

Bazy Danych 2 – opis projektu

1. Opis ogólny

Tematem naszego projektu jest baza danych dla wypożyczalni rowerów. Użytkownicy tej bazy to oczywiście ludzie chcący wypożyczyć rower, a także pracownicy- będą oni odpowiadać za jej obsługę np. aktualizacja lub dodawanie nowych danych zwłaszcza tych niedostępnych dla przeciętnego użytkownika (np. aktualizacja cen, dodanie nowego modelu roweru) oraz jej nadzorowanie (administrator będzie miał dostęp prawie do wszystkich danych na serwerze poza prywatnymi danymi użytkownika).

Baza danych będzie przechowywać różne rodzaje danych. Pierwsze z nich to podstawowe dane użytkownika (adres itp.). Każdy użytkownik będzie miał dostęp do swojej historii wypożyczeń, swojego stanu konta oraz ew. swojego statusu (czy ma jakieś sprawy do załatwienia tzn. np. wpłacenie kary za nieoddanie roweru w czasie, wprowadzenie opłaty inicjalnej itp.).

Każdy z rowerów musi być w bazie danych zarejestrowany (posiada swój numer identyfikacyjny). Każdy z nich będzie w bazie danych zawierał krótką charakterystykę (np. model, typ roweru). Oprócz tego zawarte będą takie dane jak ostatni i ogólny pokonany dystans, stan techniczny, ilość wypożyczeń, lokalizacja (jest lub nie jest w stacji) itd. Osobno będzie przechowywana lista modeli i typów rowerów udostępniana przez wypożyczalnię a także liczebność danego modelu. Rowery elektryczne mogą mieć także informacje o stopniu naładowania.

W bazie będą też przechowywane informacje o stacjach rowerowych. Znajdzie się tam jej numer, adres oraz liczba przechowywanych rowerów.

Użytkownik to osoba korzystająca z aplikacji w celu wypożyczenia roweru na określony czas za odpowiednią opłatą. Każdy z użytkowników jak to wcześniej zostało powiedziane będzie miał dostęp do swojej historii wypożyczeń. Administrator systemu będzie jednak mógł uzyskać wgląd do całej historii (dla wszystkich modeli). Każdy z takich wpisów będzie określał czas wypożyczenia. Jednocześnie będzie podany koszt w zależności od czasu.

Administrator jest pracownikiem firmy, który może dokonywać zmian w tabeli cen i opłat (użytkownik może jedynie wyświetlać te dane) i zarządzać usterkami i stacjami rowerowymi w systemie.

Dodatkową informacją w bazie danych są usterki. Nie wszystkie rowery muszą znajdować się w stacjach, część z nich może być uszkodzona lub niedziałająca i wymagająca naprawy. Baza będzie więc zawierała dane odnośnie usterek (wgląd tylko dla administratora)

2. Przykładowe czynności i procesy dokonywane w systemie.

a) Użytkownik wypożycza rower:

- Wypożyczenie odnotowywane jest w systemie

- Zmiana stanu roweru na nieznajdujący się w stacji. Jeśli rower nie zostanie oddany do stacji i wypożyczenie zostanie zakończone przez użytkownika to oprócz normalnego kosztu zostanie naliczona dodatkowa opłata.

- Po odstawieniu naliczana jest opłata (znajduje się ona również w historii rozliczeń), naliczany jest przejechany dystans

b) Pracownik dodaje nowy model/typ roweru do bazy

c) Użytkownik rejestruje się do bazy, pracownik dodaje użytkownika

d) Pracownik nadzoruje bazę danych- sprawdza czy wszystkie sprawy przez użytkownika zostały załatwione (np. rozliczenia, odstawienie roweru), może chcieć uzyskać wgląd do historii wypożyczeń (np. pracownik chce obliczyć dzienny przychód wypożyczalni)

e) Użytkownik ma wgląd do swojego profilu. Ma możliwość wyświetlenia historii wypożyczeń, tabeli rozliczeń, może zmienić swoje dane. Niektóre dane mogą być zmienione tylko przez pracownika

f) Użytkownik i pracownik mają dostęp do stacji rowerowych. Użytkownik ma tylko na nie wgląd. Pracownik może dodatkowo modyfikować dane stacji.

g) Pracownik może dodawać/usuwać i wyświetlać usterki. Użytkownik nie ma do tego dostępu, natomiast może jedynie zgłaszać usterki.

h) Pracownik może modyfikować dane w tabeli cen i opłat. Użytkownik może je tylko wyświetlić

