Uniwersytet Rzeszowski
Kolegium Nauk Przyrodniczych
Instytut Informatyki

Praca projektowa programowanie obiektowe

System symulujący działanie Bankomatu

Prowadzący:
mgr inż. Ewa Żesławska

Autor:
Dawid Olko
nr albumu: 125148
Kierunek: Informatyka, grupa lab 3

Rzeszów 2024

Spis treści

1. Opis założeń projektu

W dzisiejszych czasach, gdy sektor bankowy ewoluuje z niespotykaną dotąd prędkością, napotykamy na wyzwania związane z adaptacją technologii i utrzymaniem wysokiej jakości obsługi klienta. Jednym z kluczowych elementów tego ekosystemu są bankomaty, stanowiące ważny punkt kontaktu z klientem. Niestety, wraz ze wzrostem złożoności operacji bankowych i rosnącymi oczekiwaniami klientów, bankomaty często stają się obiektem błędów operacyjnych, awarii, a nawet przestępczości. Te problemy, choć nie związane bezpośrednio z główną działalnością banku, mają istotny wpływ na wizerunek instytucji finansowej, satysfakcję klientów oraz efektywność operacyjną.
Opracowany innowacyjny system symulacji pracy bankomatu. Jest to narzędzie, które nie tylko pozwala na dokładne modelowanie i analizę różnorodnych scenariuszy działania bankomatów, ale także służy jako platforma szkoleniowa dla pracowników. Dzięki temu systemowi możliwe jest wykrywanie i zapobieganie potencjalnym problemom, a także optymalizacja procesów obsługi klienta.

Projekt "Bankomat" to kompleksowa aplikacja GUI, której celem jest symulacja operacji wykonywanych na rzeczywistym bankomacie. Zapewnia użytkownikom interfejs graficzny, za pośrednictwem którego mogą weryfikować tożsamość za pomocą karty elektronicznej i numeru PIN, a następnie przeprowadzać podstawowe operacje bankowe. Aplikacja obsługuje karty Visa, American Express, Visa Electron, Mastercard, Diners Club, Japan Credit Bureau.

Aplikacja jest zbudowana w oparciu o Java Swing w środowisku IntelliJ IDEA i wykorzystuje MySQL jako system zarządzania relacyjną bazą danych, co umożliwia przechowywanie i zarządzanie danymi w sposób strukturalny i bezpieczny. W szczególności, baza danych składa się z trzech głównych tabel relacyjnych:

- **stan_konta**: przechowuje informacje o stanie konta, w tym saldo każdej karty.
- **karty**: zawiera informacje o dostępnych kartach elektronicznych oraz przypisane do nich numery PIN.
- **tablehistory**: rejestruje historię transakcji, w tym typ operacji, kwotę oraz datę transakcji.

Każda interakcja z użytkownikiem jest weryfikowana przez system sprawdzając zgodność wprowadzonych numerów PIN z tymi przechowywanymi w tabeli **karty**. Po pomyślnej autoryzacji użytkownik ma dostęp do funkcji bankomatu takich jak sprawdzenie salda, wypłata i wpłata gotówki oraz przegląd historii transakcji.

Dodatkowo, aplikacja oferuje unikalne funkcje dla poprawy doświadczenia użytkownika:

- Możliwość regulacji poziomu głośności tła muzycznego, co jest rzadkością w tradycyjnych bankomatach, jednak znacząco podnosi komfort użytkowania aplikacji.
- Funkcjonalność eksportu i importu danych bazy danych, co umożliwia łatwe przenoszenie stanu aplikacji między różnymi środowiskami lub szybkie przywracanie systemu do stanu poprzedniego.

Każda z tych funkcji jest zaimplementowana z myślą o maksymalnej wygodzie i intuicyjności obsługi, co jest wspierane przez przejrzysty i responsywny interfejs użytkownika stworzony przy użyciu Java Swing. Projekt "Bankomat" jest nie tylko demonstracją umiejętności programistycznych, ale także próbą innowacji w zakresie interakcji człowieka z automatyzowanymi systemami bankowymi.

Cele i założenia projektu

1. **Cel projektu**: Zaprojektowano system symulacji pracy bankomatu, który ma na celu nie tylko edukację i trening pracowników banków, ale również testowanie i optymalizację rzeczywistego działania bankomatów.
2. **Problem i jego źródło**: Rozwiązany zostanie problem związany z ograniczonym dostępem do praktycznego szkolenia z zakresu obsługi bankomatów. Podstawowym źródłem problemu jest brak odpowiednich narzędzi symulacyjnych, które w bezpiecznym środowisku oddają realia pracy tych urządzeń.
3. **Ważność problemu i dowody**: Brak praktycznych narzędzi szkoleniowych prowadzi do zwiększenia ryzyka błędów operacyjnych i oszustw, co potwierdzają raporty dotyczące błędów obsługi bankomatów przez nowo zatrudniony personel.
4. **Co jest niezbędne do rozwiązania problemu**: Niezbędne jest stworzenie interaktywnej aplikacji symulującej pracę bankomatu, która umożliwi praktyczne szkolenie pracowników oraz testowanie nowych funkcji bez ryzyka finansowego.
5. **Sposób rozwiązania problemu**: Problem zostanie rozwiązany poprzez etapowy rozwój oprogramowania, rozpoczynając od projektu interfejsu użytkownika, a kończąc na integracji z rzeczywistymi systemami bankowymi. Wynikiem prac będzie aplikacja pozwalająca na symulację różnych scenariuszy użytkowania bankomatu.
6. **Założenia do projektu**: W projekcie założono, że aplikacja będzie elastyczna i skalowalna, umożliwiająca ciągłe dodawanie nowych funkcjonalności oraz łatwe adaptacje do zmieniających się wymagań bankowości cyfrowej.

2. Specyfikacja wymagań

2.1.Wymagania funkcjonalne

- **Autentykacja Użytkownika:**

  - Weryfikacja posiadacza karty na podstawie numeru PIN zapisanego w bazie danych.

  - Blokada karty po trzech nieudanych próbach wprowadzenia PIN.

- **Obsługa różnych typów kart:**

  - Akceptacja wielu typów kart bankowych, w tym Visa, Mastercard, American Express, Visa Electron, Diners Club i Japan Credit Bureau.

  - Dynamiczne rozpoznawanie typu karty na podstawie danych wprowadzonych przez użytkownika.

- **Operacje na koncie:**

  - Sprawdzenie salda konta z wykorzystaniem informacji z tabeli **stan_konta**.

  - Wypłata gotówki z aktualizacją salda w bazie danych.

  - Wpłata gotówki z aktualizacją salda w bazie danych.

  - Wyświetlanie historii transakcji, korzystając z danych z tabeli **tablehistory**.

- **Zarządzanie danymi:**

  - Eksport danych z bazy do pliku, co umożliwia łatwe przywracanie i przenoszenie danych.

  - Import danych do bazy z pliku, co pozwala na przywracanie stanu aplikacji.

- **Bezpieczeństwo:**

  - Szyfrowanie danych wrażliwych podczas transmisji między aplikacją a bazą danych.

  - Mechanizmy zapobiegające atakom podczas podglądania pin jest zakropkowany za pomocą passwordField.

- **Uwierzytelnianie Użytkownika:**

  - System musi umożliwić autoryzację użytkownika za pomocą karty bankowej i kodu PIN.

- **Transakcje Finansowe:**

  - Bankomat powinien obsługiwać podstawowe operacje bankowe, takie jak wypłata gotówki, sprawdzanie salda, wpłata gotówki oraz zapisywanie potwierdzenia transakcji w bazie.

- **Bezpieczeństwo i Audyt:**

  - Powinna być możliwość śledzenia wszystkich operacji na bankomacie oraz zapewnienie zgodności z wymogami bezpieczeństwa.

- **Interfejs użytkownika:**
  - Wyświetlanie komunikatów o błędach i potwierdzeniach operacji.
  - Nawigacja między ekranami aplikacji za pomocą przycisków i menu.

2.2.Wymagania niefunkcjonalne

- **Wydajność:**

  - Szybka i efektywna odpowiedź na zapytania użytkownika bez długich oczekiwań.

  - Optymalizacja zapytań do bazy danych dla zapewnienia płynności działania.

- **Użyteczność:**

  - Intuicyjny interfejs użytkownika, który jest łatwy w nawigacji nawet dla osób niezaznajomionych z technologią.

  - Czytelna i estetyczna prezentacja informacji.

- **Skalowalność:**

  - Łatwość w dodawaniu nowych funkcji i obsługi większej liczby użytkowników bez degradacji wydajności.

- **Dostępność:**

  - Aplikacja musi być dostępna bez błędów na wspieranych systemach operacyjnych.

  - Mechanizmy odzyskiwania po awarii w celu minimalizacji przestojów.

- **Modułowość:**

  - Struktura aplikacji podzielona na moduły, co ułatwia zarządzanie kodem i wprowadzanie zmian.

- **Nieprzerwana Dostępność:**

  - Bankomat powinien być dostępny 24/7/365, z minimalnym czasem przestoju.

- **Bezpieczeństwo:**

  - Musi zawierać zaawansowane mechanizmy szyfrowania i autoryzacji dwuskładnikowej, aby zapewnić bezpieczeństwo transakcji.

- **Testowalność:**

  - Możliwość przeprowadzenia pełnej gamy testów (jednostkowych, integracyjnych, systemowych, akceptacyjnych) w celu zapewnienia jakości i niezawodności.

3. Opis struktury projektu

Wykorzystywany język i narzędzia:

Java JDK: Zalecana wersja: Java Development Kit (JDK) 17 – Jest to najnowsza długoterminowa wersja wsparcia (LTS), która zapewnia stabilność i szeroki zakres funkcji, w tym poprawki bezpieczeństwa i wydajności.

Środowisko programistyczne: IntelliJ IDEA Najnowsza stabilna wersja: IntelliJ IDEA 2022.1 – Zawiera ulepszenia wydajności, lepsze wsparcie dla najnowszych wersji Javy, jak również zaawansowane narzędzia do analizy kodu i faktoryzacji.

GUI Toolkit: Swing - Użyta w projekcie wersja Swing jest zintegrowana z JDK, dlatego zaleca się użycie tej samej wersji JDK dla kompatybilności.

Baza danych: MySQL Zalecana wersja: MySQL 8.0 – Oferuje ulepszone mechanizmy bezpieczeństwa, wydajności i wsparcie dla nowych funkcji SQL.

MySQL Connector/J: Zalecana wersja: MySQL Connector/J 8.0.28 – Jest to zgodny sterownik JDBC dla wersji MySQL 8.0, który zapewnia wysoką wydajność połączeń z bazą danych.

Środowisko deweloperskie: IntelliJ IDEA

IntelliJ IDEA to zaawansowane środowisko programistyczne (IDE) firmy JetBrains, przeznaczone dla języka Java, które oferuje rozbudowane wsparcie dla programowania i refaktoryzacji kodu. Posiada wszechstronne narzędzia do zarządzania kodem, takie jak wyszukiwanie semantyczne, integracja z systemem kontroli wersji (Git), analiza statyczna kodu oraz wsparcie dla Mavena i Gradle'a do zarządzania zależnościami i procesem budowania projektu. IntelliJ IDEA zapewnia również wsparcie dla projektowania interfejsów użytkownika z wykorzystaniem Swing przez wbudowany edytor formularzy GUI.

