WYŻSZA SZKOŁA
INFORMATYKI i ZARZĄDZANIA
z siedzibą w Rzeszowie

**KOLEGIUM INFORMATYKI STOSOWANEJ**

**Kierunek: INFORMATYKA**

Piotr Pająk
Nr albumu studenta: 71348

# *Aplikacja „Bankomat”*

**Projekt**

**Rzeszów 2026**

# Spis treści

.

# Wstęp

Przedmiotem niniejszego projektu jest opracowanie i implementacja autorskiego systemu informatycznego symulującego pracę bankomatu. Aplikacja została stworzona w środowisku .NET przy użyciu języka C# i integruje się z relacyjną bazą danych SQL Server. System pozwala na realizację podstawowych operacji finansowych, takich jak autoryzacja użytkownika, wypłata środków oraz sprawdzanie stanu salda.

Głównym celem pracy jest przedstawienie procesu projektowania i tworzenia aplikacji. Zakres dokumentacji obejmuje opis architektury systemu, model relacyjny bazy danych, analizę zastosowanych algorytmów walidacji oraz prezentację wyników testów funkcjonalnych.

Na repozytorium projektu, umieszczony został plik README.docx z instrukcją do uruchomienia programu.

.

# Rozdział 1

# Opis założeń i cele projektu

Celem projektu było zaprojektowanie i zaimplementowanie aplikacji symulującej działanie bankomatu przy użyciu paradygmatu programowania obiektowego. Głównym założeniem było stworzenie systemu, który w bezpieczny sposób zarządza stanem konta użytkownika, autoryzacją oraz operacjami finansowymi.

## 1.1 Wymagania projektu

### Wymagania funkcjonalne:

- **Zarządzanie kartami:** Obsługa różnych typów kart płatniczych (Visa, Mastercard, American Express) z możliwością rozbudowy o uwzględnienienie ich specyficznych systemów weryfikacji.
- **Operacje finansowe:** Możliwość pobierania aktualnego salda konta oraz dokonywania wypłat (z automatyczną aktualizacją danych w bazie).
- **Walidacja danych:** Sprawdzanie poprawności formatu wprowadzanego PIN-u (wymagane 4 cyfry).
- **Mechanizm "Anti-Brute Force":** System dopuszcza 3 próby wprowadzenia poprawnego kodu PIN. Po przekroczeniu limitu sesja jest automatycznie przerywana ze względów bezpieczeństwa.
- **Raportowanie:** Możliwość eksportu listy kont wraz z ich saldami do pliku zewnętrznego w formacie CSV.
- **Inteligentna walidacja wypłaty:** System automatycznie sprawdza dwa warunki przed wydaniem gotówki: czy kwota jest wielokrotnością 10 PLN oraz czy saldo konta jest wystarczające.

### Wymagania niefunkcjonalne:

- **Niezawodność:** System musi stabilnie obsługiwać błędy połączenia z bazą danych oraz sytuacje wyjątkowe (np. próba wypłaty kwoty większej niż dostępne saldo).
- **Architektura:** Kod powinien być napisany tak aby umożliwić jego późniejszą rozbudowę (np. dodanie nowego typu karty nie wymaga modyfikacji istniejącej logiki).
- **Trwałość danych:** Wszystkie informacje o kontach i transakcjach muszą być składowane w relacyjnej bazie danych SQL Server, zapewniając ich spójność.

.

# Rozdział 2

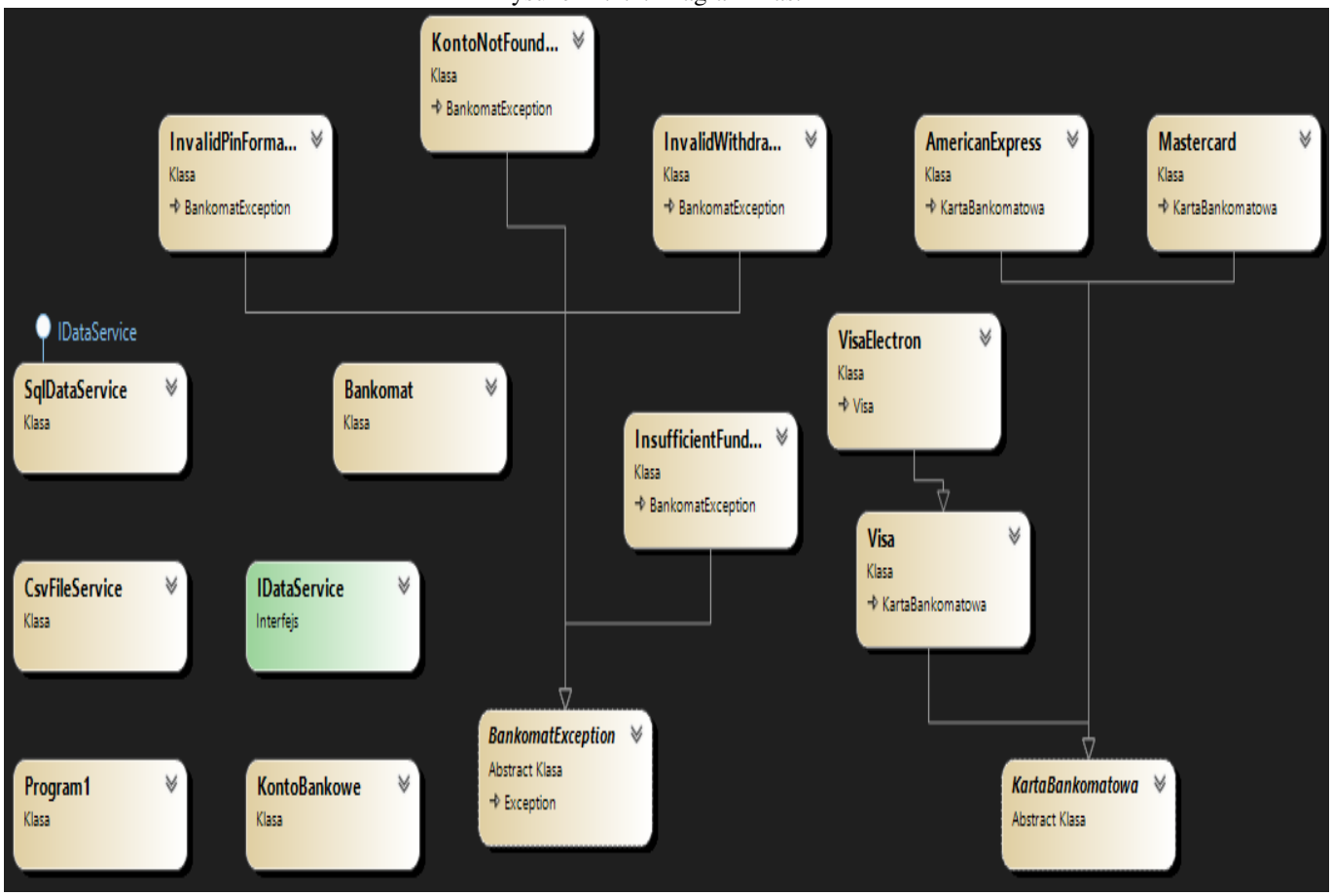
# Opis struktury projektu

## 2.1 Architektura systemu, diagram klas i komponenty struktury

Projekt został zaprojektowany w oparciu o architekturę warstwową, wykorzystującą wzorce projektowe typowe dla programowania obiektowego. Głównym założeniem systemu jest separacja logiki biznesowej od mechanizmów składowania danych.