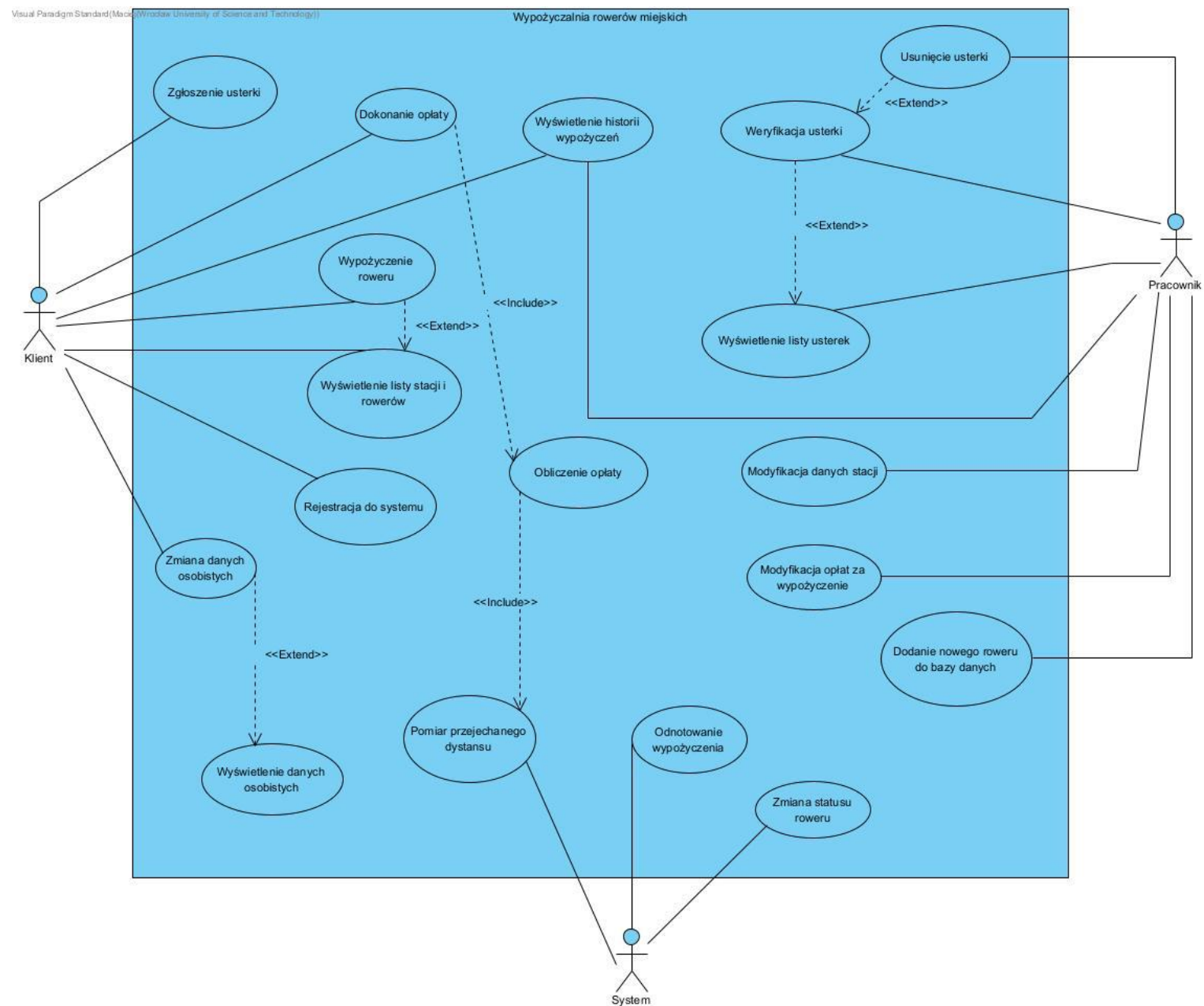
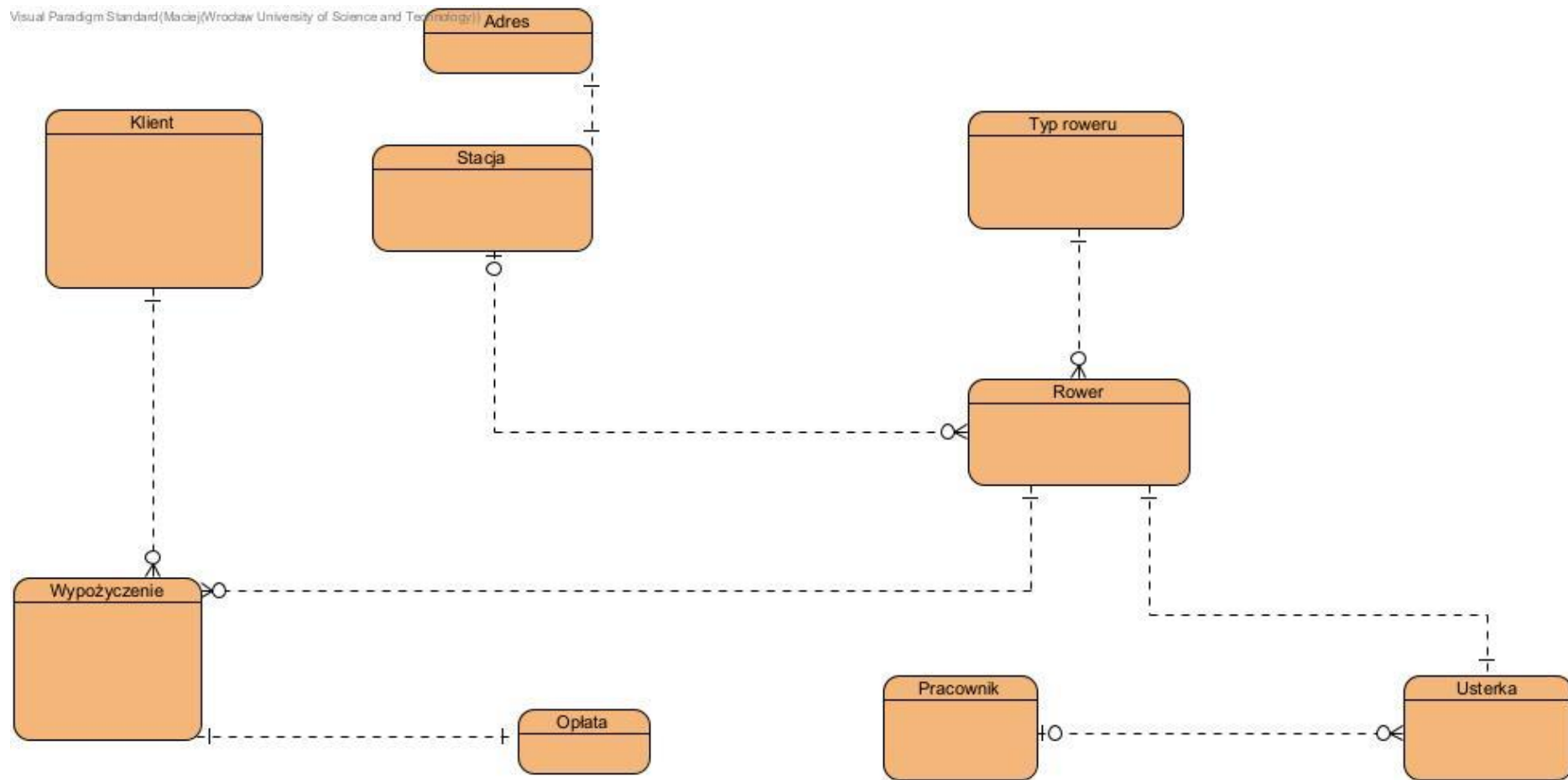
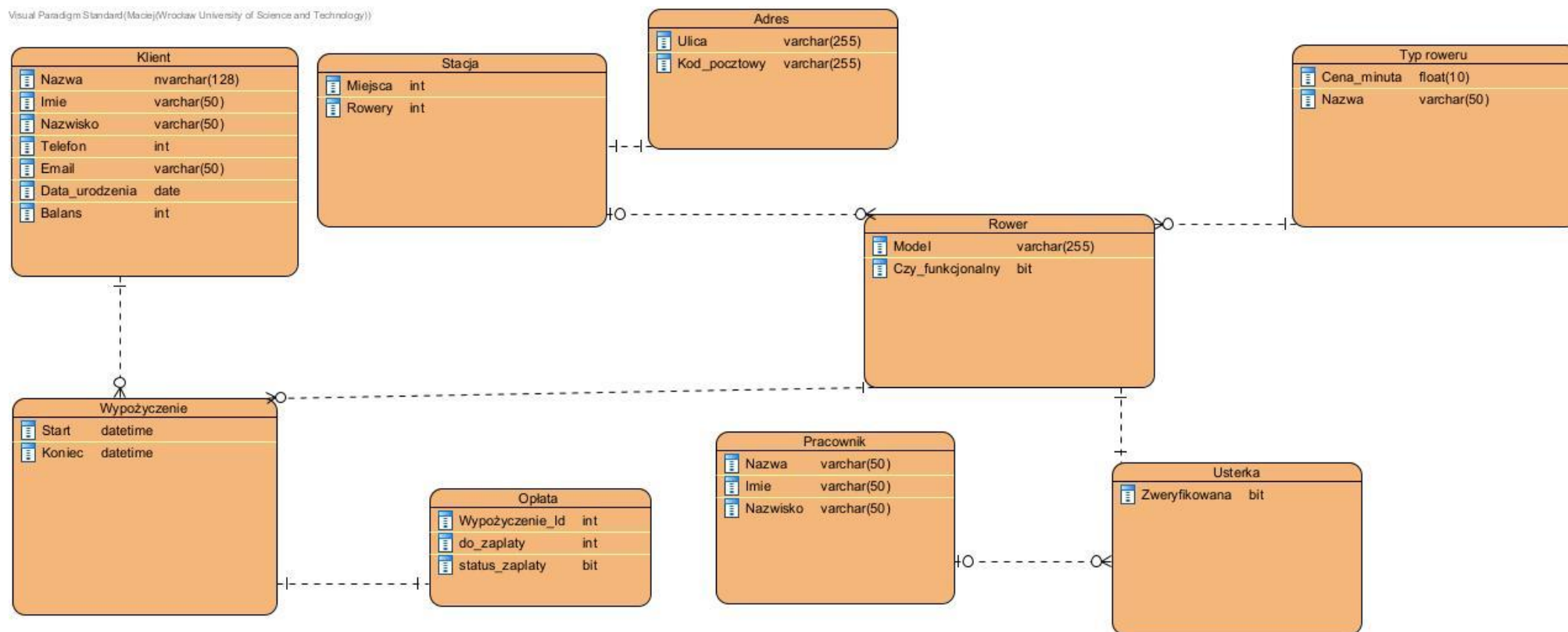