Język programowania: Java

Java to język programowania wysokiego poziomu, który charakteryzuje się silnym typowaniem, obiektowością i przenośnością kodu między różnymi platformami. Jest to język wyboru dla wielu aplikacji biznesowych i korporacyjnych ze względu na jego stabilność, dojrzałość i szerokie wsparcie społeczności. W projekcie „Bankomat” Java jest używana do tworzenia logiki biznesowej aplikacji, obsługi zdarzeń interfejsu użytkownika oraz komunikacji z bazą danych.

Biblioteki GUI: Swing

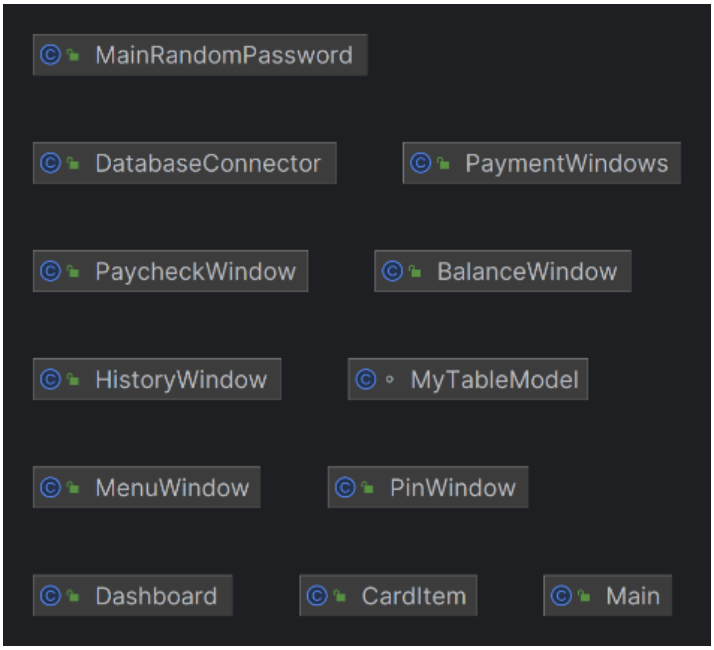
Swing to biblioteka do tworzenia graficznego interfejsu użytkownika (GUI) dla aplikacji Java. Oferuje bogaty zestaw komponentów, takich jak okna, przyciski, pola tekstowe, listy rozwijane i inne, które można łatwo dostosować i stylizować. Swing jest używany w projekcie do stworzenia responsywnego i intuicyjnego interfejsu, który umożliwia użytkownikom łatwą i bezproblemową interakcję z funkcjami bankomatu.

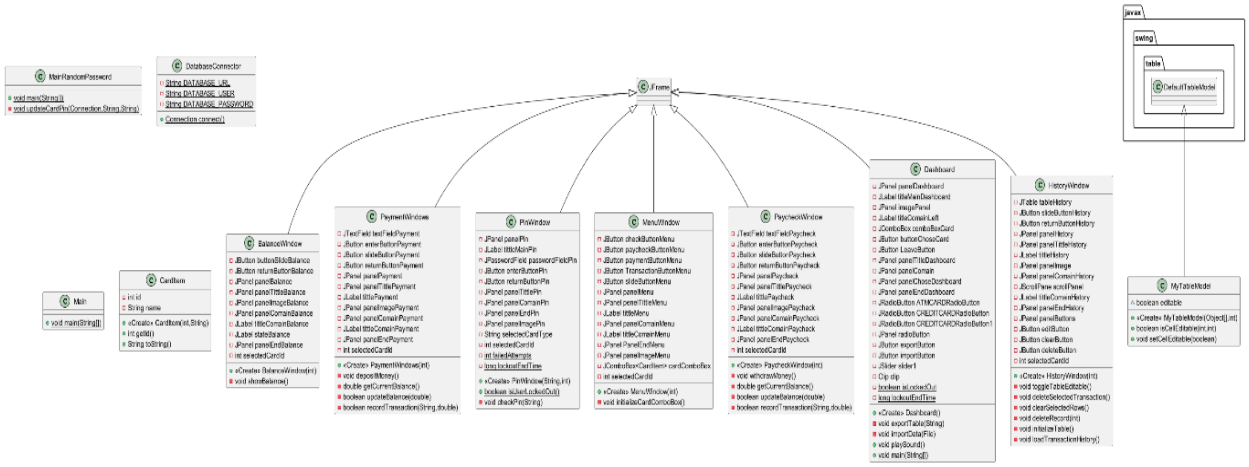
Sposób komunikacji z bazą danych: MySQL Connector

MySQL Connector/J to sterownik JDBC, który umożliwia aplikacjom Java komunikację z bazą danych MySQL. Jest to most między logiką aplikacji a bazą danych, pozwalający na wykonywanie zapytań SQL, aktualizowanie danych i odbieranie wyników. W projekcie jest wykorzystany do zarządzania transakcjami bankowymi, autentykacji użytkowników i przechowywania historii operacji. Użycie MySQL Connector/J zapewnia wydajne i bezpieczne zarządzanie danymi, co jest kluczowe dla operacji bankowych.

Każdy z tych elementów technicznych został wybrany w celu zapewnienia niezawodności, wydajności i bezpieczeństwa aplikacji "Bankomat". Język Java w połączeniu z IDE IntelliJ IDEA oferuje solidną platformę programistyczną, a Swing i MySQL Connector/J dostarczają niezbędnych narzędzi do stworzenia funkcjonalnego i bezpiecznego systemu bankowego.

Diagram klas:





Rysunki 1, 2. Diagramy klas projektowanej aplikacji programu „Bankomat”

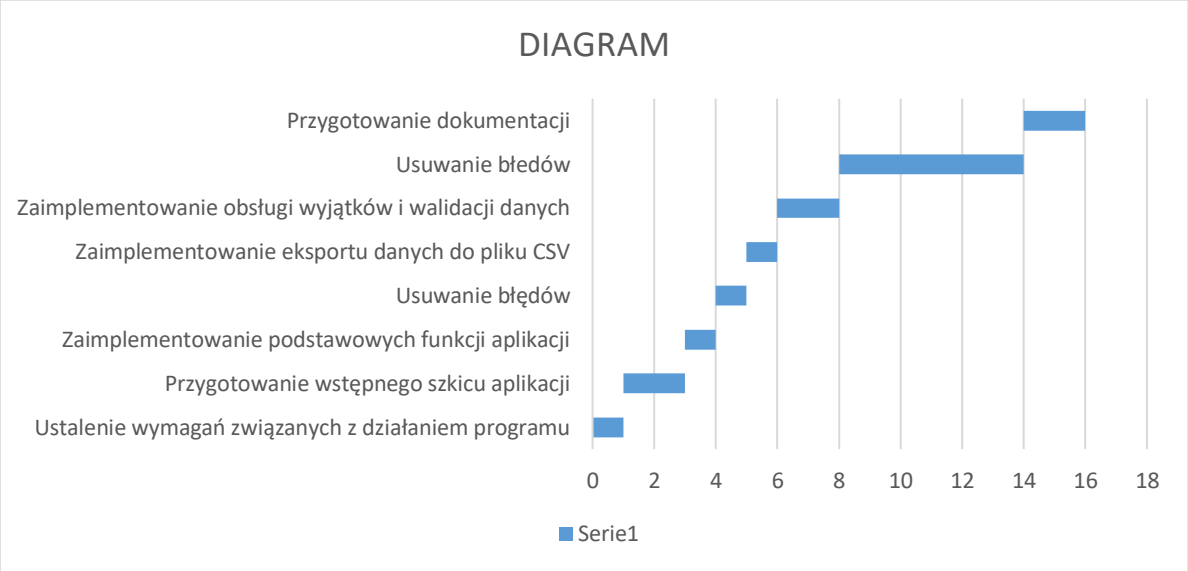
Struktury przedstawione na diagramach odnoszą się do architektury oprogramowania symulacji bankomatu. Zawierają one klasy i okna reprezentujące różne aspekty systemu:

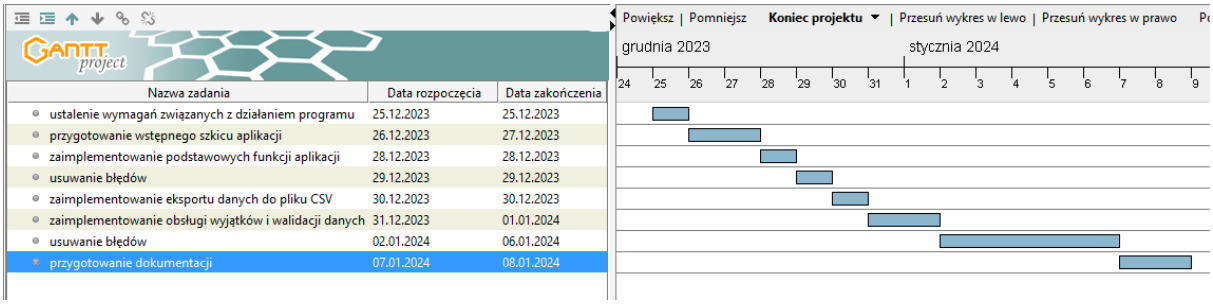
1. **Main**: Główna klasa inicjująca działanie aplikacji.
2. **DatabaseConnector**: Odpowiada za połączenie z bazą danych.
3. **CardItem**: Reprezentuje kartę bankową z jej podstawowymi atrybutami.
4. **MainRandomPassword**: Może być klasą generującą losowe hasła lub PINy.
5. **Dashboard**: Interfejs użytkownika prezentujący główne funkcje bankomatu.
6. **MenuWindow**: Okno menu oferujące różne opcje użytkownika.
7. **PinWindow**: Interfejs dla użytkownika do wprowadzania PINu.
8. **BalanceWindow**: Okno wyświetlające saldo dostępne na koncie użytkownika.
9. **PaymentWindows**: Okno umożliwiające realizację płatności.
10. **PaycheckWindow**: Okno do zarządzania czekami i wypłatami.
11. **HistoryWindow**: Okno wyświetlające historię transakcji.
12. **MyTableModel**: Może być klasą dostosowującą dane do wyświetlenia w tabelach.

Każda klasa i okno zawierają metody i atrybuty odpowiadające za specyficzne zachowanie lub prezentację danych w aplikacji. Na przykład, w klasie „PaymentWindows” znajdują się metody do zarządzania płatnościami, jak „depositMoney()” czy „updateBalance()”, natomiast „HistoryWindow” zarządza wyświetlaniem historii transakcji z użyciem metod takich jak „loadTransactionHistory()”. Diagram wskazuje na modularną i obiektową budowę systemu, gdzie każdy komponent jest odpowiedzialny za odizolowany zestaw funkcjonalności.

4. Harmonogram realizacji projektu (diagram Gantta)

Poniżej zamieszono harmonogram realizacji projektu.