- **Logika Biznesowa (Bankomat.cs):** Odpowiada za zarządzanie stanem sesji i rzucanie wyjątków w przypadku naruszenia zasad bankowych.
- **Interfejs Użytkownika (Program.cs):** Odpowiada za prezentację danych, obsługę kolorów komunikatów oraz przechwytywanie wyjątków (try-catch) w celu zapewnienia ciągłości działania programu.

Rysunek 2.1.1: Diagram klas.



### Główne komponenty struktury:

- **Hierarchia kart płatniczych:** Sercem systemu jest klasa abstrakcyjna KartaBankomatowa. Zawiera ona logikę walidacji formatu kodu PIN. Wymusza polimorficzne zachowanie poprzez metodę abstrakcyjną WeryfikujDostepnoscSystemu(). Klasy konkretne takie jak Visa, Mastercard oraz AmericanExpress implementują własne mechanizmy weryfikacji.
- **Logika usług (Services):** Za warstwę dostępu do danych odpowiada interfejs IDataService, który definiuje operacje pobierania kont i aktualizacji salda. Implementacja SqlDataService realizuje te zadania w oparciu o komunikację z bazą danych SQL Server.
- **Warstwa modeli:** Klasa KontoBankowe pełni rolę obiektu transferu danych, przechowując informacje o właścicielu, saldzie oraz skrócie kodu PIN (PINHash).

## Opis techniczny:

Aplikacja została zaimplementowana w nowoczesnym środowisku programistycznym, z wykorzystaniem technologii zapewniających stabilność i bezpieczeństwo transakcji.

- **Język programowania:** C# (platforma .NET 8.0).
- **Środowisko programistyczne (IDE):** Microsoft Visual Studio 2022.
- **System bazodanowy:** Microsoft SQL Server (wersja Express). Dane są przechowywane w relacyjnej bazie danych o nazwie SystemBankowy.
- **Komunikacja z bazą danych:** Wykorzystano bibliotekę Microsoft.Data.SqlClient. Komunikacja odbywa się za pomocą klasy SqlDataService, która realizuje zapytania SQL z wykorzystaniem parametrów, co zabezpiecza aplikację przed atakami typu SQL Injection.
- **Dodatkowe narzędzia:** Klasa CsvFileService umożliwia eksportowanie danych do formatu CSV przy użyciu klasy StringBuilder, co pozwala na generowanie raportów z zestawieniem kont i sald.

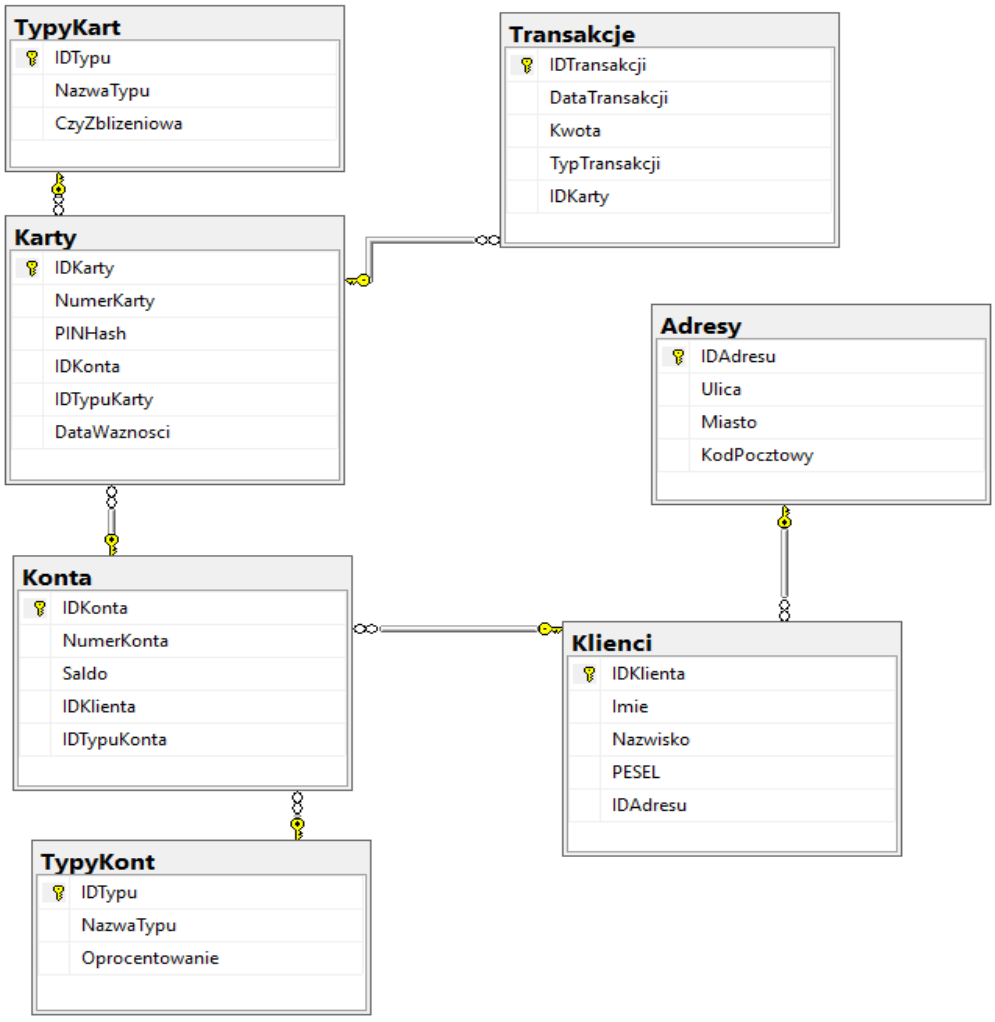
## 2.2 Zarządzanie danymi i Baza Danych

Dane w systemie są zorganizowane w sposób relacyjny. Główna logika pobierania danych opiera się na łączeniu (JOIN) trzech kluczowych tabel.