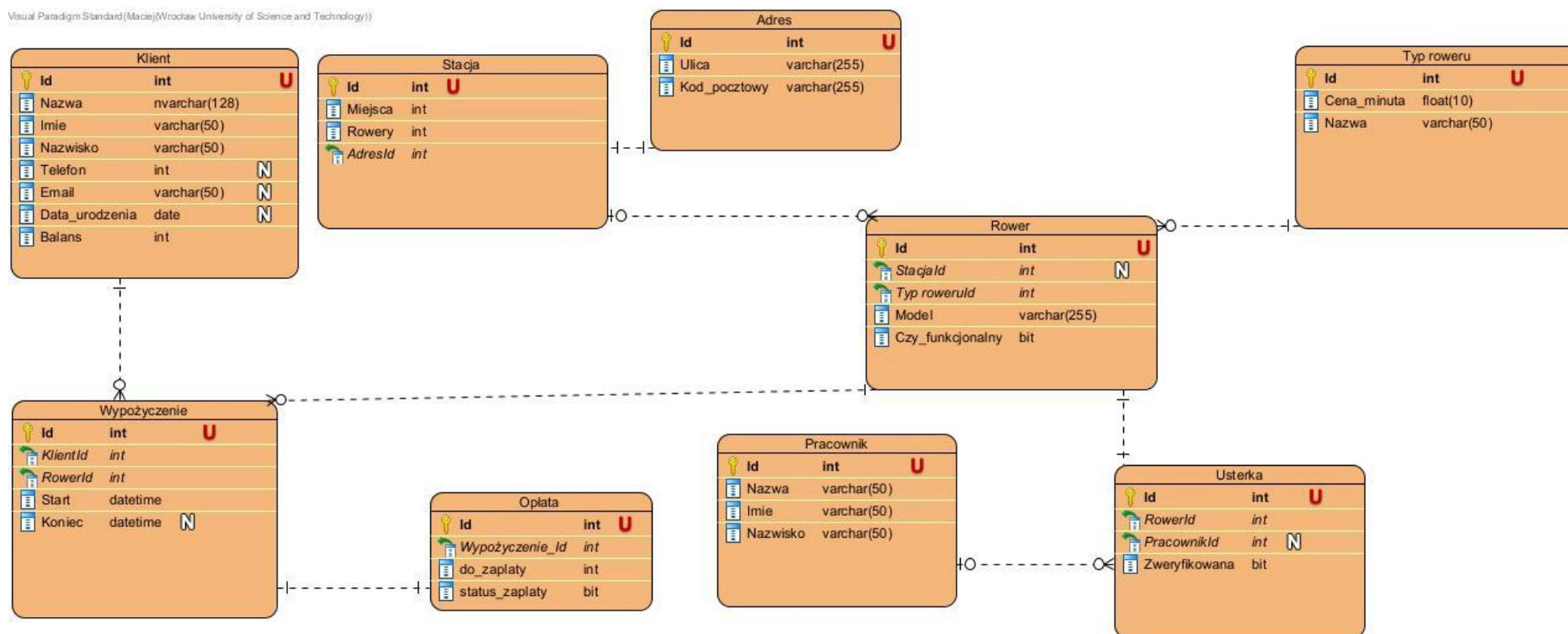
Diagram przypadków użycia



Rysunek 1 Diagram związków encji



Rysunek 2 Model logiczny




Rysunek 3 Model fizyczny

W bazie danych zdefiniowano dwie role:


1. Klient może:
 - wyświetlić swoje dane z tabeli Klienci
 - wyświetlić dane z tabeli Stacja i Rower
 - wyświetlić dane z tabel Adres i Typ_roweru połączone kluczami obcymi z odpowiednio tabelami Stacja i Rower
2. Pracownik:
 - Wyświetlić, modyfikować, wstawiać i usuwać dane ze wszystkich tabel oprócz tabeli Pracownik

Sprawdzenie formatu danych oraz kluczy głównych i obcych:


Adres:

	Column Name	Data Type	Allow Nulls
	Id	int	<input type="checkbox"/>
	Ulica	varchar(50)	<input type="checkbox"/>
	Kod_pocztowy	varchar(50)	<input type="checkbox"/>
			<input type="checkbox"/>

Klient:

	Column Name	Data Type	Allow Nulls
	Id	int	<input type="checkbox"/>
	Nazwa	nvarchar(128)	<input type="checkbox"/>
	Imie	varchar(50)	<input type="checkbox"/>
	Nazwisko	varchar(50)	<input type="checkbox"/>
	Telefon	int	<input checked="" type="checkbox"/>
	Email	varchar(50)	<input checked="" type="checkbox"/>
	Data_urodzenia	date	<input checked="" type="checkbox"/>
	Balans	float	<input checked="" type="checkbox"/>

Pracownik:

	Column Name	Data Type	Allow Nulls
	Id	int	<input type="checkbox"/>
	Nazwa	varchar(50)	<input type="checkbox"/>
	Imie	varchar(50)	<input type="checkbox"/>
	Nazwisko	varchar(50)	<input type="checkbox"/>
			<input type="checkbox"/>

Wypożyczenie:

	Column Name	Data Type	Allow Nulls
🔑	Id	int	<input type="checkbox"/>
	Klient_Id	int	<input type="checkbox"/>
	Rower_Id	int	<input type="checkbox"/>
	Start_wypozyczenia	datetime	<input type="checkbox"/>
	Koniec_wypozyczenia	datetime	<input checked="" type="checkbox"/>

Opłata:

	Column Name	Data Type	Allow Nulls
🔑	Id	int	<input type="checkbox"/>
	Wypozyczenie_Id	int	<input type="checkbox"/>
	do_zaplaty	float	<input type="checkbox"/>
	status_zaplaty	bit	<input type="checkbox"/>
			<input type="checkbox"/>


Stacja:

	Column Name	Data Type	Allow Nulls
🔑	Id	int	<input type="checkbox"/>
	Adres_Id	int	<input type="checkbox"/>
	Miejsca	int	<input type="checkbox"/>
	Rowery	int	<input type="checkbox"/>


Rower:

	Column Name	Data Type	Allow Nulls
🔑	Id	int	<input type="checkbox"/>
	Typ_roweru_Id	int	<input type="checkbox"/>
	Stacja_Id	int	<input checked="" type="checkbox"/>
	Model	varchar(50)	<input type="checkbox"/>
	Czy_funkcjonalny	bit	<input type="checkbox"/>

Typ roweru:

	Column Name	Data Type	Allow Nulls
	Id	int	<input type="checkbox"/>
	Cena_minuta	float	<input type="checkbox"/>
	Nazwa	varchar(50)	<input type="checkbox"/>

Usterka:

	Column Name	Data Type	Allow Nulls
	Id	int	<input type="checkbox"/>
	Zweryfikowana	bit	<input type="checkbox"/>
	Rower_Id	int	<input type="checkbox"/>
	Pracownik_Id	int	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Test procedur w bazie danych:

- Dodanie roweru do bazy przez pracownika:

```

USE [RowerMiejski]
GO
/***** Object: StoredProcedure [dbo].[dodaj_rower]    Script Date: 03.12.2022 19:28:57 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:      <Author,,Name>
-- Create date: <Create Date,,>
-- Description: <Description,,>
-- =====
ALTER PROCEDURE [dbo].[dodaj_rower] @Typ_roweru int, @Stacja int, @Model varchar(50)
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    INSERT INTO Rower(Typ_roweru_Id, Stacja_Id, Model, Czy_funkcjonalny)
    VALUES(@Typ_roweru, @Stacja, @Model, 1)

END

```

Rysunek 1 Kod procedury wstawiania roweru

```
USE [RowerMiejski]
GO

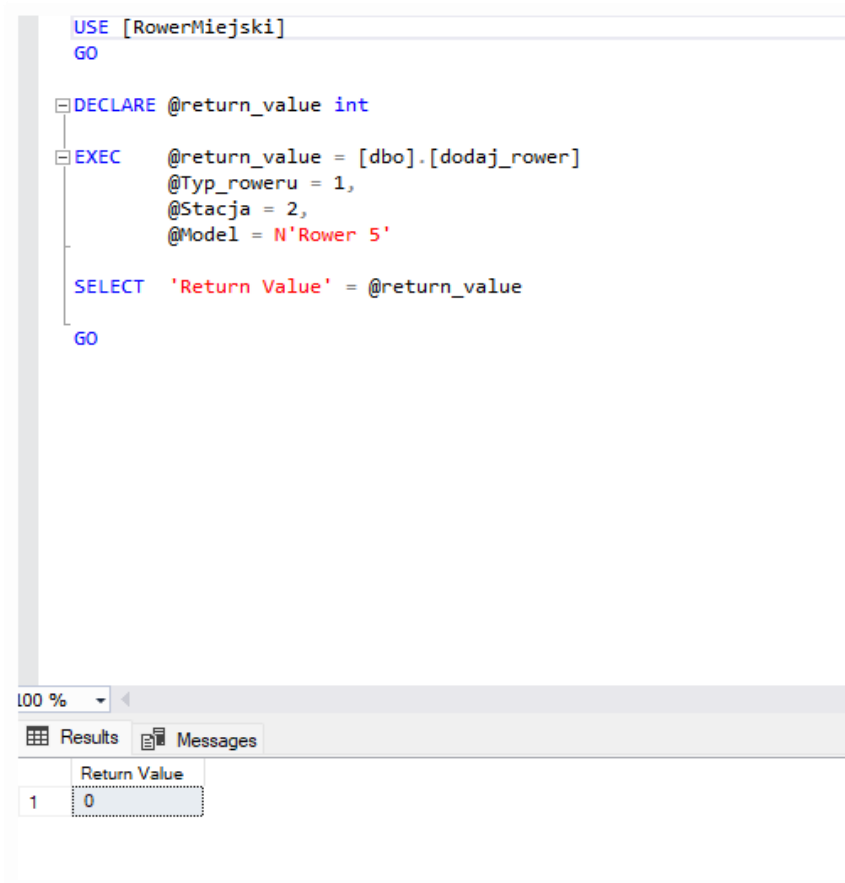
EXEC [dbo].[wyswietl_rowery_na_stacji]
@stacja = 2
GO
```

100 %

Results Messages

	Rodzaj roweru	Ulica	Kod_pocztowy	Model	Czy jest funkcjonalny
1	SuperTyp	Wroblewskiego 25	51-867	Rower 2	1

Rysunek 2 Stacja przed wstawieniem roweru



Rysunek 3 Wstawiamy rower o danych atrybutach

USE [RowerMiejski]
GO

EXEC [dbo].[wyswietl_rowery_na_stacji]
@stacja = 2
GO

100 %

Results Messages

	Rodzaj roweru	Ulica	Kod_pocztowy	Model	Czy jest funkcjonalny
1	SuperTyp	Wroblewskiego 25	51-867	Rower 2	1
2	SuperTyp	Wroblewskiego 25	51-867	Rower 5	1

Rysunek 4 Stacja po wstawieniu roweru

- Usunięcie roweru z bazy danych:

```
USE [RowerMiejski]
GO
/***** Object:  StoredProcedure [dbo].[usun_rower]    Script Date: 03.12.2022 20:33:01 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[usun_rower] @rower int
AS
BEGIN
    DELETE FROM Wypożyczenie WHERE Rower_Id = @rower
    DELETE FROM Rower
    WHERE Id = @rower;
    UPDATE Stacja
    SET Rowery = Rowery - 1
    WHERE Id = @rower
END
```