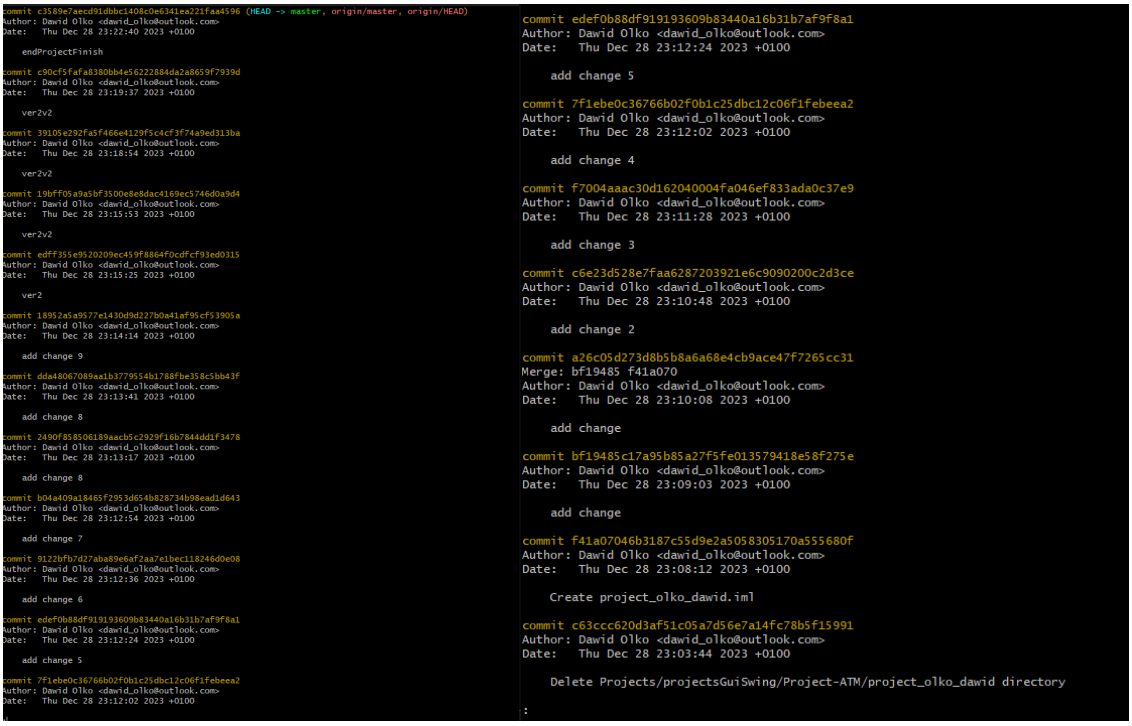


Rysunek 3, 4. Diagram Gantta Excel / GanttProject

Obydwa diagramy Gantta przedstawiają harmonogram realizacji projektu, zaznaczając kluczowe etapy prac oraz ich daty rozpoczęcia i zakończenia. Pierwszy diagram wskazuje na procesy takie jak przygotowanie dokumentacji, usuwanie błędów, implementację eksportu danych do pliku CSV, a także przygotowanie wstępnego szkicu aplikacji. Drugi diagram pokazuje plan prac na poszczególne dni miesiąca i wyraźnie dzieli działania na kolejne etapy.

Trudności i problemy pojawiły się w przypadku obsługi wyjątków i walidacji danych, gdzie widać zwiększoną alokację czasu na te działania. Wymagało to szczególnego skupienia na szczegółach i testowaniu, aby zapewnić poprawność działania aplikacji. Ponadto, istotną część projektu stanowiło przygotowanie dokumentacji, co sugeruje, że duży nacisk kładziono na dokładne opisanie specyfikacji i funkcjonalności systemu.

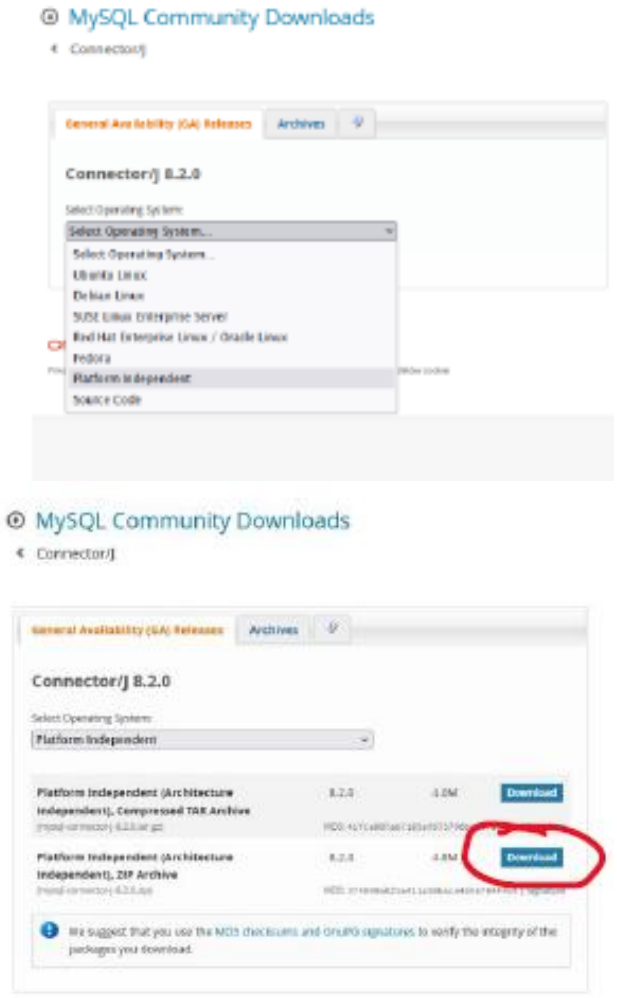
Projekt realizowany był z wykorzystaniem systemu kontroli wersji Git, wszystkie pliki źródłowe projektu znajdują się pod adres: https://github.com/dawidolko/java-ATM-system.git i będą dostępne do 31.01.2025.



Rysunek 5. Historia comitów

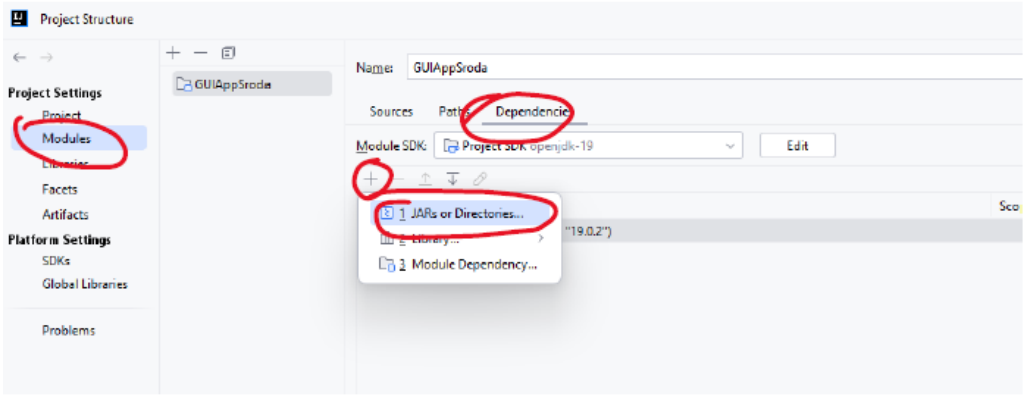
5. Prezentacja warstwy użytkowej projektu

Program „Bankomat” działa po połączeniu się wszystkich klas z tabelami z bazy, którą łączymy przy pomocy SQL i JDBC za pomocą SQL-Connecor-j-8.2.0.jar, aby to uczynić musimy pobrać plik JAR ze strony https://dev.mysql.com/downloads/connector/j/. Wybrać platformę zgodnie z poniższym rysunkiem.



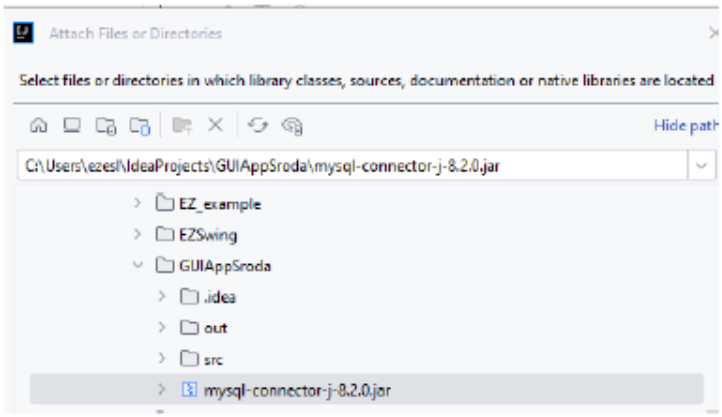
Rysunek 6, 7. Zdjęcia przedstawiające pobieranie pliku JAR

Gdy już został pobrany, trzeba go wgrać do projektu w IntelliJ IDEA, gdy to zostało wykonane przechodzimy do ustawień projektu FILE -> Project Structure, następnie należy zrobić to co jest na poniższym zdjęciu.



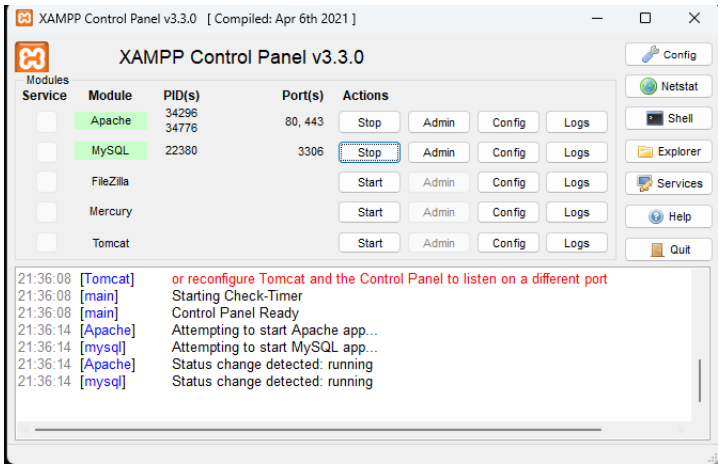
Rysunek 8. Zdjęcie przedstawiające wgrywanie pliku JAR.

Następnie pobrany plik przerzucamy do folderu projektu jak na poniższym zdjęciu.



Rysunek 9. Zdjęcie przedstawiające wgrywanie pliku JAR.

Gdy już to zostało wykonane to tworzymy bazę za pomocą xampp, którego można pobrać z Internetu pod tym linkiem https://www.apachefriends.org/pl/index.html. Jeśli już go mamy i został zainstalowany plik instalacyjny, po włączeniu xampp uruchamiamy akcję modułów Apache i MySQL jak na poniższym zdjęciu.



Rysunek 10. Zdjęcie przedstawiające xampp.