- **Klienci** – przechowuje dane osobowe właścicieli kont.
- **Konta** – zawiera informacje o numerach kont i aktualnym saldzie.
- **Karty** – przechowuje kody PIN oraz powiązania kart z kontami.

Poniższy diagram przedstawia graficzną strukturę bazy danych, uwzględniając tabele, klucze główne (PK) oraz powiązania za pomocą kluczy obcych (FK).

Rysunek 2.4.1: Baza danych - diagram ERD.




Do utworzenia oraz zarządzania bazą danych wykorzystany został program SQL Server Management Studio 21. Skrypt do utworzenia bazy danych wraz z instrukcją został umieszczony na repozytorium projektu.

## 2.3 Minimalne wymagania

### Minimalne wymagania sprzętowe:

- **Procesor:** Dwurdzeniowy, o taktowaniu min. 2.0 GHz.
- **Pamięć RAM:** 4 GB.
- **Miejsce na dysku:** 200 MB (na aplikację i lokalną instancję bazy danych SQL Express).

### Minimalne wymagania programowe:

- **System operacyjny:** Windows 10 lub nowszy (zalecany ze względu na pełne wsparcie dla SQL Server).
- **Środowisko uruchomieniowe:** .NET Core 6.0 / .NET 8.0
- **Silnik bazy danych:** Microsoft SQL Server 2019 (lub nowszy) w wersji Express
- **Narzędzia administracyjne (Zalecane):** SQL Server Management Studio (do uruchomienia skryptu tworzącego bazę danych)
- **Środowisko programistyczne (do edycji):** Visual Studio 2022

## 2.4 Obsługa wyjątków

System posiada dedykowany moduł obsługi błędów oparty na klasie bazowej BankomatException. Zastosowano własne klasy wyjątków, takie jak:

- **BankomatException** - Klasa bazowa, pozwalająca na scentralizowaną obsługę błędów biznesowych.
- **KontoNotFoundException** – Rzucany w przypadku błędów w zapytaniach bazodanowych.
- **InsufficientFundsException** – Rzucany, gdy użytkownik próbuje wypłacić kwotę większą niż dostępne saldo.
- **InvalidPinFormatException** – Rzucany, gdy format PIN-u wprowadzany przez użytkownika jest niepoprawny.
- **InvalidWithdrawalAmountException** - Rzucany, gdy kwota nie jest wielokrotnością 10 PLN lub jest mniejsza/równa 0.

.

# Rozdział 3

# Harmonogram realizacji projektu

## 3.1 Etapy realizacji prac

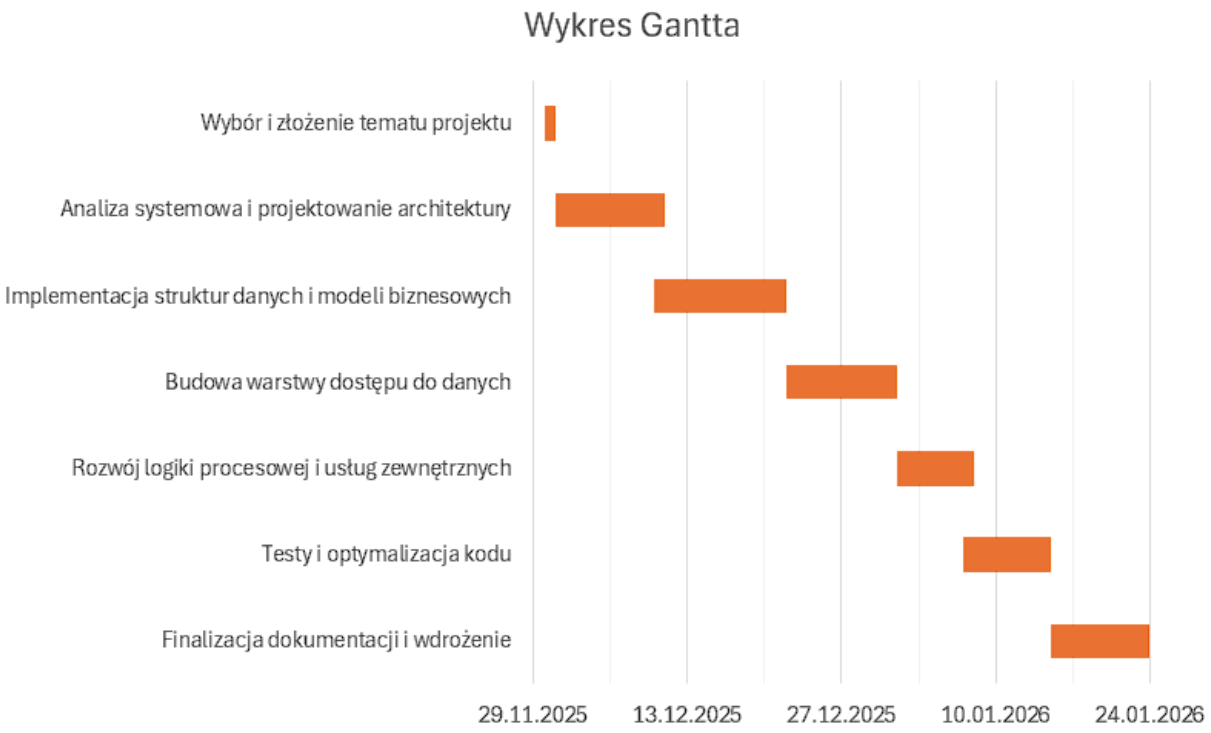
Proces tworzenia systemu bankomatowego został podzielony na pięć głównych etapów, z których każdy kończył się weryfikacją poprawności działania danego modułu.

- **Wybór i złożenie tematu projektu:** Analiza zaproponowanych tematów. Złożenie tematu na forum moodle.
- **Analiza systemowa i projektowanie architektury:** Opracowanie specyfikacji wymagań, zaprojektowanie relacyjnego modelu bazy danych oraz zdefiniowanie przepływu procesów biznesowych (autoryzacja, wypłata).
- **Implementacja struktur danych i modeli biznesowych:** Stworzenie klas KontoBankowe, KartaBankomatowa oraz dedykowanego systemu obsługi wyjątków BankomatException.
- **Budowa warstwy dostępu do danych:** Konfiguracja środowiska SQL Server Express. Implementacja serwisu SqlDataService wraz z logiką CRUD, zapewniającą bezpieczną komunikację z bazą danych poprzez interfejs IDataService.
- **Rozwój logiki procesowej i usług zewnętrznych:** Implementacja zaawansowanej walidacji kodów PIN (mechanizm 3 prób) oraz modułu CsvFileService odpowiedzialnego za generowanie raportów i eksport danych.
- **Testy i optymalizacja kodu:** Przeprowadzenie testów integracyjnych (komunikacja aplikacja-baza), weryfikacja obsługi sytuacji błędnych.
- **Finalizacja dokumentacji i wdrożenie:** Opracowanie pełnej dokumentacji technicznej i instrukcji użytkownika oraz przygotowanie wersji dystrybucyjnej aplikacji (plik .exe).

