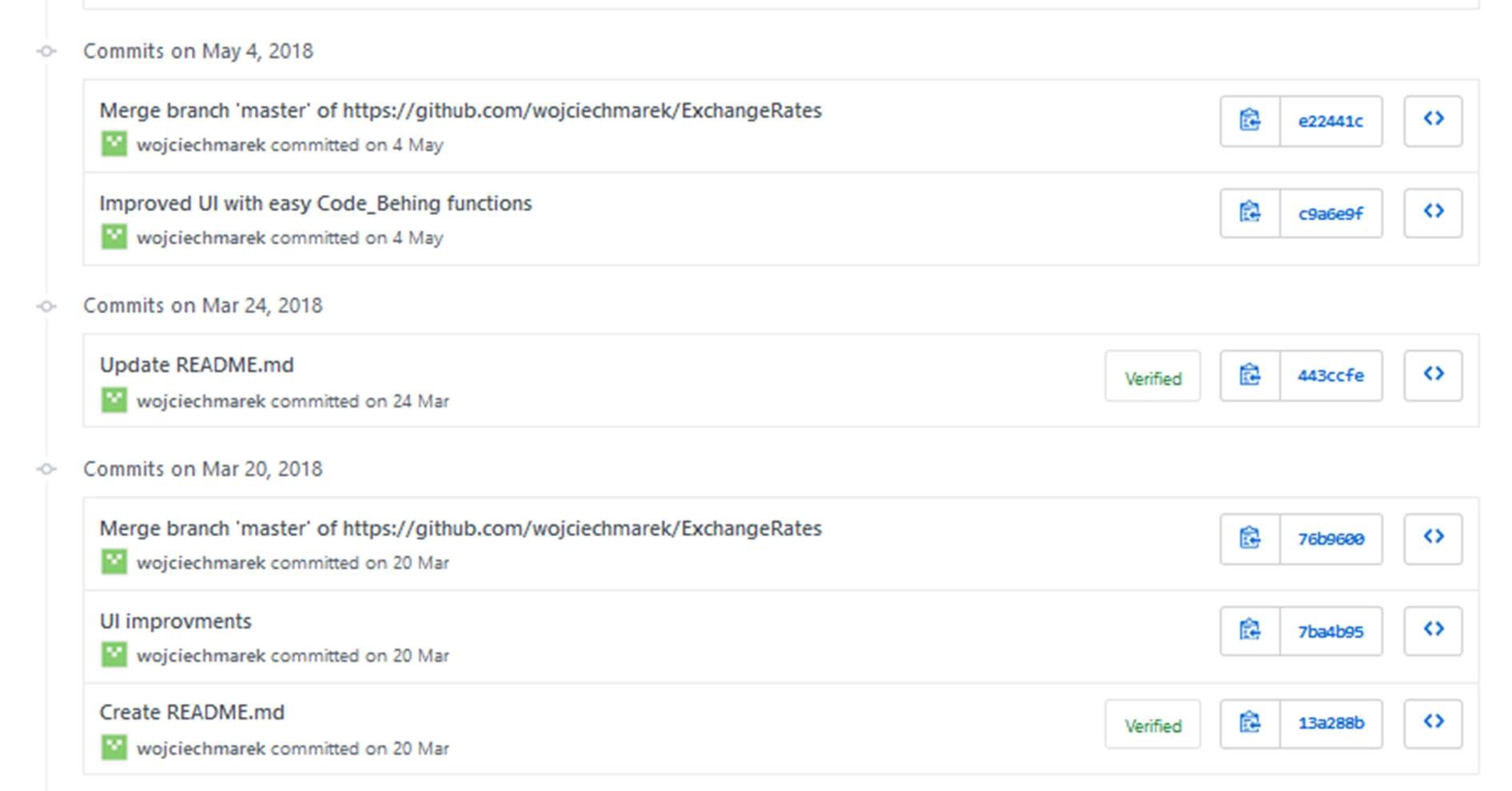
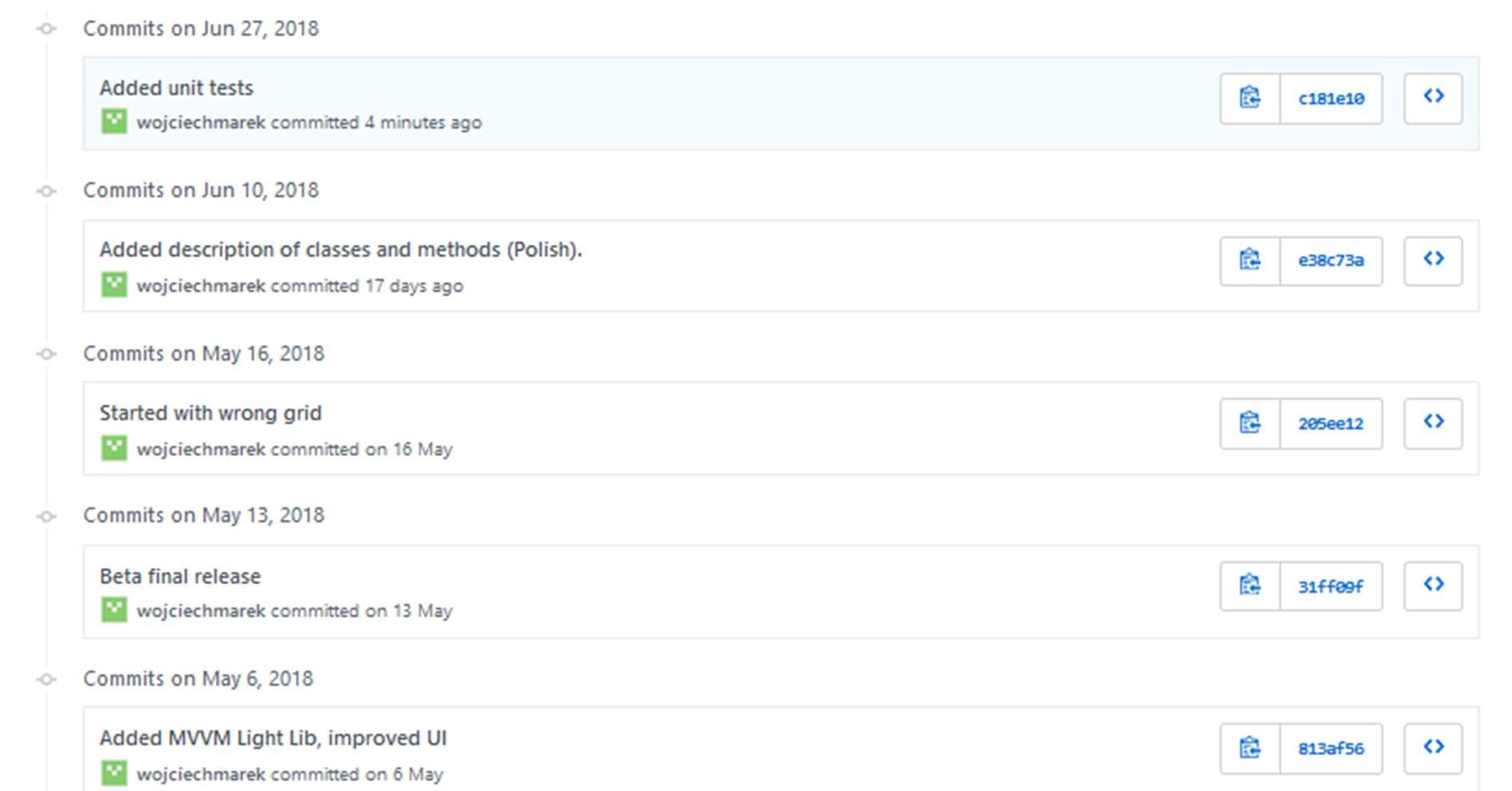
Rys. Zrzut ekranu przeprowadzonych testów.

* 1. System kontroli wersji.

Projekt został realizowany za pomocą systemu kontroli wersji GitHub. Commit’y były wykonywane w głównej mierze po implementacji kluczowych funkcjonalności w powstającym programie.

Adres repozytorium z projektem i testami jednostkowymi pod adresem:

https://github.com/wojciechmarek/ExchangeRates



Rys. Historia Commit’ów projektu.

* 1. Materiały źródłowe.
* https://stackoverflow.com/
* http://www.mvvmlight.net/
* https://4programmers.net/
* https://docs.microsoft.com/