Gdy już został połączony i otwarty port Apache, jak i MySQL należy przejść do połączenia się na przykład w przeglądarce z adresem: http://localhost/phpmyadmin, który automatycznie przeniesie nas do naszego serwera lokalnego. Tam poprzez kliknięcie zakładki „New" tworzymy nową bazę, którą nazwiemy „atm" jest to ważne, ponieważ jeśli nazwa będzie inna nasz program nie połączy się z bazą i wyrzuci odpowiedni błąd o złym powiązaniu bazy. Gdy już została utworzona baza „atm" należy przejść do zakładki „SQL" i dodać 3 zapytania tworzące tabele z wartościami, które możemy według preferencji zmienić, trzeba pamięć jednak, aby kolumny nie miały innych nazw niż te które podane są niżej z racji tego iż program „Bankomat" będzie wyrzucał błędy.

```sql
--
-- Struktura tabeli dla tabeli `stan_konta`
--

CREATE TABLE `stan_konta` (
  `id` int(11) NOT NULL,
  `karta_id` int(11) DEFAULT NULL,
  `saldo` decimal(10,2) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

--
-- Dumping data for table `stan_konta`
--

INSERT INTO `stan_konta` (`id`, `karta_id`, `saldo`) VALUES
(1, 1, 6467.30),
(2, 2, 987.85),
(3, 3, 5657.33),
(4, 4, 5323.09),
(5, 5, 9643.47),
(6, 6, 2248.10);

--
-- Struktura tabeli dla tabeli `karty`
--

CREATE TABLE `karty` (
  `id` int(11) NOT NULL,
  `typ` varchar(200) NOT NULL,
  `PIN` varchar(4) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_c

--
-- Dumping data for table `karty`
--

INSERT INTO `karty` (`id`, `typ`, `PIN`) VALUES
(1, 'Visa', '1234'),
(2, 'American Express', '2345'),
(3, 'Visa Electron', '3456'),
(4, 'Mastercard', '4567'),
(5, 'Diners Club', '5678'),
(6, 'Japan Credit Bureau', '6789');

--
-- Indeksy dla zrzutów tabel
--

--
-- Struktura tabeli dla tabeli `tablehistory`
--

CREATE TABLE `tablehistory` (
  `transaction_id` int(11) NOT NULL,
  `karta_id` int(11) NOT NULL,
  `transaction_type` varchar(10) NOT NULL,
  `amount` decimal(10,2) NOT NULL,
  `transaction_date` timestamp NOT NULL DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

--
-- Dumping data for table `tablehistory`
--

INSERT INTO `tablehistory` (`transaction_id`, `karta_id`, `transaction_type`, `amount`, `transaction_date`) VALUES
(1, 1, 'WITHDRAW', 10.00, '2023-12-27 14:45:56'),
(2, 1, 'DEPOSIT', 10.00, '2023-12-27 14:46:00'),
(3, 1, 'WITHDRAW', 100.00, '2023-12-27 14:46:45'),
(4, 1, 'DEPOSIT', 300.00, '2023-12-27 14:46:55'),
(5, 1, 'WITHDRAW', 10.00, '2023-12-27 15:09:41'),
(6, 1, 'DEPOSIT', 10.00, '2023-12-27 15:09:50'),
(7, 1, 'WITHDRAW', 10.00, '2023-12-27 15:12:17'),
(8, 1, 'WITHDRAW', 10.00, '2023-12-27 15:13:50'),
(9, 1, 'WITHDRAW', 10.00, '2023-12-27 15:15:04'),
(10, 1, 'WITHDRAW', 10.00, '2023-12-27 15:15:50'),
(11, 1, 'WITHDRAW', 10.00, '2023-12-27 15:22:33'),
(12, 1, 'WITHDRAW', 123.00, '2023-12-27 15:31:46'),
(13, 1, 'DEPOSIT', 123.00, '2023-12-27 15:31:51'),
(14, 1, 'WITHDRAW', 10.00, '2023-12-27 15:48:24'),
(15, 1, 'WITHDRAW', 10.00, '2023-12-27 15:49:09'),
(16, 1, 'WITHDRAW', 10.00, '2023-12-27 15:57:21'),
(17, 1, 'DEPOSIT', 10.00, '2023-12-27 15:57:36'),
(18, 1, 'DEPOSIT', 10.00, '2023-12-27 16:08:25'),
(19, 1, 'WITHDRAW', 10.00, '2023-12-27 16:25:05'),
(20, 1, 'WITHDRAW', 10.00, '2023-12-27 16:25:56'),
(21, 1, 'WITHDRAW', 10.00, '2023-12-27 22:36:16'),
(22, 1, 'DEPOSIT', 10.00, '2023-12-27 22:36:20');

--
-- Indeksy dla zrzutów tabel
--
```

Rysunek 11, 12, 13. Zdjęcie przedstawiające tabele i wartości jakie dodajemy do naszej bazy „atm”

Gdy już zostały dodane te zawartości do bazy, można zacząć pracę z programem „Bankomat”. Zaczynając od klasy która połączy nam wszystkie klasy za pomocą naszej wyżej stworzonej bazy, klasa która za to odpowiada to „.DatabaseConnector”. Klasa „DatabaseConnector” służy jako centralny punkt do nawiązywania połączenia z bazą danych. Jest to przykładowy wzorzec projektowy używany w aplikacjach Java do izolowania szczegółów połączenia z bazą danych od reszty aplikacji.

Oto dokładny opis jej działania:

Klasa definiuje trzy stałe - DATABASE_URL, DATABASE_USER, i DATABASE_PASSWORD. Te stałe przechowują informacje niezbędne do połączenia z bazą danych, w tym URL (adres bazy danych), nazwę użytkownika oraz hasło. W przypadku tej klasy, są one ustawione na konkretne wartości, które wskazują na lokalną bazę danych MySQL (jdbc:mysql://localhost:3306/atm).

Metoda connect(): connect(): Jest to główna metoda w klasie „DatabaseConnector”, służąca do nawiązywania połączenia z bazą danych.

Oto co się dzieje podczas jej wywołania:

Ładowanie Sterownika: Java wymaga załadowania odpowiedniego sterownika bazy danych przed nawiązaniem połączenia. W przypadku nowszych wersji JDBC, sterownik ładuje się automatycznie, więc ten krok często można pominąć.

Nawiązywanie Połączenia:

Metoda DriverManager.getConnection() jest używana z wcześniej zdefiniowanymi stałymi DATABASE_URL, DATABASE_USER, i DATABASE_PASSWORD do stworzenia i zwrócenia obiektu Connection. Ten obiekt Connection jest później wykorzystywany przez inne części aplikacji do wysyłania zapytań SQL i zarządzania transakcjami z bazą danych. Obsługa Wyjątków: W przypadku problemów z połączeniem (np. błędne dane logowania, baza danych nie działa, itp.), metoda rzuca wyjątek RuntimeException z odpowiednią wiadomością o błędzie. To pozwala wywołującemu kodowi na obsłużenie problemu połączenia w odpowiedni sposób.

Wykorzystanie:

Klasa „DatabaseConnector” jest typowo używana w całej aplikacji do uzyskiwania połączenia z bazą danych. Na przykład, inne klasy, które potrzebują wykonać zapytanie SQL, mogą najpierw wywołać DatabaseConnector.connect() do uzyskania obiektu Connection, a następnie użyć tego obiektu do wykonania zapytania.

Podsumowanie:

„DatabaseConnector” jest istotnym elementem aplikacji, umożliwiającym abstrakcję i centralizację zarządzania połączeniem z bazą danych. Dzięki izolowaniu detali połączenia i sposobu uzyskiwania połączenia, klasa ta zapewnia elastyczność (łatwość zmiany baz danych, parametrów połączenia itp.) oraz czystość kodu, zmniejszając powtarzalność kodu i skupiając obsługę bazy danych w jednym miejscu.

Teraz już jest wiadome jak połączono bazę z programem, a więc można przejść do przedstawienia jego działania. Przejdźmy więc do elementu głównego, mianowicie funkcji „main” oraz „mainRandomPassword”.

Main:

Klasa „Main„ pełni funkcję głównego punktu wejścia do aplikacji. Wykorzystuje ona połączenie z bazą danych za pomocą klasy DatabaseConnector, aby pobrać informacje o kartach (id, typ, PIN) z tabeli karty. Dzieje się to w bloku try-with-resources, co zapewnia automatyczne zamknięcie zasobów po zakończeniu operacji. Po pobraniu danych, aplikacja wyświetla ścieżkę katalogu roboczego i uruchamia interfejs użytkownika Dashboard, który jest prawdopodobnie panelem zarządzania czy kontrolnym.

Klasa ta skupia się głównie na dwóch aspektach:

Połączenie z bazą danych i pobranie danych: Wykorzystuje zapytania SQL do pobrania danych z tabeli karty w bazie danych i przetwarza te dane w kontekście aplikacji.

Uruchomienie interfejsu użytkownika: Po pobraniu danych i obsłudze bazy danych, program inicjuje i wyświetla interfejs użytkownika (Dashboard), który służy jako główne okno aplikacji.

MainRandomPassword:

„MainRandomPassword” jest alternatywną wersją klasy Main, która generuje losowe numery PIN dla różnych typów kart i aktualizuje je w bazie danych. To może być użyteczne do resetowania lub inicjalizacji danych PIN w systemie. Oto kluczowe aspekty działania tej klasy: Połączenie z bazą danych: Podobnie jak Main, łączy się z bazą danych za pomocą DatabaseConnector. Generowanie losowych PIN-ów: Używa klasy Random do wygenerowania losowych numerów PIN dla zdefiniowanych typów kart.

Aktualizacja bazy danych: Dla każdego typu karty, klasa aktualizuje numer PIN w bazie danych, używając wygenerowanych losowo numerów. Funkcja „updateCardPin” zajmuje się aktualizacją PIN-u w bazie danych.

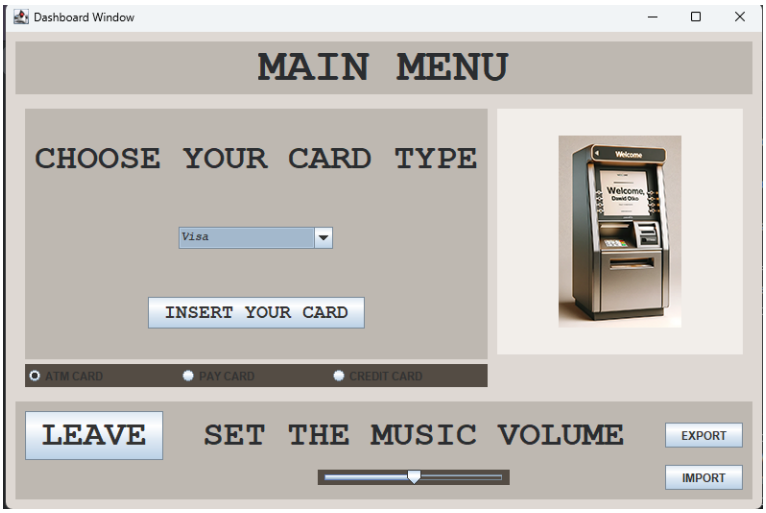
Różnice między Main a MainRandomPassword:

Podczas gdy Main służy do pobierania danych z bazy danych i uruchamiania UI, MainRandomPassword ma specyficzne zastosowanie w generowaniu i aktualizowaniu losowych numerów PIN dla kart. Można przypuszczać, że MainRandomPassword jest narzędziem administracyjnym lub częścią procesu inicjalizacji systemu, gdzie konieczne jest zapewnienie, że wszystkie karty mają ustawione PIN-y, być może przed uruchomieniem systemu lub jako część regularnej procedury bezpieczeństwa.