## 3.2 Diagram Gantta

Poniższy wykres obrazuje ramy czasowe poszczególnych zadań.

Rysunek 3.2.1: Diagram Gantta.




### Podsumowanie przebiegu prac

Prace przebiegały zgodnie z założonym planem. Najwięcej czasu poświęcono na etap 2 i 3, ze względu na konieczność zapewnienia pełnej spójności między obiektami w języku C# a relacyjną strukturą bazy danych.

.

# Rozdział 4

# Repozytorium i kontrola wersji

## 4.1 System kontroli wersji

W celu zapewnienia bezpieczeństwa kodu źródłowego oraz możliwości śledzenia historii zmian, w projekcie wykorzystano system kontroli wersji Git. Wybór ten podyktowany był standardami rynkowymi oraz doskonałą integracją z systemami hostującymi repozytoria zdalne.

### Struktura repozytorium zdalnego

Kod projektu został umieszczony w publicznym repozytorium na platformie GitHub. Dzięki temu zapewniono:

- **Archiwizację:** Pełna kopia zapasowa wszystkich wersji plików projektu.
- **Dostępność:** Możliwość pracy nad kodem z różnych stacji roboczych bez ryzyka utraty spójności danych.

Repozytorium będzie dopstępne publicznie przez rok od daty dnia złożenia projektu. Adres repozytorium: https://github.com/AsadiXX/Bankomat-OOP

.

# Rozdział 5

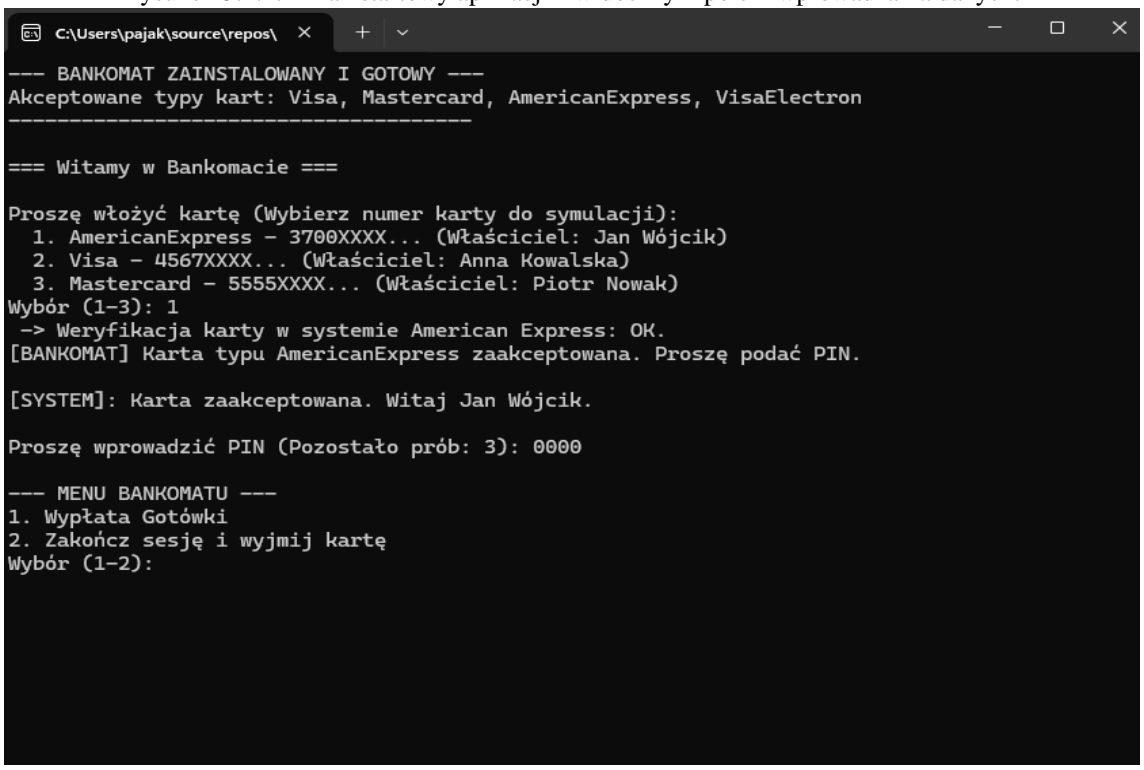
# Prezentacja warstwy użytkowej

W niniejszym rozdziale przedstawiono interfejs użytkownika oraz sposób działania aplikacji.

## 5.1 Logowanie i autoryzacja

Po uruchomieniu programu, użytkownik proszony jest o wybranie danych identyfikacyjnych karty oraz kodu PIN. System na tym etapie wykorzystuje klasę SqlDataService do weryfikacji danych w bazie SQL Server.

Rysunek 5.1.1: Ekran startowy aplikacji z widocznym polem wprowadzania danych.

