AKADEMIA
NAUK STOSOWANYCH
W ŁOMŻY

Wydział Nauk Informatyczno-Technologicznych

Projekt aplikacji

Temat: Aplikacja Bankowa

Prowadzący: Przemysław Grabowski

Wykonujący projekt:

Adrian Kozłowski, Dawid Kucisz, Piotr Gałązka, Artur Lubiński

Studia Stacjonarne I stopnia

Kierunek: Informatyka

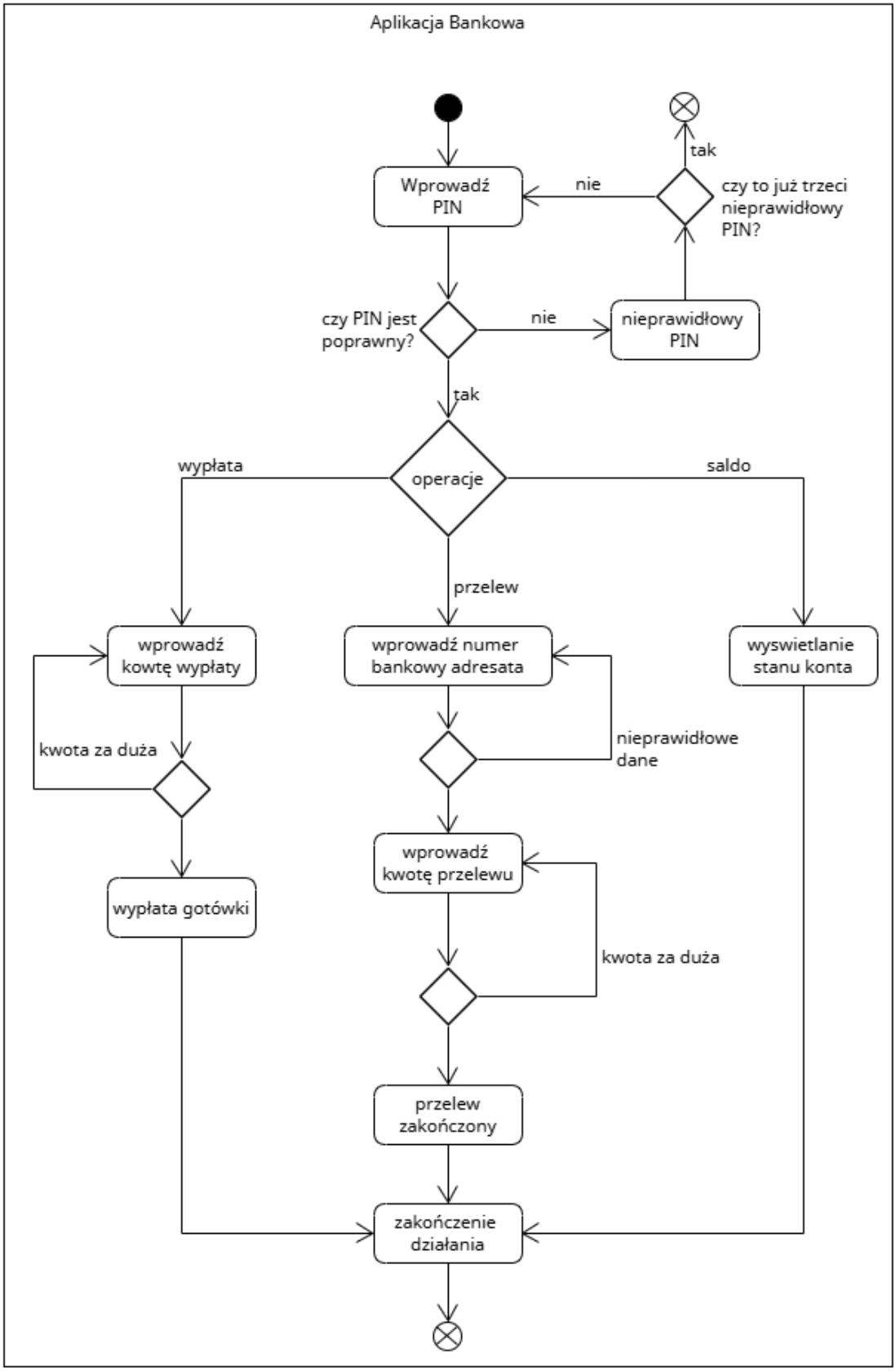
Semestr: IV, grupa I

Data: 04.05.2025

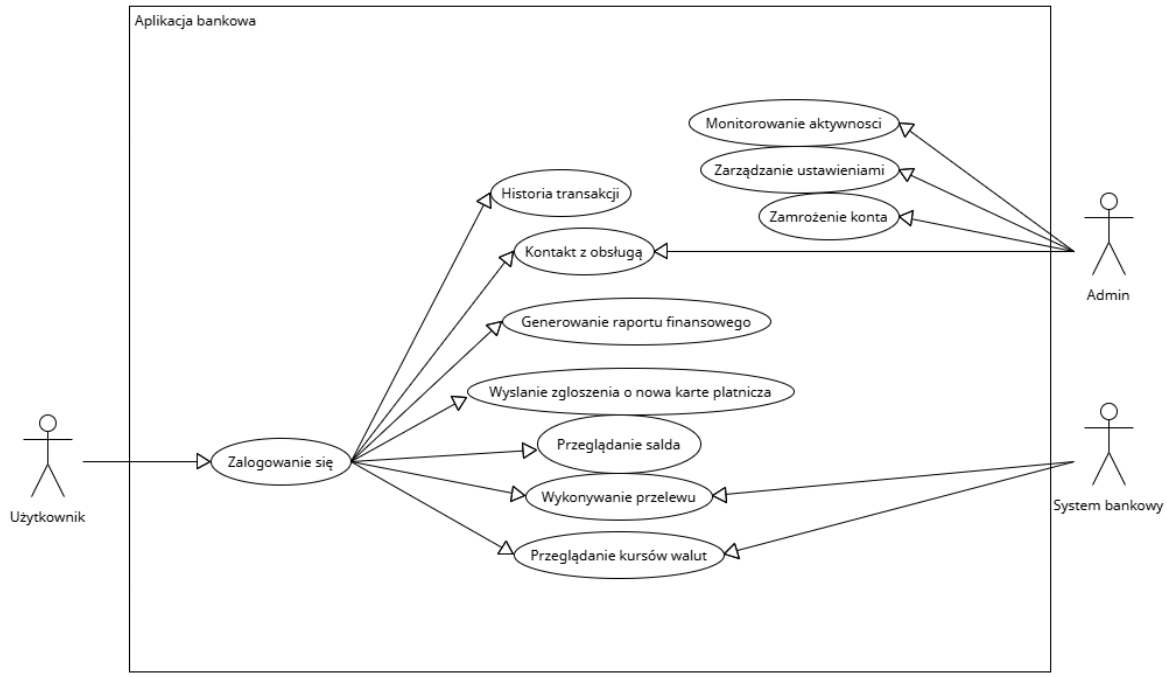
Spis treści

Diagramy

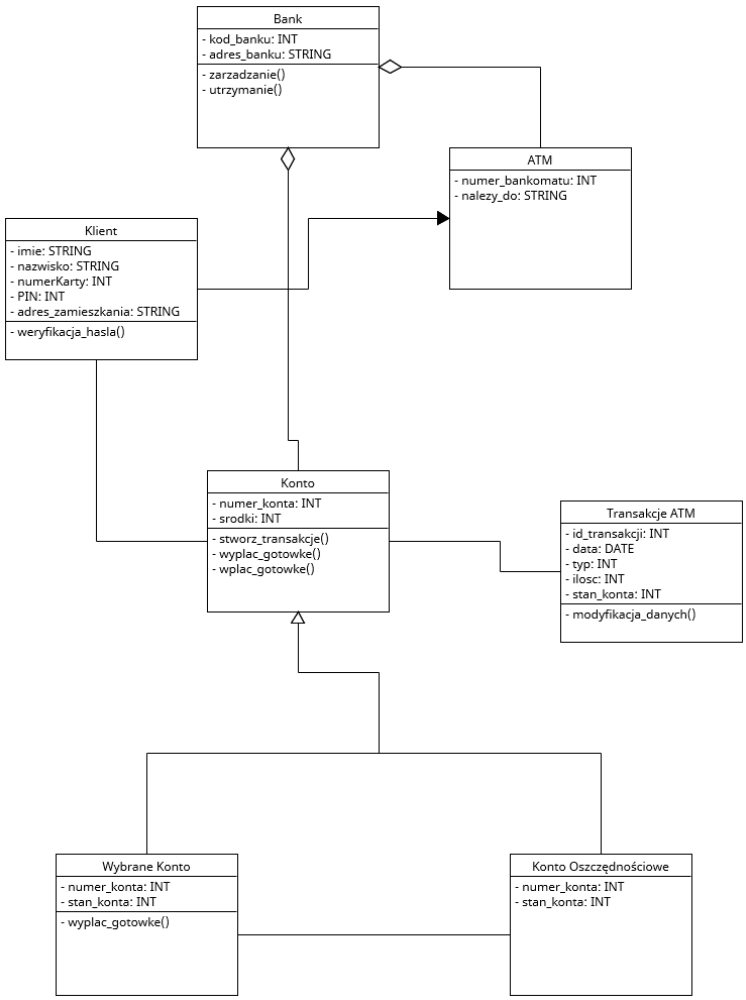
1. Diagram Czynności



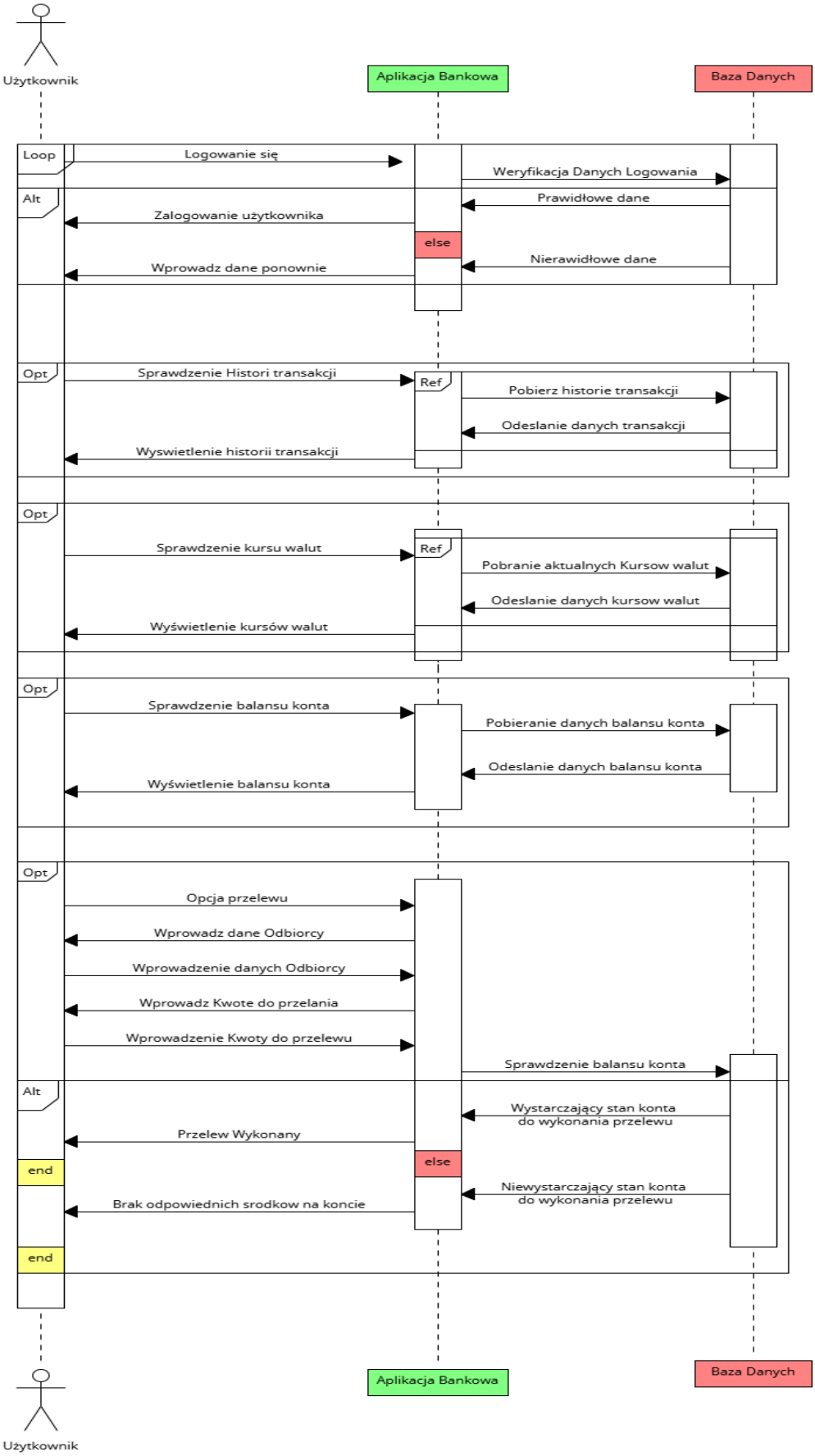
2. Diagram Przypadków użycia



3. Diagram Klas



4. Diagram Sekwencji



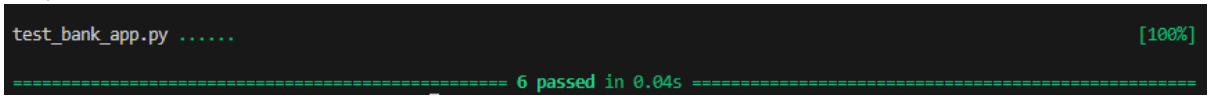
Funkcje Programu

```
class Account:
    def __init__(self, account_id, balance=0.0):
        self.account_id = account_id
        self.balance = balance
        self.transactions = []

    def get_balance(self):
        return self.balance

    def add_funds_via_transfer(self, amount):
        if amount <= 0:
            raise ValueError("Amount must be positive")