Dlaczego istnieje MainRandomPassword:

„MainRandomPassword” istnieje jako oddzielna funkcjonalność do resetowania lub inicjalizacji PIN-ów kart w systemie. Może być używana do masowego ustawiania lub resetowania PIN-ów, co jest przydatne w różnych scenariuszach administracyjnych, takich jak pierwsze uruchomienie systemu, okresowe resetowanie dla bezpieczeństwa, czy też w przypadku, gdy organizacja decyduje się na zmianę schematu PIN-ów dla swoich kart. Jest to przykład, jak aplikacja może zawierać narzędzia pomocnicze lub administracyjne oprócz swojej głównej funkcjonalności.

Interfejs użytkownika Dashboard prezentowany na przesłanych zdjęciach to aplikacja z graficznym interfejsem użytkownika (GUI), która służy do zarządzania kartami płatniczymi i transakcjami w bankomacie lub systemie bankowym. Omówię funkcje każdego elementu w dashboardzie.



Rysunek 14. Zdjęcie przedstawiające dashboard.

Nagłówek 'MAIN MENU': Tytuł ekranu, informujący użytkownika, że znajduje się w głównym menu aplikacji.

Pole wyboru 'CHOOSE YOUR CARD TYPE': ComboBox umożliwiający użytkownikowi wybór typu karty (np. Visa, Mastercard itd.), z którą chce interagować.

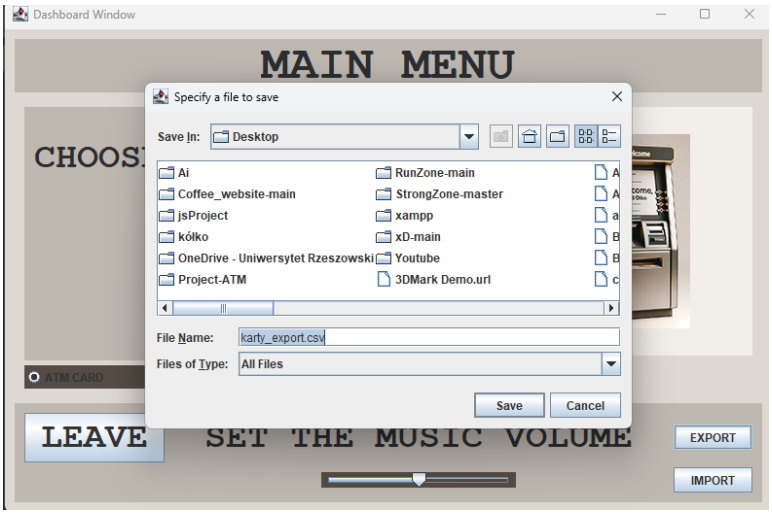
Przycisk 'INSERT YOUR CARD': Przycisk, który użytkownik może użyć do "włożenia" wybranej karty, co prawdopodobnie inicjuje proces weryfikacji lub transakcji.

Radio Buttons dla typu karty: Grupa przycisków radiowych pozwalających użytkownikowi wybrać rodzaj karty (ATM CARD, PAY CARD, CREDIT CARD), które mogą zmieniać opcje lub procesy dostępne w zależności od wyboru.

Przycisk 'LEAVE': Pozwala użytkownikowi na wyjście z aplikacji.

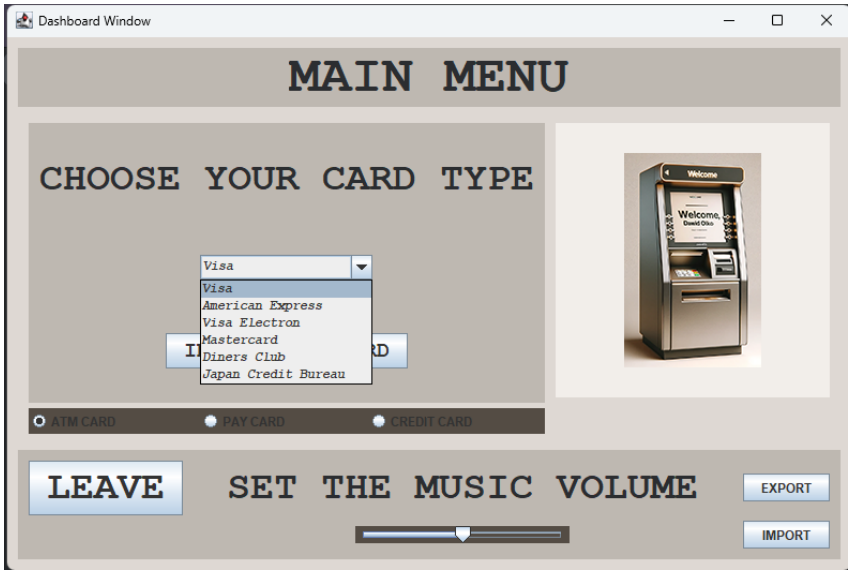
Regulacja głośności 'SET THE MUSIC VOLUME': Suwak do regulacji głośności dźwięków w aplikacji, co może dotyczyć sygnałów dźwiękowych wydawanych przez aplikację.

Przyciski 'EXPORT' i 'IMPORT': Umożliwiają eksport danych do pliku CSV oraz import danych z pliku CSV.



Rysunek 15. Zdjęcie przedstawiające export/import.

Ekrany wyboru pliku: Pojawiają się po kliknięciu przycisków 'EXPORT' lub 'IMPORT'. Pozwalają użytkownikowi na wybór lokalizacji, do której eksportować dane, lub z której importować dane.



Rysunek 16. Zdjęcie przedstawiające listę rozwijalną.

Rozszerzona lista rozwijana: Wyświetla wszystkie dostępne typy kart, które użytkownik może wybrać. W tym przypadku pokazuje różne rodzaje kart płatniczych, takie jak Visa, American Express, Visa Electron, Mastercard, Diners Club, Japan Credi Bureau.

Funkcjonalność: Kiedy użytkownik wybierze typ karty i kliknie 'INSERT YOUR CARD', aplikacja może przechodzić do kolejnego etapu, który może wymagać wprowadzenia PIN-u lub wykonania innej czynności.

Przyciski 'EXPORT' i 'IMPORT' służą do zarządzania danymi aplikacji. 'EXPORT' pozwala na zapisanie danych do pliku CSV, który może być używany do tworzenia kopii zapasowych lub przenoszenia danych między systemami. 'IMPORT' umożliwia wczytanie danych z pliku CSV, co jest przydatne przy przywracaniu danych lub aktualizacji systemu nowymi informacjami.

Dźwięki w aplikacji: Możliwość regulacji głośności wskazuje, że aplikacja zawiera elementy multimedialne, takie jak dźwięki, które mogą być odtwarzane podczas różnych akcji, na przykład gdy karta jest akceptowana lub odrzucana, lub podczas wypłaty gotówki.

Ogólnie: Dashboard jest zaprojektowany w taki sposób, aby umożliwić łatwą interakcję z systemem i zarządzanie kartami płatniczymi i transakcjami. Wszystkie te elementy są zaprojektowane z myślą o intuicyjności, tak aby użytkownik mógł szybko i skutecznie korzystać z funkcji oferowanych przez system.

Okno **„PinWindow”**, widoczne na poniższych zdjęciach, jest interfejsem użytkownika do autoryzacji użytkownika przez wprowadzenie kodu PIN karty bankomatowej.

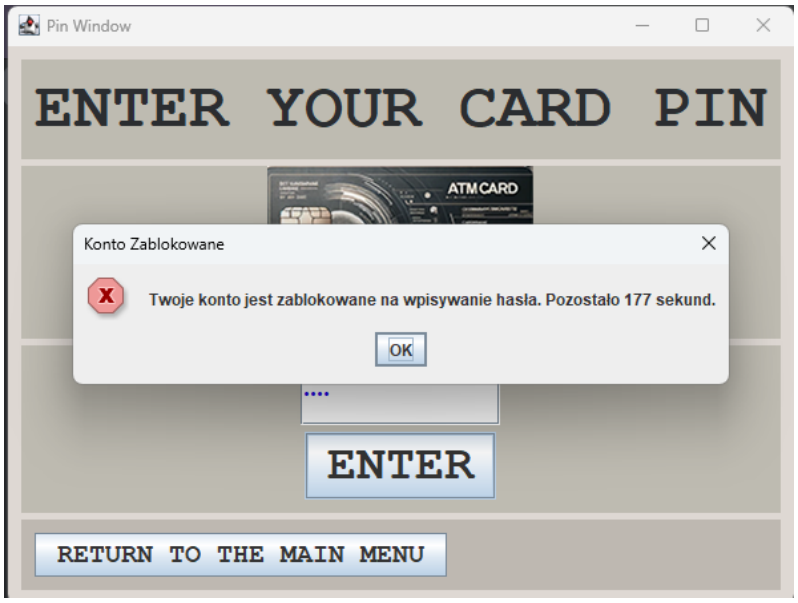
Główne Elementy **PinWindow**:

Nagłówek "ENTER YOUR CARD PIN": Informuje użytkownika, że w tym miejscu powinien wprowadzić swój PIN.

Pole tekstowe do wprowadzania PIN-u: Miejsce, gdzie użytkownik wpisuje kod PIN. Zazwyczaj ukrywa wpisywane cyfry za pomocą gwiazdek lub kropek dla bezpieczeństwa.

Przycisk "ENTER": Po wprowadzeniu PIN-u, użytkownik naciska ten przycisk, aby zatwierdzić i przesłać informacje do systemu.

Przycisk "RETURN TO THE MAIN MENU": Umożliwia powrót do głównego menu bez wprowadzania PIN-u.

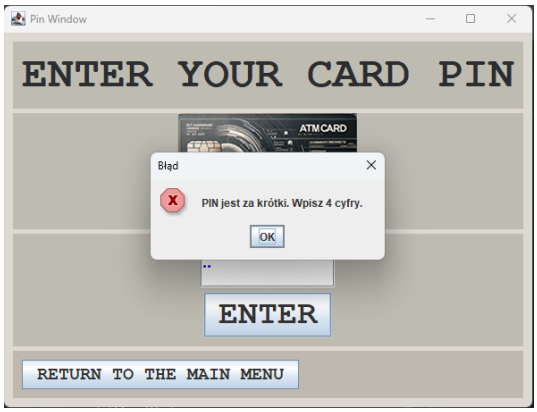


Rysunek 17. Zdjęcie przedstawiające walidację.

Funkcjonalność i Walidacja: Walidacja PIN-u: Wprowadzony PIN jest walidowany przy użyciu zdefiniowanych reguł. W przypadku, gdy PIN jest za krótki (mniej niż 4 cyfry), użytkownik otrzyma komunikat o błędzie, informujący o niewłaściwej długości PIN-u.

Blokada Konta: Jeśli użytkownik wprowadzi błędny PIN kilka razy z rzędu, konto zostaje zablokowane na określony czas, co jest sygnalizowane przez odpowiedni komunikat z licznikiem sekund pozostałych do odblokowania.

Obsługa Zdarzeń: Obsługa Przycisku "ENTER": Po kliknięciu tego przycisku, kod przetwarza wprowadzony PIN i porównuje go z PIN-em przechowywanym w bazie danych. Jeśli PIN jest poprawny, użytkownik może przejść dalej, jeśli nie - może zostać wyświetlony komunikat o błędzie lub blokada.