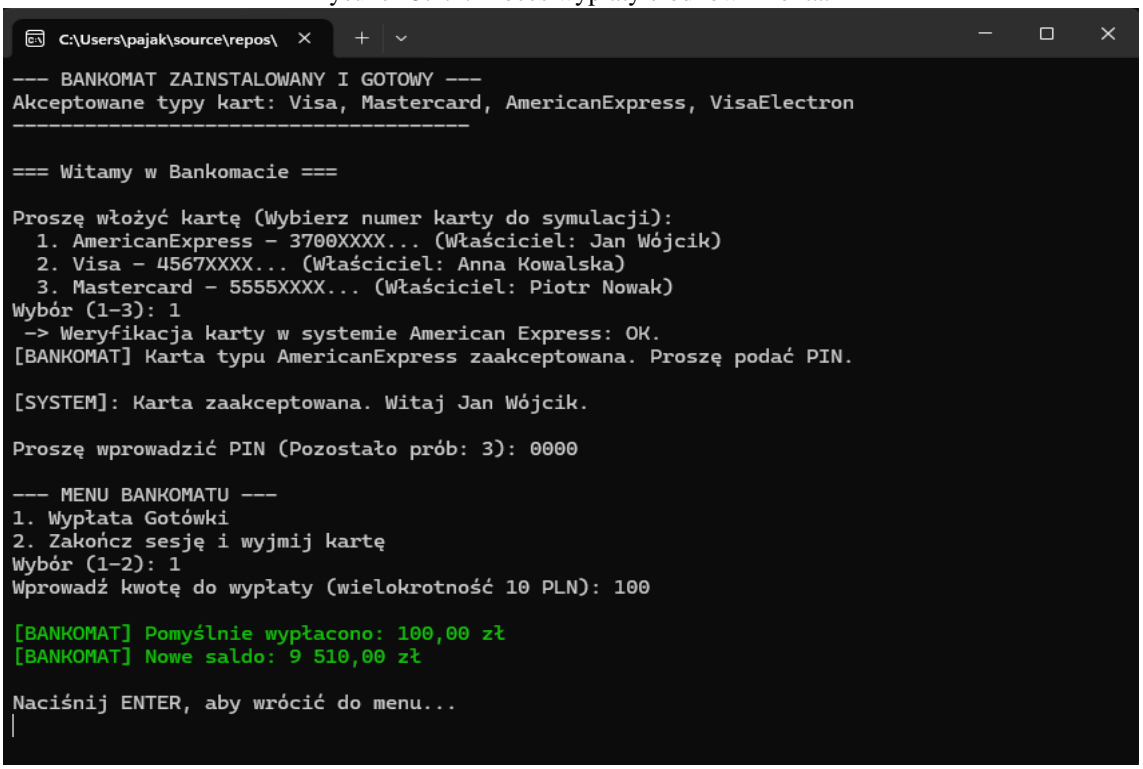



Na powyższym zrzucie widać proces pomyślnej autoryzacji. System rozpoznał typ karty (np. AmericanExpress) i przywitał użytkownika imieniem pobranym z bazy danych.

## 5.2 Obsługa operacji finansowych (Wypłata)

Głównym modułem aplikacji jest panel transakcyjny. Użytkownik może zadeklarować kwotę wypłaty, a system automatycznie sprawdza warunek wystarczających środków.

Rysunek 5.2.1: Proces wypłaty środków z konta.

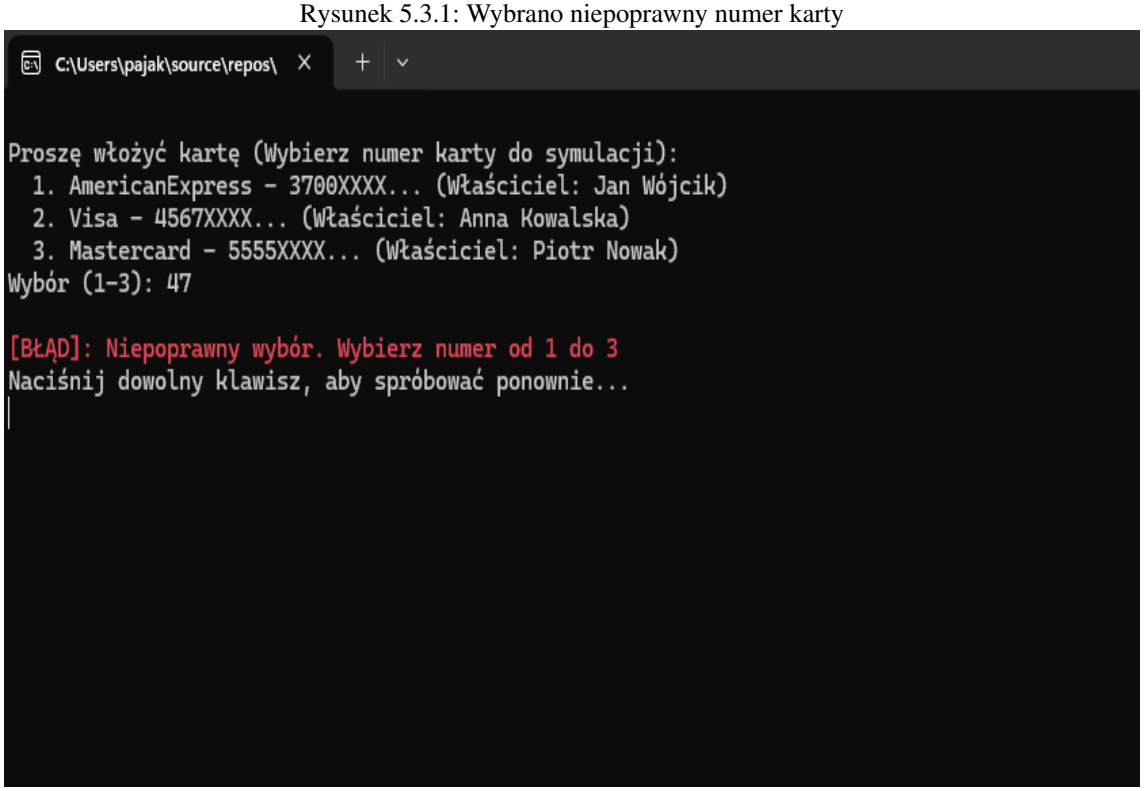



Zrzut ekranu prezentuje scenariusz, w którym saldo jest wystarczające. Widać komunikat o pomyślnym pobraniu gotówki oraz informację o nowym stanie konta po aktualizacji w bazie danych.

## 5.3 Obsługa sytuacji wyjątkowych (Błędy)

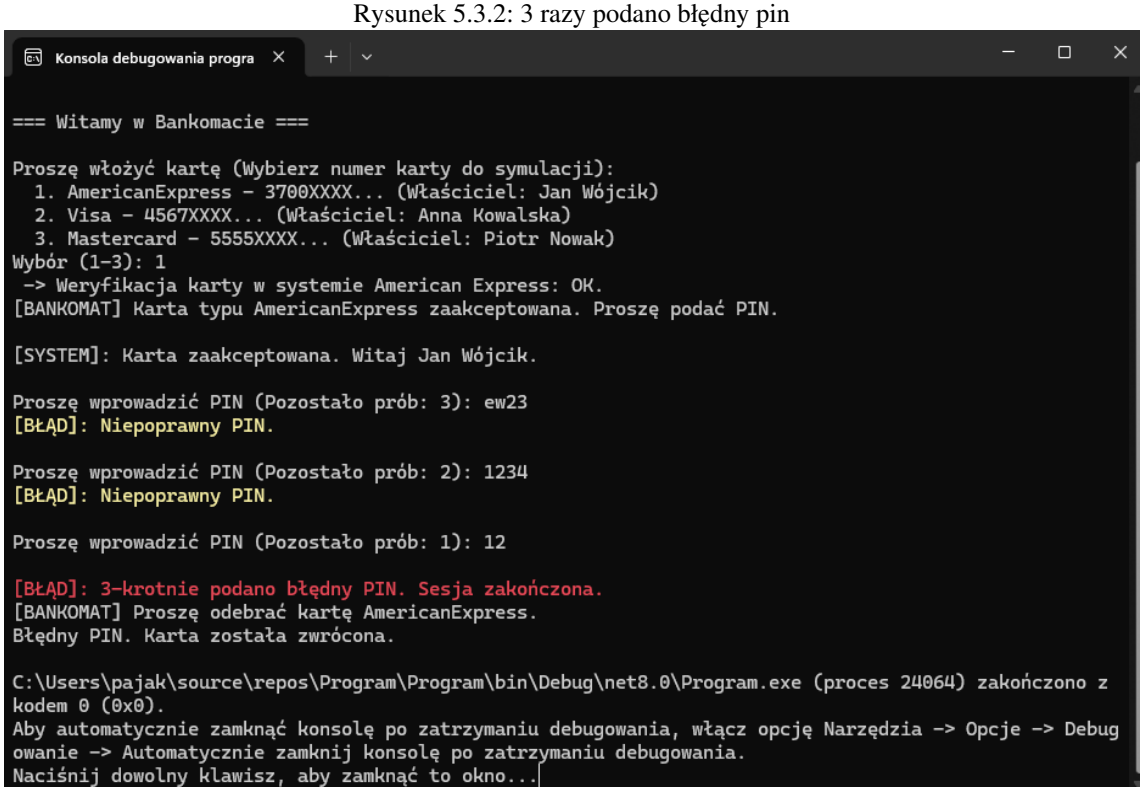
Aplikacja została zaprojektowana tak, aby nie przerywać działania w momencie wystąpienia błędu.