Rysunek 5 Kod procedury usuwania roweru

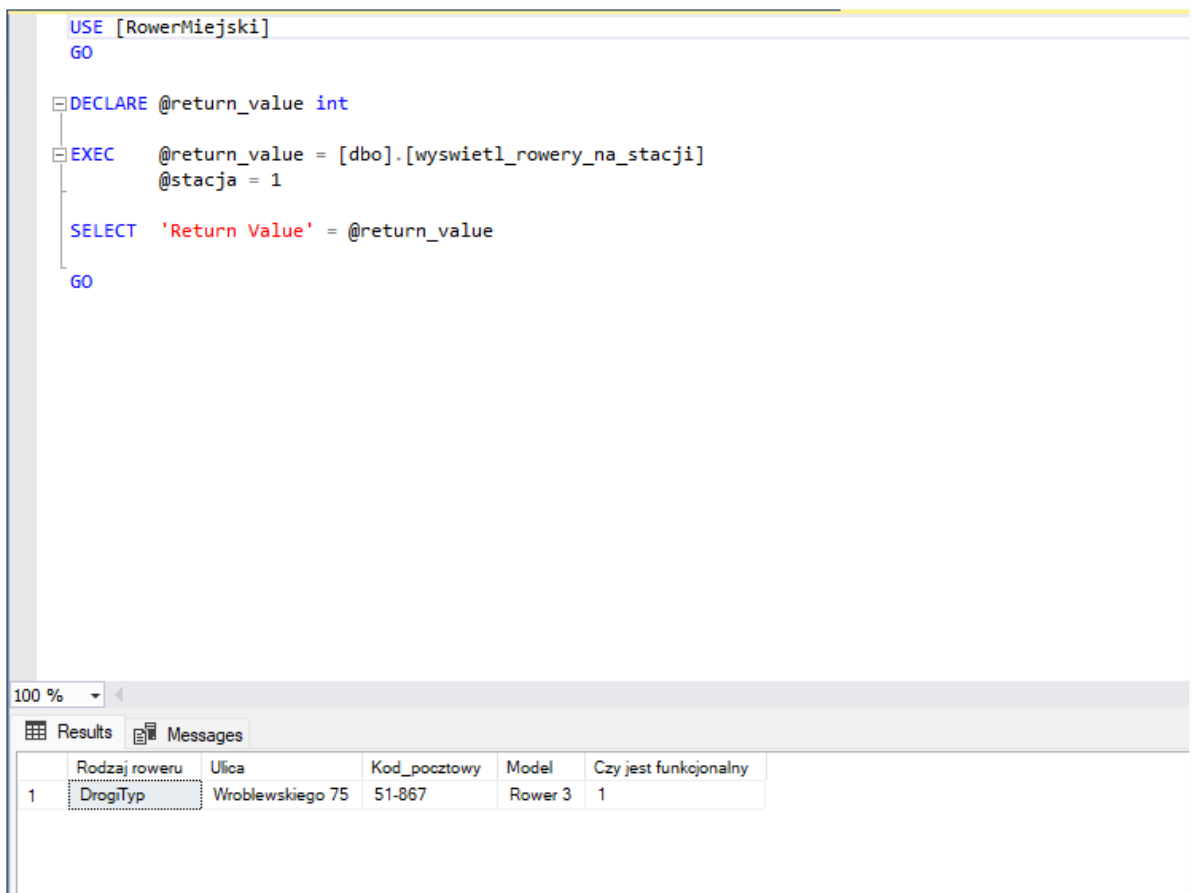
```
USE [RowerMiejski]
GO
DECLARE @return_value int
EXEC    @return_value = [dbo].[wyswietl_rowery_na_stacji]
        @stacja = 1
SELECT  'Return Value' = @return_value
GO
```

100 %

Results Messages

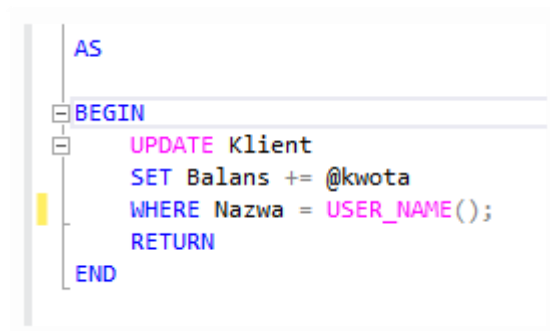
	Rodzaj roweru	Ulica	Kod_pocztowy	Model	Czy jest funkcjonalny
1	DrogTyp	Wroblewskiego 75	51-867	Rower 3	1
2	DrogTyp	Wroblewskiego 75	51-867	Rower 4	1

Rysunek 6 Stacja przed usunięciem roweru



Rysunek 7 Stacja po usunięciu

- Wyświetlenie swoich danych przez klienta i doładowanie konta



Rysunek 8 Doładowanie konta

```
USE [RowerMiejski]
GO

EXEC [dbo].[zaktualizuj_bilans]
    @kwota = 50
GO
```

Rysunek 9 Zwiększamy saldo klienta o 50

```
USE [RowerMiejski]
GO

EXEC [dbo].[wyswietl_dane]
```

00 %

Results Messages

	Nazwa	Imie	Nazwisko	Telefon	Email	Data_urodzenia	Balans
1	Maciej123	Maciej	Demucha	511867807	NULL	NULL	184,8

Rysunek 10 Klient wyświetla dane przed doładowaniem

The screenshot shows the SQL Server Enterprise Manager interface. In the top pane, a query is entered: `USE [RowerMiejski]` followed by `GO`, and then `EXEC [dbo].[wyswietl_dane]` followed by `GO`. The bottom pane displays the results of the query execution. It includes a 'Results' tab with a table of data and a 'Messages' tab. The table has columns: Nazwa, Imie, Nazwisko, Telefon, Email, Data_urodzenia, and Balans. The first row of data shows 'Maciej123' as the name, 'Maciej' as the first name, 'Demucha' as the last name, '511867807' as the phone number, and '234,8' as the balance. The 'Email' and 'Data_urodzenia' columns are highlighted in yellow and contain 'NULL'.

```
USE [RowerMiejski]
GO

EXEC [dbo].[wyswietl_dane]
GO
```

00 %

Results Messages

	Nazwa	Imie	Nazwisko	Telefon	Email	Data_urodzenia	Balans
1	Maciej123	Maciej	Demucha	511867807	NULL	NULL	234,8

Rysunek 11 Klient wyświetla dane po doładowaniu

- Wyświetlenie przez klienta listy stacji, rowerów znajdujących się na danej stacji oraz wypożyczenie i zwrot wybranego roweru

The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows a SQL query script with the following content:

```
USE [RowerMiejski]
GO

EXEC [dbo].[wyswietl_stacje]
GO
```