Rysunek 18. Zdjęcie przedstawiające alert o za krótkim haśle.

Powrót do Głównego Menu: Przycisk "RETURN TO THE MAIN MENU" pozwala użytkownikowi anulować proces i wrócić do poprzedniego interfejsu.

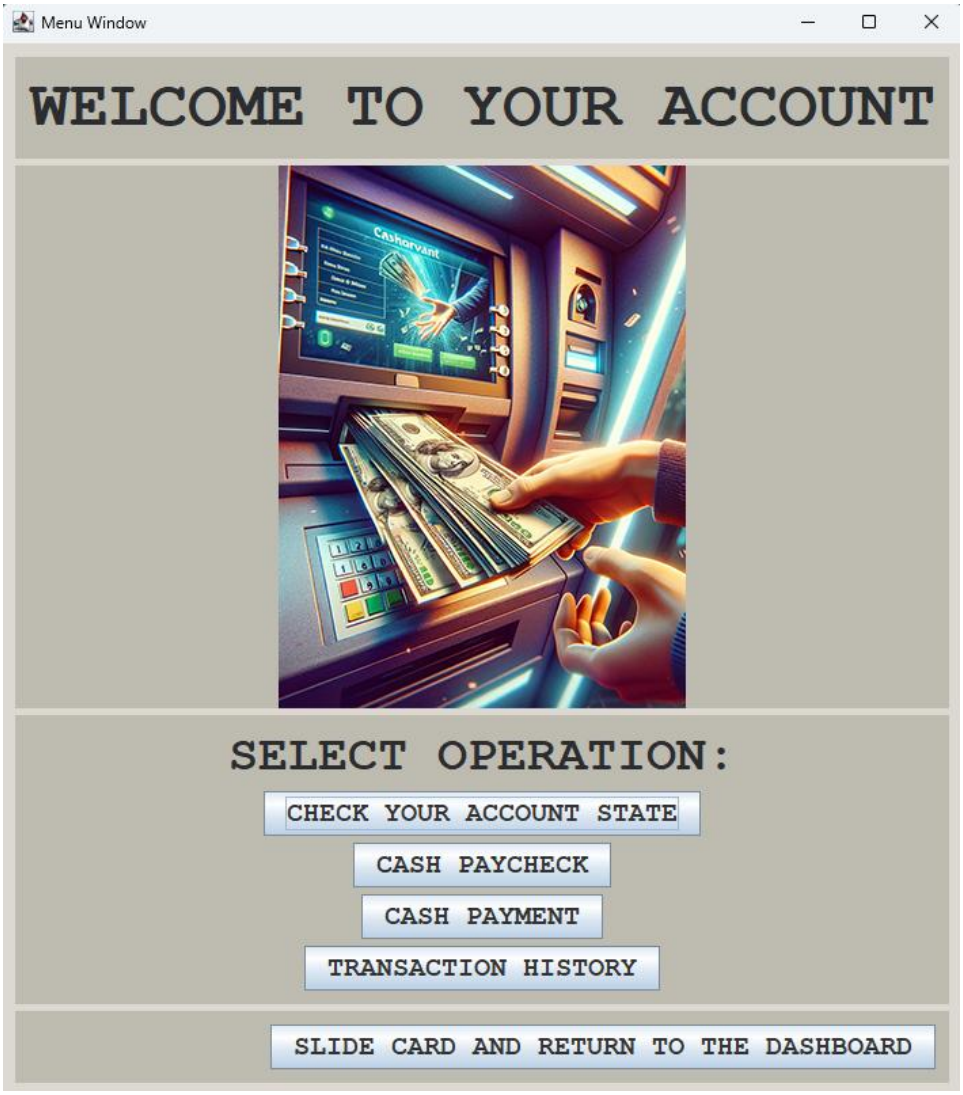
Komunikaty o Błędach: Na jednym ze zdjęć widać komunikat o zablokowanym koncie, co oznacza, że system zaimplementował mechanizm bezpieczeństwa, który blokuje dostęp po kilku nieudanych próbach wprowadzenia PIN-u. Na innym zdjęciu pojawia się komunikat informujący, że PIN jest za krótki, co sugeruje, że system oczekuje czterocyfrowego kodu.

Backend (Kod): W kodzie, klasa PinWindow jest odpowiedzialna za tworzenie i zarządzanie tym oknem. Używa ona metody checkPin do weryfikacji wprowadzonego PIN-u. System zlicza nieudane próby wprowadzenia PIN-u i może zablokować dostęp do konta na określony czas (lockoutEndTime), co zostało pokazane w jednym z komunikatów. W przypadku poprawnego wprowadzenia PIN-u, aplikacja prawdopodobnie przechodzi do kolejnego okna lub wykonuje kolejną akcję, która nie jest widoczna w przesłanym kodzie.

Rysunek 19. Zdjęcie przedstawiające PinWindow.

Ogólna Analiza: PinWindow jest kluczowym elementem aplikacji bankowej, który zapewnia bezpieczeństwo dostępu do konta użytkownika. Weryfikacja kodu PIN, obsługa błędów i blokada konta po nieudanych próbach to standardowe praktyki w aplikacjach wymagających autoryzacji. Interfejs jest prosty i bezpośredni, co minimalizuje ryzyko błędów ze strony użytkownika i zwiększa bezpieczeństwo.

Okno „**MenuWindow**” przedstawione na zdjęciu jest kolejnym etapem interakcji użytkownika z systemem bankowym po zalogowaniu się za pomocą kodu PIN. To okno jest centrum, z którego użytkownik może wybrać różne operacje bankowe.



Rysunek 20. Zdjęcie przedstawiające MenuWindow.

Główne Elementy **MenuWindow**:

Nagłówek "WELCOME TO YOUR ACCOUNT": Przywitanie użytkownika i potwierdzenie, że został pomyślnie zautoryzowany.

Sekcja "SELECT OPERATION": Lista opcji, z których użytkownik może wybrać:

- CHECK YOUR ACCOUNT STATE: Pozwala użytkownikowi sprawdzić bieżący stan swojego konta, takie jak saldo.

- CASH PAYCHECK: Funkcja umożliwiająca użytkownikowi wypłatę środków z konta.

- CASH PAYMENT: Służy do wpłacania pieniędzy na konto lub dokonywania płatności.

- TRANSACTION HISTORY: Przegląd historii transakcji przeprowadzonych za pomocą konta.

Przycisk "SLIDE CARD AND RETURN TO THE DASHBOARD": Umożliwia użytkownikowi szybki powrót do głównego menu aplikacji.

Kod i Jego Funkcjonalność:

W kodzie, klasa MenuWindow jest odpowiedzialna za wyświetlenie i zarządzanie tym oknem. Zawiera ona logikę inicjalizującą i obsługującą interakcje użytkownika:

Przyciski Funkcji: Każdy przycisk odpowiada za inicjowanie określonej operacji. Kliknięcie któregokolwiek z nich może prowadzić do otwarcia nowego okna z odpowiednią funkcją, np. sprawdzenie stanu konta, wypłata, wpłata czy historia transakcji.

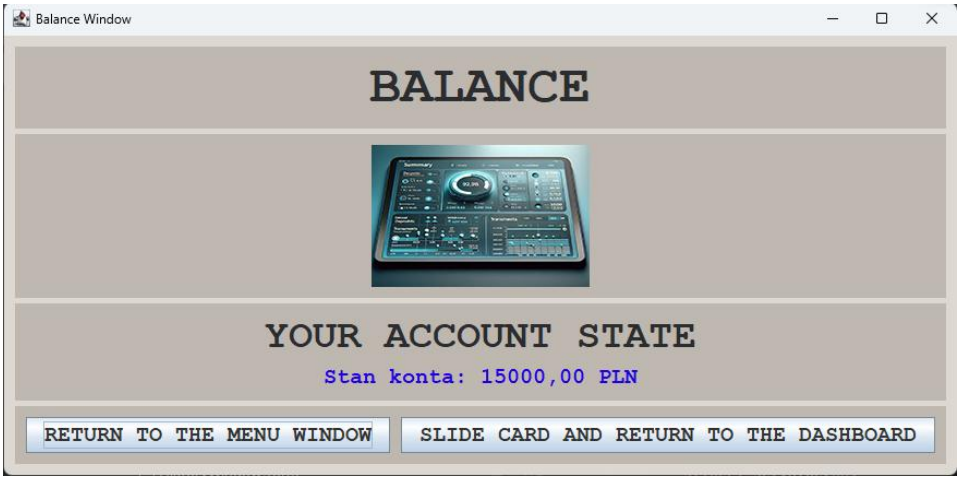
Powrót do Dashboardu: Przycisk "RETURN TO THE MAIN MENU" kończy sesję w MenuWindow i przywraca Dashboard, co może być realizowane przez zamknięcie obecnego okna i otwarcie nowego okna klasy Dashboard.

Dialogi i Akcje: W aplikacji są zaimplementowane dialogi potwierdzające akcje użytkownika lub wyświetlające komunikaty o błędach, które jednak nie są pokazane na dostarczonym zrzucie ekranu. Akcje przycisków są obsługiwane przez ActionListener, które reagują na kliknięcia i wykonują określone metody, takie jak dispose() do zamknięcia bieżącego okna oraz setVisible(true) do wyświetlenia nowego okna.

Ogólna Analiza:

MenuWindow jest zaprojektowane, aby być przyjaznym dla użytkownika centrum zarządzania kontem bankowym, które umożliwia łatwy dostęp do podstawowych funkcji bankowych. Kolejne kroki po wybraniu opcji są prawdopodobnie zaimplementowane w innych klasach i oknach, które były częścią dostarczonego kodu, takich jak BalanceWindow, PaycheckWindow, PaymentWindows i HistoryWindow, odpowiadających za szczegółową obsługę każdej z operacji.

Okno „**BalanceWindow**”, widoczne na zdjęciu, służy do wyświetlania stanu konta użytkownika w systemie bankowym. Jest to interfejs, w którym użytkownik może zobaczyć aktualne saldo swojego konta bankowego po zalogowaniu się i przejściu przez odpowiednie procedury autentykacje.



Rysunek 21. Zdjęcie przedstawiające BalanceWindow.

Główne Elementy **BalanceWindow**:

Nagłówek "BALANCE": Wyraźnie informuje użytkownika, że znajduje się w sekcji dotyczącej salda konta.

Informacja o stanie konta "YOUR ACCOUNT STATE": Wyświetla aktualne saldo konta użytkownika. Na przesłanym zdjęciu saldo to "15000,00 PLN", lecz dla każdej karty jest to wartość pobrana z bazy.

Przycisk "RETURN TO THE MENU WINDOW": Pozwala użytkownikowi na powrót do poprzedniego menu (MenuWindow) w celu wybrania innych opcji.

Przycisk "SLIDE CARD AND RETURN TO THE DASHBOARD": Umożliwia wyjście z obecnego widoku i powrót do głównego menu aplikacji (Dashboard).