Rysunek 5.3.1: Wybrano niepoprawny numer karty



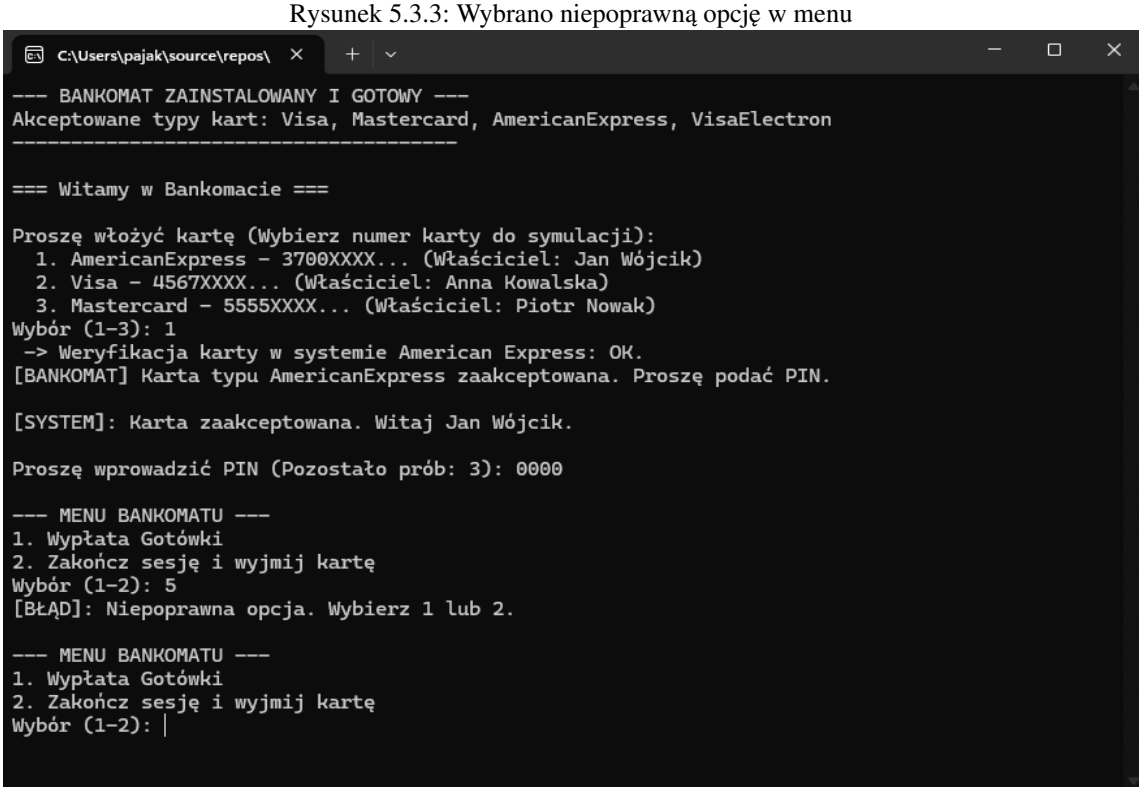

Na zrzucie ekranu widoczny jest przypadek w którym użytkownik wybrał błędną kartę z listy.

Rysunek 5.3.2: 3 razy podano błędny pin



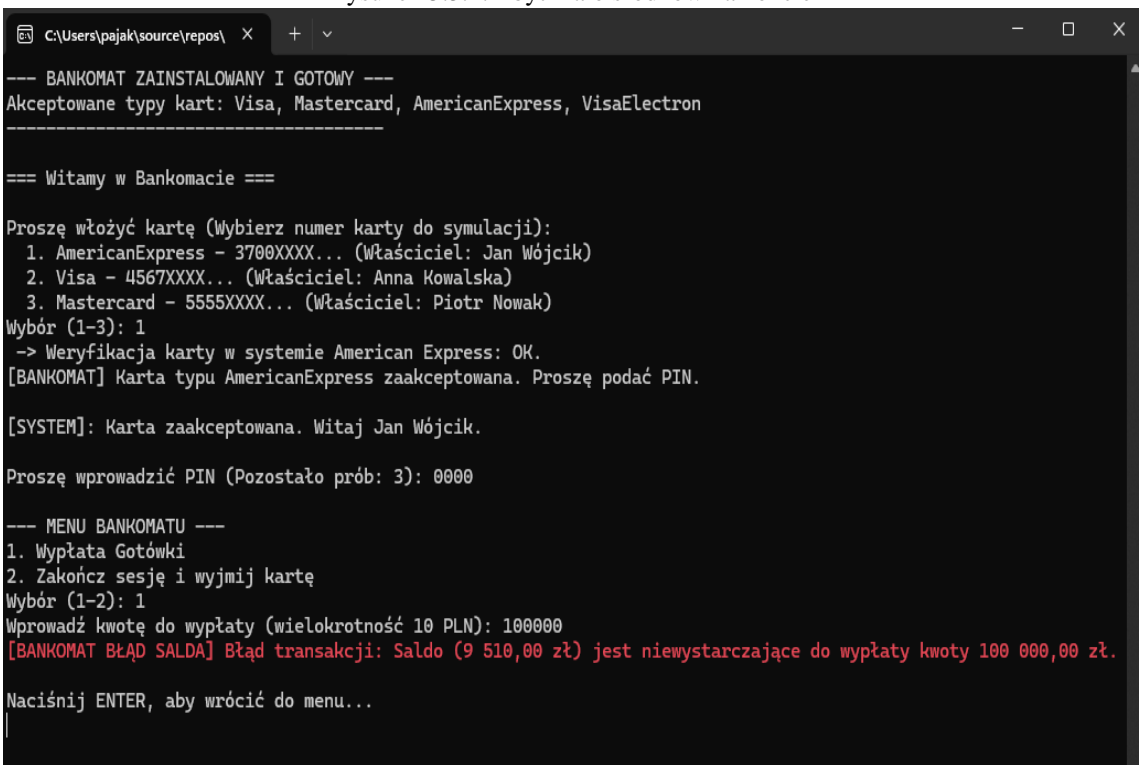

Na zrzucie ekranu widoczny jest przypadek w którym użytkownik 3 razy podał błędny pin.