The bottom pane shows the results of the query execution. The 'Results' tab is active, displaying a table with four columns: 'Ulica', 'Kod_pocztowy', 'Miejsca', and 'Rowery'. The table contains two rows of data:

Ulica	Kod_pocztowy	Miejsca	Rowery
Wroblewskiego 75	51-867	50	3
Wroblewskiego 25	51-867	40	1

Rysunek 12 Wyświetlenie listy stacji

```

USE [RowerMiejski]
GO

EXEC [dbo].[wyswietl_rowery_na_stacji]
    @stacja = 1
GO

```

Rodzaj roweru	Ulica	Kod_pocztowy	Model	Czy jest funkcjonalny
DrogiTyp	Wroblewskiego 75	51-867	Rower 1	1
DrogiTyp	Wroblewskiego 75	51-867	Rower 3	1
SuperTyp	Wroblewskiego 75	51-867	Rower 4	1

Rysunek 13 Wyświetlenie rowerów na stacji

```

BEGIN
IF (SELECT COUNT(klient_view_wypozyczenie.[Koniec wypożyczenia]) from klient_view_wypozyczenie
WHERE klient_view_wypozyczenie.[Koniec wypożyczenia] is NULL) > 1
BEGIN
SELECT 'Nie można wypożyczyć więcej niż jeden rower' AS Komunikat
END
ELSE IF (SELECT Czy_funkcjonalny FROM Rower WHERE Id = @rower) = 1
BEGIN
INSERT INTO Wypożyczenie(Klient_Id, Rower_Id, Start_wypożyczenia)
VALUES((SELECT ID FROM Klient WHERE Nazwa = USER_NAME()), @rower, GETDATE());
UPDATE Stacja
SET Rowery = Rowery - 1
WHERE Id = (SELECT Stacja_Id FROM Rower
WHERE Rower.Id = @rower)
UPDATE Rower
SET Stacja_Id = NULL
WHERE Id = @rower
END
END

```

Rysunek 14 Kod procedury wypożyczenia

```

EXEC @return_value = [dbo].[wypożycz_rower]
    @rower = 2

```

Rysunek 15 Wypożyczamy rower o nazwie modelu 'Rower 1'

```
USE [RowerMiejski]
GO

EXEC [dbo].[wyswietl_rowery_na_stacji]
    @stacja = 1
GO
```

100 %

Results Messages

	Rodzaj roweru	Ulica	Kod_pocztowy	Model	Czy jest funkcjonalny
1	DrogiTyp	Wroblewskiego 75	51-867	Rower 3	1
2	SuperTyp	Wroblewskiego 75	51-867	Rower 4	1

Rysunek 16 Ze stacji z której wypożyczyliśmy rower znika z listy

```
EXEC @return_value = [dbo].[zwroc_rower]
    @rower = 2,
    @stacja = 2
```

Rysunek 17 Zwracamy rower na innej stacji

```

USE [RowerMiejski]
GO
/***** Object:  StoredProcedure [dbo].[zwroc_rower]    Script Date: 03.12.2022 20:29:34 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[zwroc_rower] @rower int, @stacja int
AS
BEGIN
IF((SELECT Rowery FROM Stacja WHERE Id = @stacja) = (SELECT Miejsca FROM Stacja WHERE Id = @stacja))
SELECT 'Nie ma wolnych miejsc' AS Komunikat
ELSE
    UPDATE Wypożyczenie
    SET Koniec_wypożyczenia = GETDATE()
    WHERE Rower_Id = @rower and (SELECT Nazwa FROM klient_view_customer) = USER_NAME() and Koniec_wypożyczenia is NULL;
    UPDATE Stacja
    SET Rowery = Rowery + 1
    WHERE Id = @stacja;
    UPDATE Rower
    SET Stacja_Id = @stacja
    WHERE Id = @rower
END

```

Rysunek 18 Kod procedury zwrotu roweru

```
USE [RowerMiejski]
GO

EXEC [dbo].[wyswietl_moja_historie_wypozycczen]
GO
```

100 %

Results Messages

	Id	Model	Nazwa	Start wypożyczenia	Koniec wypożyczenia
1	1	Rower 1	Maciej123	2022-12-03 16:18:59.800	2022-12-03 16:21:39.003
2	2	Rower 1	Maciej123	2022-12-03 18:52:31.907	2022-12-03 18:53:11.917
3	3	Rower 1	Maciej123	2022-12-03 20:22:21.693	2022-12-03 20:24:10.113

Rysunek 19 Wyświetlenie wypożyczeń

The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays a query script with the following content:

```
USE [RowerMiejski]
GO

EXEC [dbo].[wyswietl_oplate_dla_wypozyczenia]
@wypozyczenie = 3
GO
```

The bottom pane shows the execution results. The 'Results' tab is active, displaying a table with two columns: 'Do zapłaty' and 'Status opłaty'. The table contains one row of data.

	Do zapłaty	Status opłaty
1	2	1

Rysunek 20 Wyświetlenie opłaty za ostatnie wypożyczenie

```
USE [RowerMiejski]
GO

EXEC [dbo].[wyswietl_dane]
GO
```

100 %

Results Messages

	Nazwa	Imie	Nazwisko	Telefon	Email	Data_urodzenia	Balans
1	Maciej123	Maciej	Demucha	511867807	NULL	NULL	234,8

Rysunek 21 Dane klienta przed wypożyczeniem


```
USE [RowerMiejski]
GO

EXEC [dbo].[wyswietl_dane]
GO
```

100 %

Results Messages

	Nazwa	Imie	Nazwisko	Telefon	Email	Data_urodzenia	Balans
1	Maciej123	Maciej	Demucha	511867807	NULL	NULL	232,8

Rysunek 22 Dane klienta po wypożyczeniu

```
USE [RowerMiejski]
GO
```

EXEC [dbo].[wyswietl_rowery_na_stacji]
@stacja = 2

```
GO
```

100 %

Results Messages

	Rodzaj roweru	Ulica	Kod_pocztowy	Model	Czy jest funkcjonalny
1	DrogiTyp	Wroblewskiego 25	51-867	Rower 1	1
2	SuperTyp	Wroblewskiego 25	51-867	Rower 2	1
3	SuperTyp	Wroblewskiego 25	51-867	Rower 5	1

Rysunek 23 'Rower 1' jest na liście stacji w której został oddany

- Zgłoszenie usterki przez klienta

```
USE [RowerMiejski]
GO

DECLARE @return_value int

EXEC     @return_value = [dbo].[zglos_usterke]
        @rower = 2

SELECT   'Return Value' = @return_value

GO
```