Kod i Jego Funkcjonalność:

W kodzie, klasa BalanceWindow jest odpowiedzialna za wyświetlenie i zarządzanie tym oknem. Używa ona połączenia z bazą danych do pobrania i wyświetlenia aktualnego salda konta użytkownika:

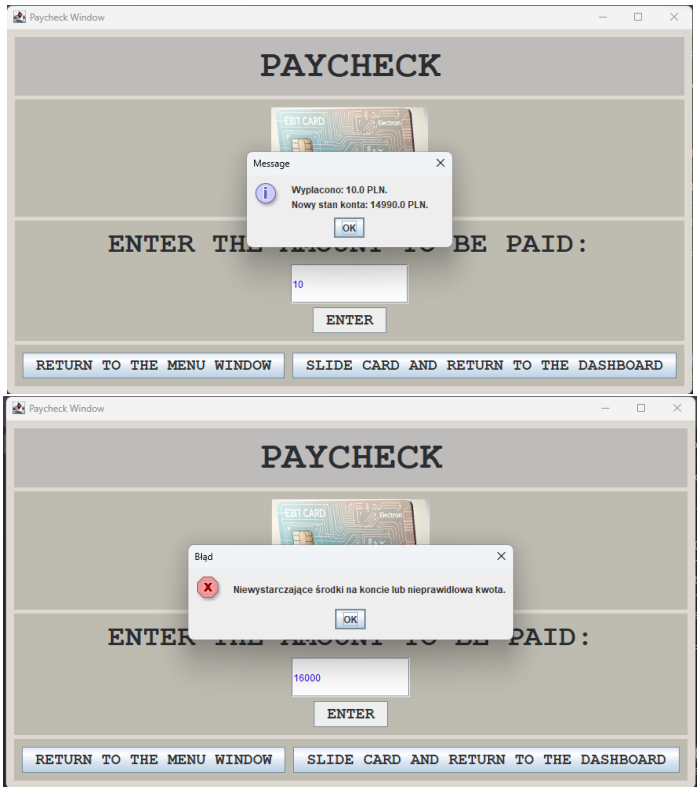
- Pobieranie salda: W metodzie showBalance(), BalanceWindow łączy się z bazą danych za pomocą DatabaseConnector.connect(). Następnie wykonywane jest zapytanie SQL, które pobiera saldo konta związane z określonym identyfikatorem karty użytkownika.

- Obsługa zdarzeń przycisków: Przyciski w oknie obsługują akcje, takie jak zamykanie okna BalanceWindow i otwieranie okna MenuWindow lub Dashboard poprzez metody dispose() i setVisible(true). Dzięki temu użytkownik może łatwo nawigować między różnymi częściami aplikacji.

Ogólna Analiza:

BalanceWindow to proste i intuicyjne narzędzie, które umożliwia użytkownikom szybki i łatwy dostęp do informacji o stanie swoich środków finansowych. Jest to kluczowa funkcja w aplikacjach bankowych, gdyż pozwala na bieżące monitorowanie zasobów finansowych, co jest istotne z punktu widzenia zarządzania osobistymi finansami. Kod za tym oknem zapewnia potrzebne funkcje do interakcji z bazą danych i obsługi zdarzeń interfejsu użytkownika, umożliwiając płynne doświadczenie użytkownika.

P**aycheckWindow** i **PaymentWindow** to okna dialogowe w aplikacji bankowej, które pozwalają użytkownikowi na realizację transakcji finansowych - odpowiednio wypłatę i wpłatę środków na konto.

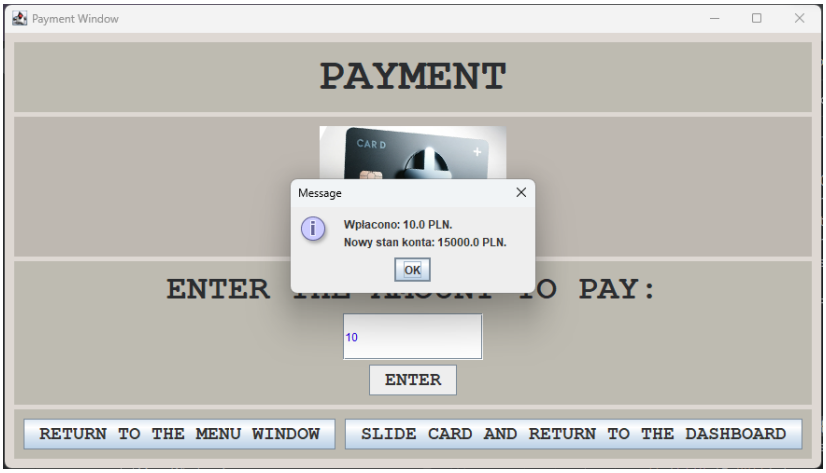


Rysunek 22, 23. Zdjęcie przedstawiające PaycheckWindow i alert braku środków.

Okno to służy do wypłaty środków z konta użytkownika. Funkcjonalność ta jest widoczna w kodzie w klasie PaycheckWindow, gdzie użytkownik wpisuje kwotę, którą chce wypłacić, a następnie potwierdza operację przyciskiem "ENTER". Po wykonaniu operacji:

- Jeśli wypłata jest możliwa (saldo na koncie pozwala na to), wyświetlany jest komunikat z nowym stanem konta po transakcji.

- Jeśli próba wypłaty przekracza dostępne środki lub jest nieprawidłowa, użytkownik otrzymuje komunikat o błędzie.



Rysunek 24. Zdjęcie przedstawiające PaymentWindow.

Okno „**PaymentWindow**” działa podobnie do PaycheckWindow, ale służy do wpłacania pieniędzy na konto użytkownika. Użytkownik wpisuje kwotę do wpłaty i potwierdza operację. Kod zawiera logikę, która aktualizuje stan konta po wpłacie środków i wyświetla stosowny komunikat z nowym saldem.

Funkcjonalność wspólna dla obu okien: Pole tekstowe: Miejsce, gdzie użytkownik może wpisać kwotę do wypłaty lub wpłaty.

Przycisk "ENTER": Służy do zatwierdzenia wprowadzonej kwoty i inicjacji operacji finansowej.

Przyciski nawigacyjne: Pozwalają użytkownikowi na powrót do głównego menu (RETURN TO THE MENU WINDOW) lub do głównego ekranu aplikacji (SLIDE CARD AND RETURN TO THE DASHBOARD).

Kod i jego działanie:

Klasy PaycheckWindow i PaymentWindow wykorzystują metody takie jak withdrawMoney() i depositMoney() do przetwarzania transakcji.

Podczas przetwarzania:

- Sprawdzają, czy wprowadzona kwota jest prawidłowa i czy stan konta pozwala na realizację transakcji.
- Aktualizują stan konta w bazie danych za pomocą zapytań SQL.
- Rejestrują transakcję w historii konta.
- Wyświetlają stosowne komunikaty z wynikiem operacji.

Walidacja i bezpieczeństwo:

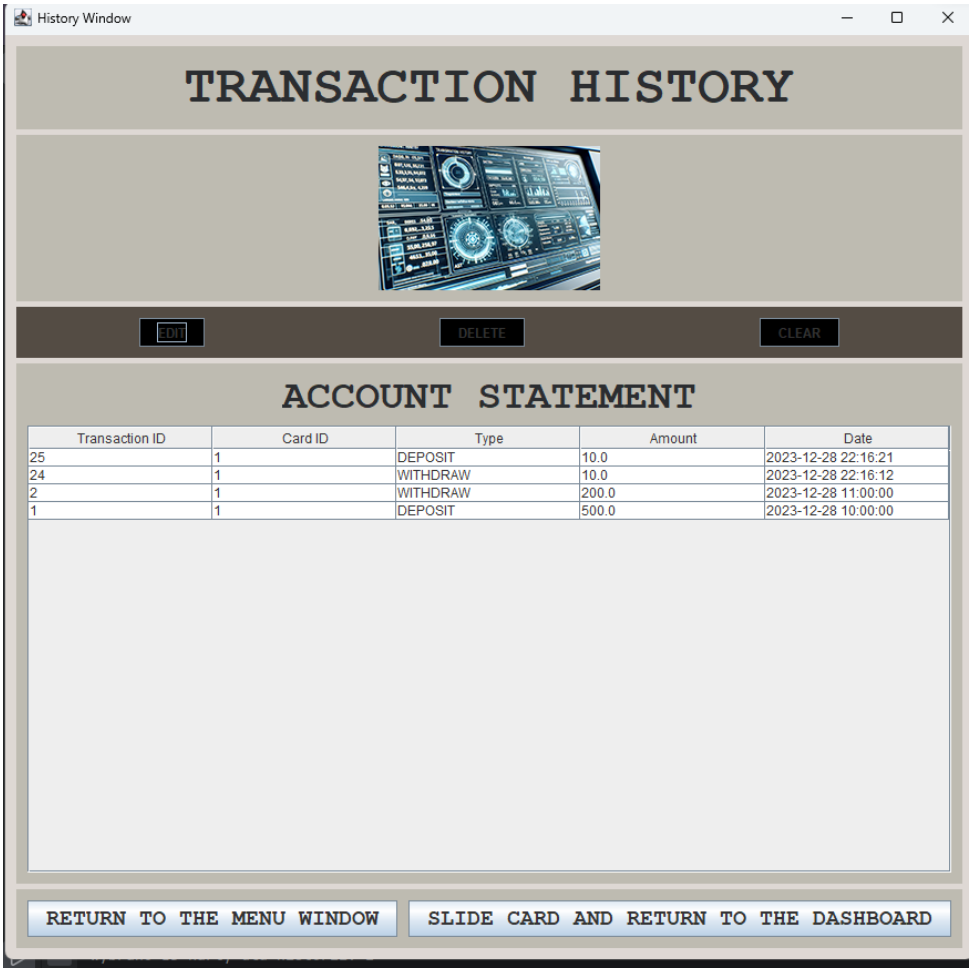
Kod zawiera mechanizmy walidacji wprowadzonych danych oraz zabezpieczenia przed przekroczeniem dostępnych środków. W przypadku błędów walidacji, użytkownik otrzymuje komunikaty, które informują go o błędzie i pozwalają na korektę akcji.

Ogólna analiza:

Oba okna dialogowe są zaprojektowane w sposób prosty i intuicyjny, co sprzyja łatwemu korzystaniu z funkcji bankowych bez konieczności bezpośredniego kontaktu z

personelem banku. Jest to wygodna funkcja w aplikacjach bankowych, pozwalająca użytkownikom na szybkie zarządzanie swoimi środkami finansowymi.

HistoryWindow jest oknem w aplikacji bankowej, które wyświetla historię transakcji przeprowadzonych przez użytkownika.



Rysunek 25. Zdjęcie przedstawiające HistoryWindow.

Główne Elementy **HistoryWindow**:

Nagłówek "TRANSACTION HISTORY": Informuje użytkownika, że znajduje się w sekcji historii transakcji.

Tabela "ACCOUNT STATEMENT": Wyświetla listę transakcji zawierających ID transakcji, ID karty, typ transakcji (wpłata, wypłata), kwotę i datę.

Przyciski "EDIT", "DELETE", "CLEAR":

- EDIT: Umożliwia edycję wybranej transakcji.

- DELETE: Umożliwia usunięcie wybranej transakcji z historii.

- CLEAR: Czyści zaznaczone wiersze w tabeli.

Przyciski nawigacyjne: RETURN TO THE MENU WINDOW: Pozwala użytkownikowi na powrót do głównego menu operacji (MenuWindow).