Rysunek 5.3.3: Wybrano niepoprawną opcję w menu



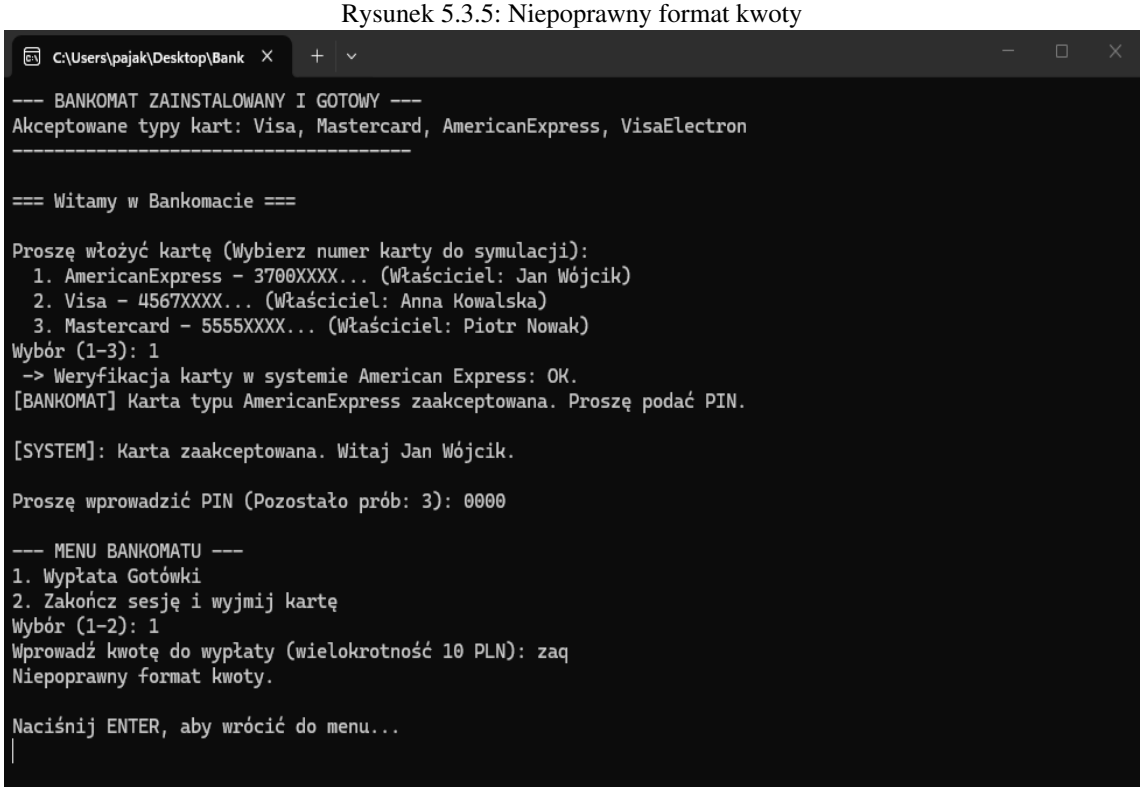

Gdy użytkownik wybierze niepoprawną opcje z menu, program wyświetli o tym informację.

Rysunek 5.3.4: Zbyt mało środków na koncie




Na zrzucie ekranu widoczny jest przypadek w którym użytkownik próbuję wypłacić kwotę większą niż posiada na swoim koncie.

Rysunek 5.3.5: Niepoprawny format kwoty

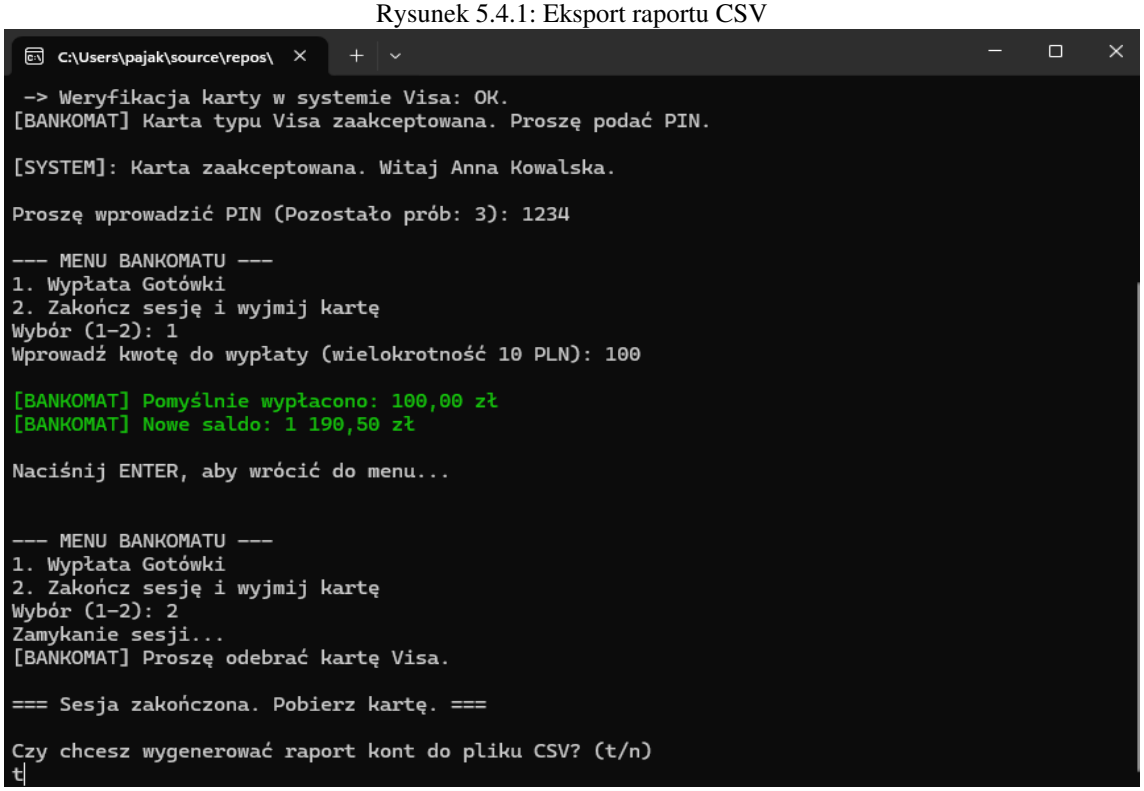



Na zrzucie ekranu widoczny jest przypadek w którym użytkownik podał niepoprawny format kwoty.