        self.balance += amount
        self.transactions.append(f"Incoming transfer: {amount}")
        return self.balance

    def schedule_payment(self, amount):
        if amount <= 0:
            raise ValueError("Amount must be positive")
        if amount > self.balance:
            raise ValueError("Insufficient funds")
        self.balance -= amount
        self.transactions.append(f"Scheduled payment: {amount}")
        return self.balance

    def transfer(self, target_account, amount):
        self.schedule_payment(amount)
        target_account.add_funds_via_transfer(amount)
        self.transactions.append(f"Transfer to {target_account.account_id}: {amount}")
        return self.balance

    def get_transaction_history(self):
        return self.transactions


class User:
    def __init__(self, username, password):
        self.username = username
        self.password = password
        self.logged_in = False

    def login(self, username, password):
        self.logged_in = self.username == username and self.password == password
        return self.logged_in
```

Testy

```
import pytest
from bank_app import Account, User

def test_add_funds_via_transfer():
    acc = Account(1, 100)
    assert acc.add_funds_via_transfer(50) == 150
    assert acc.add_funds_via_transfer(1) == 151
    with pytest.raises(ValueError):
        acc.add_funds_via_transfer(0)
    with pytest.raises(ValueError):
        acc.add_funds_via_transfer(-10)

def test_schedule_payment():
    acc = Account(1, 100)
    assert acc.schedule_payment(50) == 50
    assert acc.schedule_payment(10) == 40
    with pytest.raises(ValueError):
        acc.schedule_payment(0)
    with pytest.raises(ValueError):
        acc.schedule_payment(100)

def test_transfer():
    acc1 = Account(1, 200)
    acc2 = Account(2, 50)
    assert acc1.transfer(acc2, 50) == 150
    assert acc2.get_balance() == 100
    with pytest.raises(ValueError):
        acc1.transfer(acc2, 300)
    with pytest.raises(ValueError):
        acc1.transfer(acc2, -5)

def test_get_balance():
    acc = Account(1, 123.45)
    assert acc.get_balance() == 123.45
    acc.add_funds_via_transfer(10)
    assert acc.get_balance() == 133.45
    acc.schedule_payment(3.45)
    assert acc.get_balance() == 130.0

def test_get_transaction_history():
    acc = Account(1)
    acc.add_funds_via_transfer(100)
    acc.schedule_payment(30)
    history = acc.get_transaction_history()
    assert "Incoming transfer: 100" in history
    assert "Scheduled payment: 30" in history
    assert len(history) == 2

def test_login():
    user = User("john", "1234")
    assert user.login("john", "1234") is True
    assert user.logged_in is True
    assert user.login("john", "wrong") is False
    assert user.login("wrong", "1234") is False
    assert user.login("", "") is False
```

Wyniki testów



Podział Obowiązków

Diagram czynności: Piotr Gałązka

Diagram przypadków użycia: Adrian Kozłowski

Diagram klas: Artur Lubiński

Diagram sekwencji: Dawid Kucisz

Funkcje i testy programu: Adrian Kozłowski, Dawid Kucisz, Piotr Gałązka, Artur Lubiński