SLIDE CARD AND RETURN TO THE DASHBOARD: Umożliwia wyjście z obecnego widoku i powrót do głównego menu aplikacji (Dashboard).

Kod i Jego Funkcjonalność: W kodzie, klasa HistoryWindow jest odpowiedzialna za wyświetlenie historii transakcji użytkownika. Używa ona połączenia z bazą danych do pobrania historii transakcji i wyświetla je w tabeli. Funkcjonalność tej klasy obejmuje:

- Ładowanie historii transakcji: Metoda loadTransactionHistory() pobiera z bazy danych i wyświetla transakcje.

- Edycja i usuwanie transakcji: Przyciski "EDIT" i "DELETE" są połączone z funkcjami, które pozwalają na modyfikowanie i usuwanie rekordów z tabeli. Usunięcie rekordu jest odzwierciedlone również w bazie danych.

- Czyszczenie danych: Przycisk "CLEAR" pozwala na wyczyszczenie danych z tabeli, ale niekoniecznie usuwa je z bazy danych.

Walidacja i Bezpieczeństwo: Kod zawiera mechanizmy walidacji i zabezpieczeń, aby upewnić się, że operacje edycji i usuwania są wykonane prawidłowo i że użytkownik posiada odpowiednie uprawnienia do wykonania tych akcji.

Interakcje z Użytkownikiem: Przyciski "EDIT", "DELETE" i "CLEAR" w tabeli mogą wywoływać okna dialogowe lub dodatkowe formularze, gdzie użytkownik może edytować lub potwierdzić swoje działania.

Ogólna Analiza:

HistoryWindow umożliwia użytkownikowi przeglądanie i zarządzanie jego historią transakcji, co jest kluczową funkcją dla aplikacji bankowych. Użytkownik ma możliwość wglądu w swoją aktywność finansową, co pozwala na lepsze zarządzanie finansami. Kod za tym oknem zapewnia niezbędne funkcje do interakcji z bazą danych, obsługi zdarzeń interfejsu użytkownika oraz prezentacji danych w sposób czytelny i dostępny.

PODSUMOWANIE:

Cały program, który został opisany przez różne okna interfejsu użytkownika i odpowiadający im kod, jest kompleksowym systemem zarządzania kontem bankowym, prawdopodobnie przeznaczonym dla bankomatów lub aplikacji bankowych online. Oto podsumowanie ogólnej funkcjonalności i działania programu:

Główne Funkcje: Autentykacja Użytkownika: System rozpoczyna się od autentykacji użytkownika przy użyciu PinWindow, gdzie użytkownik musi wprowadzić swój kod PIN, aby uzyskać dostęp do swojego konta.

Dashboard: Po pomyślnym zalogowaniu się, użytkownik jest przenoszony do Dashboard, który działa jako centrum nawigacyjne, oferując różne opcje, takie jak wyświetlanie salda, wypłaty, wpłaty i historii transakcji.

Operacje Konta: BalanceWindow: Umożliwia użytkownikowi sprawdzenie bieżącego salda konta.

PaycheckWindow: Pozwala na wypłatę określonej kwoty z konta, z odpowiednimi walidacjami, aby zapobiec przekroczeniu dostępnych środków.

PaymentWindow: Umożliwia wpłatę pieniędzy na konto użytkownika.

HistoryWindow: Prezentuje użytkownikowi historię wykonanych transakcji, dając wgląd w aktywność konta oraz umożliwiając zarządzanie tymi transakcjami.

Zarządzanie Danymi: Eksport i import danych w formacie CSV, umożliwiający backup i transfer danych między systemami. Edycja i usuwanie rekordów w historii transakcji, co pozwala na utrzymanie porządku i precyzyjne śledzenie aktywności finansowej.

Bezpieczeństwo i Walidacja: Program zawiera szereg mechanizmów bezpieczeństwa, w tym blokadę konta po kilku nieudanych próbach wprowadzenia PIN-u, walidację kwot transakcji w celu uniknięcia nadmiernego zadłużenia, oraz potwierdzenia i komunikaty błędów, które informują użytkownika o stanie operacji.

Użytkownik i Interfejs: Interfejs użytkownika został zaprojektowany z myślą o prostocie i intuicyjności, co pozwala użytkownikowi na łatwe nawigowanie i wykonanie potrzebnych operacji finansowych bez konieczności zaawansowanej wiedzy technicznej.

Program jako całość stanowi kompletny system do zarządzania kontem bankowym, który integruje kluczowe funkcje wymagane dla nowoczesnych aplikacji bankowych. Zapewnia on użytkownikowi kontrolę nad swoimi finansami poprzez proste w użyciu narzędzia do monitorowania i zarządzania swoimi środkami oraz historią transakcji. Jest to przykład dobrze zintegrowanego oprogramowania, które skupia się na użytkowniku i jego potrzebach, zapewniając przy tym bezpieczeństwo i zaufanie do przeprowadzanych operacji finansowych.

6. Podsumowanie

Opracowanie aplikacji GUI: Stworzono zaawansowaną aplikację z graficznym interfejsem użytkownika, która naśladuje działanie rzeczywistego bankomatu. Interfejs został zaprojektowany tak, aby był intuicyjny i łatwy w obsłudze, co umożliwia użytkownikom wykonywanie różnych operacji bankowych w sposób, który odzwierciedla rzeczywiste doświadczenia z bankomatem.

Wykorzystanie Java Swing i MySQL: Aplikacja została zbudowana w środowisku Java Swing, co zapewnia stabilność i efektywność działania. Zastosowanie MySQL do zarządzania bazą danych gwarantuje bezpieczeństwo i integralność danych, co jest kluczowe w symulacji operacji bankowych.

Implementacja funkcji bankowych: Zaimplementowano podstawowe funkcje bankomatu, takie jak weryfikacja PIN, sprawdzanie salda, realizacja wypłat i wpłat gotówki oraz przeglądanie historii transakcji. Te funkcje zapewniają realistyczne doświadczenie korzystania z bankomatu, co jest niezbędne w szkoleniach i edukacji pracowników banków.

Dodatkowe funkcje dla użytkownika: Wprowadzono unikalne funkcje, takie jak regulacja głośności i możliwość eksportu/importu danych bazy, co zwiększa funkcjonalność aplikacji i poprawia doświadczenie użytkownika.

Planowany rozwój funkcjonalności: Projekt przewiduje rozszerzenie funkcjonalności o nowe opcje bankowe i usprawnienia związane z bezpieczeństwem. Planowane jest także ulepszenie interfejsu użytkownika, aby był jeszcze bardziej intuicyjny i dostępny.

Integracja z systemami bankowymi i testowanie: Planuje się integrację z dodatkowymi systemami bankowymi, co pozwoli na szersze wykorzystanie aplikacji w rzeczywistym środowisku bankowym. Rozbudowa możliwości testowania aplikacji pozwoli na lepsze przygotowanie do rzeczywistych scenariuszy funkcjonowania bankomatów.

Zaawansowane techniki szyfrowania i zabezpieczeń: W celu ochrony danych użytkowników, planowane jest wdrożenie zaawansowanych technik szyfrowania i zabezpieczeń, co jest kluczowe w kontekście ochrony informacji finansowych i osobowych.

Dalsze planowane prace rozwojowe projektu obejmują:

1. **Integracja z mobilnymi systemami płatności**: Rozszerzenie funkcjonalności bankomatu o obsługę płatności zbliżeniowych i mobilnych, co zwiększy dostępność i wygodę dla użytkowników.
2. **Zastosowanie uczenia maszynowego do profilowania ryzyka**: Implementacja algorytmów uczenia maszynowego w celu wykrywania podejrzanych transakcji i zapobiegania oszustwom.
3. **Rozbudowa interfejsu użytkownika**: Modernizacja UI/UX, aby była bardziej intuicyjna i przyjazna dla użytkownika, potencjalnie z wykorzystaniem technologii dotykowych i graficznych.
4. **Wprowadzenie funkcji personalizacji**: Pozwolenie użytkownikom na dostosowanie interfejsu bankomatu do indywidualnych preferencji, np. wybór języka czy wyglądu ekranu.
5. **Zwiększenie interaktywności**: Dodanie interaktywnych tutoriali pomagających nowym użytkownikom w nauczeniu się obsługi bankomatu.
6. **Bezpieczeństwo cybernetyczne**: Wzmocnienie zabezpieczeń przez wprowadzenie nowoczesnych rozwiązań kryptograficznych i multi-faktorowej autentykacji.
7. **Ekspansja na nowe rynki**: Dostosowanie oprogramowania do wymagań i specyfikacji różnych krajów i instytucji finansowych, co pozwoli na globalne wdrożenie systemu.
8. **Wdrażanie aktualizacji**: Regularne aktualizacje oprogramowania w celu wprowadzania nowych funkcji i utrzymania zgodności z najnowszymi standardami branżowymi.

Każdy z tych punktów można by dalej rozwijać, uwzględniając specyficzne wymagania i cele projektu.

7. Literatura

W celu zgłębienia wiedzy na temat systemów bankomatów oraz ogólnie systemów transakcyjnych, można sięgnąć do następujących źródeł:

Książki i Podręczniki:

1. "Bank 3.0: Why Banking Is No Longer Somewhere You Go But Something You Do" - Brett King. Książka ta dostarcza wglądu w zmiany technologiczne w bankowości i przyszłość transakcji finansowych.

2. "The ATM and the Internet of Things" - Richard P. Smyth. Pozycja ta omawia rozwój bankomatów w kontekście rosnącej sieci urządzeń połączonych przez Internet.

3. "Designing the Digital Experience: How to Use EXPERIENCE DESIGN Tools & Techniques to Build Websites Customers Love" - David Lee King. Ta książka jest przydatna do zrozumienia jak projektować przyjazne użytkownikowi interfejsy dla aplikacji bankowych.

Artykuły:

1. "How ATMs Work" - Artykuł w czasopiśmie "How Stuff Works", który szczegółowo wyjaśnia działanie bankomatów.

2. "The Evolution of Automated Teller Machines" - Artykuł naukowy omawiający historię i rozwój bankomatów.

Linki do Stron WWW:

1. Official Documentation of Java Platform, Standard Ed. - Dokumentacja Java SE dostarcza informacji na temat używania języka Java, który jest często wykorzystywany do tworzenia aplikacji bankomatów.

Link: https://docs.oracle.com/javase/8/docs/api/ (data dostępu: 19.01.2024)

2. Stack Overflow - Społeczność programistów, gdzie można znaleźć odpowiedzi na wiele pytań związanych z problemami programistycznymi, w tym związanymi z tworzeniem oprogramowania bankowego.

Link: https://stackoverflow.blog (data dostępu: 19.01.2024)

3. GitHub - Na tej platformie można znaleźć projekty open-source związane z systemami bankomatów, które mogą służyć jako przykład lub punkt wyjścia dla własnych projektów.

Link: https://github.com/ (data dostępu: 19.01.2024)

Korzystanie z wymienionych źródeł znacząco wzbogaciło moją wiedzę na temat systemów bankomatów, ich bezpieczeństwa, interfejsu użytkownika oraz najlepszych praktyk w tworzeniu oprogramowania w tej dziedzinie.