## 5.4 Eksport danych do pliku CSV

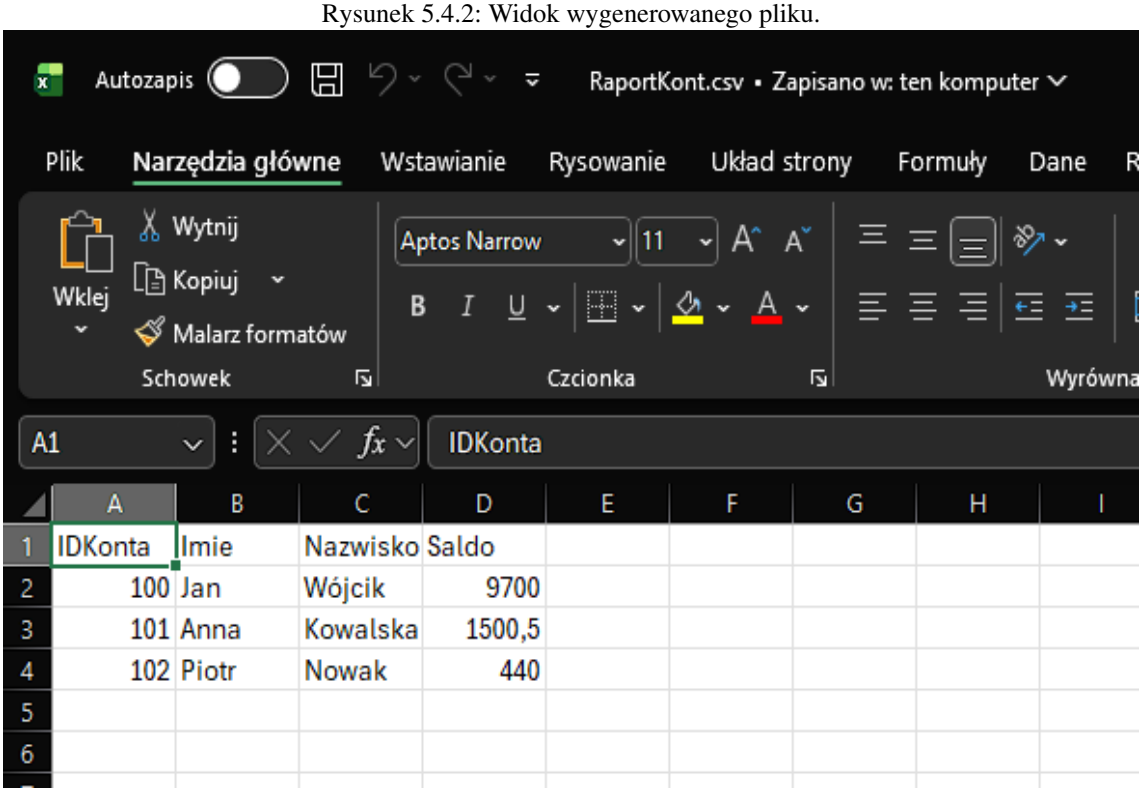
Po zakończeniu operacji wypłaty środków, system pozwala na wygenerowanie raportu wszystkich kont zapisanych w bazie. Jest to funkcja administracyjna realizowana przez klasę CsvFileService.

Rysunek 5.4.1: Eksport raportu CSV




Na końcu sesji użytkownik ma wybór wygenerowania raportu do pliku CSV

Rysunek 5.4.2: Widok wygenerowanego pliku.




Zrzut ekranu pokazuje strukturę pliku wynikowego z nagłówkami (IDKonta, Imie, Nazwisko, Saldo).

.

# Rozdział 6

# Podsumowanie

## 6.1 Wnioski z realizacji projektu

Projekt systemu obsługi bankomatu został zrealizowany zgodnie ze wszystkimi założeniami określonymi w rozdziale trzecim. Proces implementacji pozwolił na praktyczne zastosowanie zaawansowanych mechanizmów programowania obiektowego w języku C#

### Kluczowe osiągnięcia projektu to:

- **Poprawna implementacja OOP:** Wykorzystanie klas abstrakcyjnych (KartaBankomatowa) oraz dziedziczenia (Visa, Mastercard) pozwoliło na stworzenie przejrzystej struktury kodu.
- **Integracja z bazą danych:** Skuteczne połączenie aplikacji z serwerem SQL Server Express przy użyciu biblioteki SqlClient umożliwiło trwałe przechowywanie danych i bezpieczne wykonywanie transakcji.
- **System obsługi błędów:** Szczególny nacisk położono na odporność aplikacji na błędy użytkownika. Dzięki zastosowaniu pętli walidacyjnych oraz metod TryParse, aplikacja eliminuje ryzyko niekontrolowanego zamknięcia procesu przy podaniu nieprawidłowych znaków lub kwot.

## 6.2 Dalsze prace rozwojowe

Mimo że aplikacja spełnia wszystkie wymagania funkcjonalne, istnieje kilka obszarów, o które można ją rozbudować w przyszłości:

- **Interfejs graficzny (GUI):** Migracja z interfejsu konsolowego na okienkowy.
- **Logowanie transakcji:** Dodanie tabeli Transakcje w bazie danych przechowującej historię wszystkich operacji (tzw. logi transakcyjne).
- **Szyfrowanie zaawansowane:** Zastosowanie silniejszych algorytmów kryptograficznych (np. AES) do ochrony danych wrażliwych.
- **Baza danych:** Pełna migracja zarządzania bazą danych do skryptów migracyjnych aby wyeliminować konieczność ręcznego uruchamiania skryptów SQL w SSMS.

.

# Bibliografia


[1] Matulewski J., *C#: lekcje programowania: praktyczna nauka programowania dla platform .NET i .NET Core*, Helion, Gliwice 2021.

[2] Albahari J., Johannsen E., *C# 8.0 w pigułce*, Helion, Gliwice, 2021.

[3] Miles R.S., *C#: zacznij programować!*, Helion, Gliwice, 2020.