Rysunek 24 Klient zgłasza usterkę w 'Rowerze 1'

```
USE [RowerMiejski]
GO
/***** Object: StoredProcedure [dbo].[zglos_usterke]    Script Date: 03.12.2022 19:43:55 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[zglos_usterke] @rower int
AS
BEGIN
    IF(SELECT COUNT(Rower_Id) FROM Usterka WHERE Rower_Id = @rower) > 0
        SELECT 'Ten rower ma juz zgloszona usterke' AS Komunikat
    ELSE
        INSERT INTO Usterka(Rower_Id, Zweryfikowana)
        VALUES(@rower, 0)
END
```

Rysunek 25 Kod procedury zgłaszania usterki

```
USE [RowerMiejski]
GO

EXEC [dbo].[wyswietl_rowery_na_stacji]
    @stacja = 1
GO
```

00 %

Results Messages

	Rodzaj roweru	Ulica	Kod_pocztowy	Model	Czy jest funkcjonalny
1	DrogiTyp	Wroblewskiego 75	51-867	Rower 1	0
2	DrogiTyp	Wroblewskiego 75	51-867	Rower 3	1
3	SuperTyp	Wroblewskiego 75	51-867	Rower 4	1

Rysunek 26 'Rower 1' jest oznaczony jako niefunkcjonalny

- Obsługa zgłoszonej usterki przez pracownika

```
USE [RowerMiejski]
GO

EXEC [dbo].[wyswietl_usterki]
GO
```

00 %

Results Messages

	Zweryfikowana	Model	Nazwa	Imie	Nazwisko
1	0	Rower 1	NULL	NULL	NULL

Rysunek 27 Zgłoszone usterki

```
USE [RowerMiejski]
GO
/***** Object: StoredProcedure [dbo].[weryfikuj_usterke]    Script Date: 03.12.2022 19:56:27 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[weryfikuj_usterke] @usterka int
AS
BEGIN
    UPDATE Usterka
    SET Zweryfikowana = 1, Pracownik_Id = (SELECT Id FROM Pracownik WHERE Nazwa = USER_NAME())
    WHERE Id = @usterka
END
```

Rysunek 28 Kod procedury weryfikacji usterki przez pracownika

```
USE [RowerMiejski]
GO

DECLARE @return_value int

EXEC     @return_value = [dbo].[weryfikuj_usterke]
        @usterka = 1

SELECT   'Return Value' = @return_value

GO
```

Rysunek 29 Pracownik weryfikuje usterkę

```
USE [RowerMiejski]
GO

EXEC     [dbo].[wyswietl_liste_usterek_dla_pracownika]
        @pracownik = 1

GO
```

00 %

Results Messages

	Zweryfikowana	Model	Nazwa	Imie	Nazwisko
1	1	Rower 1	Pracownik123	Jan	Kowal

Rysunek 30 Usterka zostaje przypisana do pracownika

```

USE [RowerMiejski]
GO

DECLARE @return_value int

EXEC     @return_value = [dbo].[usun_usterke]
        @usterka = 1

SELECT   'Return Value' = @return_value

GO

```

Rysunek 31 Po rzeczywistym naprawieniu roweru pracownik usuwa usterkę z bazy

```

USE [RowerMiejski]
GO

DECLARE @return_value int

EXEC     @return_value = [dbo].[wyswietl_rowery_na_stacji]
        @stacja = 1

SELECT   'Return Value' = @return_value

GO

```

Results				
Rodzaj roweru	Ulica	Kod_pocztowy	Model	Czy jest funkcjonalny
DrogiTyp	Wroblewskiego 75	51-867	Rower 1	1
DrogiTyp	Wroblewskiego 75	51-867	Rower 3	1
SuperTyp	Wroblewskiego 75	51-867	Rower 4	1

Rysunek 32 'Rower 1' jest znów oznaczony jako funkcjonalny

Skrypt:

```
CREATE DATABASE [RowerMiejski]
```

```
USE [RowerMiejski]
```

```
GO
```

```
CREATE USER [Pracownik123] FOR LOGIN [Pracownik123] WITH DEFAULT_SCHEMA=[dbo]
```

```
GO
```

```
CREATE USER [Michal123] FOR LOGIN [Michal123] WITH DEFAULT_SCHEMA=[dbo]
```

```
GO
```

```
CREATE USER [Maciej123] FOR LOGIN [Maciej123] WITH DEFAULT_SCHEMA=[dbo]
```

```
GO
```

```
CREATE ROLE [Pracownik]
```

```
GO
```

```
CREATE ROLE [Klient]
```

```
GO
```

```
ALTER ROLE [Pracownik] ADD MEMBER [Pracownik123]
```

```
GO
```

```
ALTER ROLE [Klient] ADD MEMBER [Michal123]
```

```
GO
```

```
ALTER ROLE [Klient] ADD MEMBER [Maciej123]
```

```
GO
```



```
CREATE TABLE [dbo].[Klient](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [Nazwa] [nvarchar](128) NOT NULL,
    [Imie] [varchar](50) NOT NULL,
    [Nazwisko] [varchar](50) NOT NULL,
    [Telefon] [int] NULL,
    [Email] [varchar](50) NULL,
    [Data_urodzenia] [date] NULL,
    [Balans] [float] NULL,

    CONSTRAINT [PK_Klient] PRIMARY KEY CLUSTERED
    (
        [Id] ASC
    )
)
```

```
CREATE VIEW [dbo].[klient_view_customer]
AS
SELECT    Nazwa, Imie, Nazwisko, Telefon, Email, Data_urodzenia, Balans
FROM      dbo.Klient
WHERE     (Nazwa = USER_NAME())
GO
```

```

CREATE TABLE [dbo].[Wypożyczenie](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [Klient_Id] [int] NOT NULL,
    [Rower_Id] [int] NOT NULL,
    [Start_wypożyczenia] [datetime] NOT NULL,
    [Koniec_wypożyczenia] [datetime] NULL,
    CONSTRAINT [PK_Wypożyczenie] PRIMARY KEY CLUSTERED
(
    [Id] ASC
)
)

```

```

CREATE TABLE [dbo].[Rower](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [Typ_roweru_Id] [int] NOT NULL,
    [Stacja_Id] [int] NULL,
    [Model] [varchar](50) NOT NULL,
    [Czy_funkcjonalny] [bit] NOT NULL,
    CONSTRAINT [PK_Rower] PRIMARY KEY CLUSTERED
(
    [Id] ASC
)
)

```

```

CREATE VIEW [dbo].[klient_view_wypożyczenie]
AS
SELECT    dbo.Wypożyczenie.Id, dbo.Rower.Model, dbo.Klient.Nazwa,
dbo.Wypożyczenie.Start_wypożyczenia AS 'Start wypożyczenia',
dbo.Wypożyczenie.Koniec_wypożyczenia AS 'Koniec wypożyczenia'
FROM      dbo.Wypożyczenie INNER JOIN
          dbo.Klient ON dbo.Wypożyczenie.Klient_Id = dbo.Klient.Id INNER JOIN
          dbo.Rower ON dbo.Rower.Id = dbo.Wypożyczenie.Rower_Id
WHERE     (dbo.Klient.Nazwa = USER_NAME())

```

GO

```
CREATE TABLE [dbo].[Pracownik](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [Nazwa] [varchar](50) NOT NULL,
    [Imie] [varchar](50) NOT NULL,
    [Nazwisko] [varchar](50) NOT NULL,
    CONSTRAINT [PK_Pracownik] PRIMARY KEY CLUSTERED
(
    [Id] ASC
)
)
```

```
CREATE VIEW [dbo].[pracownik_view_employee]
```

```
AS
```

```
SELECT    Id, Nazwa, Imie, Nazwisko
```

```
FROM      dbo.Pracownik
```

```
WHERE     (Nazwa = USER_NAME())
```

GO

```
CREATE TABLE [dbo].[Adres](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [Ulica] [varchar](50) NOT NULL,
    [Kod_pocztowy] [varchar](50) NOT NULL,
    CONSTRAINT [PK_Adres] PRIMARY KEY CLUSTERED
(
    [Id] ASC
)
)
```

```

CREATE TABLE [dbo].[Opłata](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [Wypozyczenie_Id] [int] NOT NULL,
    [do_zaplaty] [float] NOT NULL,
    [status_zaplaty] [bit] NOT NULL,
    CONSTRAINT [PK_Payment] PRIMARY KEY CLUSTERED
(
    [Id] ASC
)
)

CREATE TABLE [dbo].[Stacja](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [Adres_Id] [int] NOT NULL,
    [Miejsca] [int] NOT NULL,
    [Rowery] [int] NOT NULL,
    CONSTRAINT [PK_Stacja] PRIMARY KEY CLUSTERED
(
    [Id] ASC
)
)

CREATE TABLE [dbo].[Typ_roweru](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [Cena_minuta] [float] NOT NULL,
    [Nazwa] [varchar](50) NOT NULL,
    CONSTRAINT [PK_Typ_roweru] PRIMARY KEY CLUSTERED
(
    [Id] ASC
)
)

```

```

CREATE TABLE [dbo].[Usterka](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [Zweryfikowana] [bit] NOT NULL,
    [Rower_Id] [int] NOT NULL,
    [Pracownik_Id] [int] NULL,
    CONSTRAINT [PK_Usterka] PRIMARY KEY CLUSTERED
(
    [Id] ASC
)
)

ALTER TABLE [dbo].[Opłata] WITH CHECK ADD CONSTRAINT [FK_Payment_Wypożyczenie] FOREIGN
KEY([Wypożyczenie_Id])
REFERENCES [dbo].[Wypożyczenie] ([Id])
ON DELETE CASCADE
GO

ALTER TABLE [dbo].[Opłata] CHECK CONSTRAINT [FK_Payment_Wypożyczenie]
GO

ALTER TABLE [dbo].[Rower] WITH CHECK ADD CONSTRAINT [FK_Rower_Stacja] FOREIGN
KEY([Stacja_Id])
REFERENCES [dbo].[Stacja] ([Id])
GO

ALTER TABLE [dbo].[Rower] CHECK CONSTRAINT [FK_Rower_Stacja]
GO

ALTER TABLE [dbo].[Rower] WITH CHECK ADD CONSTRAINT [FK_Rower_Typ_roweru] FOREIGN
KEY([Typ_roweru_Id])
REFERENCES [dbo].[Typ_roweru] ([Id])
GO

ALTER TABLE [dbo].[Rower] CHECK CONSTRAINT [FK_Rower_Typ_roweru]
GO

ALTER TABLE [dbo].[Stacja] WITH CHECK ADD CONSTRAINT [FK_Stacja_Adres] FOREIGN
KEY([Adres_Id])
REFERENCES [dbo].[Adres] ([Id])

```

GO

ALTER TABLE [dbo].[Stacja] CHECK CONSTRAINT [FK_Stacja_Adres]

GO

ALTER TABLE [dbo].[Usterka] WITH CHECK ADD CONSTRAINT [FK_Usterka_Pracownik] FOREIGN KEY([Pracownik_Id])

REFERENCES [dbo].[Pracownik] ([Id])

GO

ALTER TABLE [dbo].[Usterka] CHECK CONSTRAINT [FK_Usterka_Pracownik]

GO

ALTER TABLE [dbo].[Usterka] WITH CHECK ADD CONSTRAINT [FK_Usterka_Rower] FOREIGN KEY([Rower_Id])

REFERENCES [dbo].[Rower] ([Id])

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[Usterka] CHECK CONSTRAINT [FK_Usterka_Rower]

GO

ALTER TABLE [dbo].[Wypożyczenie] WITH CHECK ADD CONSTRAINT [FK_Wypożyczenie_Klient] FOREIGN KEY([Klient_Id])

REFERENCES [dbo].[Klient] ([Id])

GO

ALTER TABLE [dbo].[Wypożyczenie] CHECK CONSTRAINT [FK_Wypożyczenie_Klient]

GO

ALTER TABLE [dbo].[Wypożyczenie] WITH CHECK ADD CONSTRAINT [FK_Wypożyczenie_Rower] FOREIGN KEY([Rower_Id])

REFERENCES [dbo].[Rower] ([Id])

GO

ALTER TABLE [dbo].[Wypożyczenie] CHECK CONSTRAINT [FK_Wypożyczenie_Rower]

GO

```
CREATE PROCEDURE [dbo].[dodaj_rower] @Typ_roweru int, @Stacja int, @Model varchar(50)
```

```
AS
```

```
BEGIN
```

```
    INSERT INTO Rower(Typ_roweru_Id, Stacja_Id, Model, Czy_funkcjonalny)
```

```
    VALUES(@Typ_roweru, @Stacja, @Model, 1)
```

```
END
```

```
GO
```

```
CREATE PROCEDURE [dbo].[usun_klienta] @klient int
```

```
AS
```

```
BEGIN
```

```
    DELETE FROM Klient
```

```
    WHERE Id = @klient
```

```
END
```

```
GO
```

```
CREATE PROCEDURE [dbo].[usun_rower] @rower int
```

```
AS
```

```
BEGIN
```

```
    DELETE FROM Wypożyczenie WHERE Rower_Id = @rower
```

```
    DELETE FROM Rower
```

```
    WHERE Id = @rower;
```

```
    UPDATE Stacja
```

```
    SET Rowery = Rowery - 1
```

```
    WHERE Id = @rower
```

```
END
```

```
GO
```

```
CREATE PROCEDURE [dbo].[usun_usterke] @usterka int
AS
BEGIN
    UPDATE Rower
    SET Czy_funkcjonalny = 1
    WHERE Id = (SELECT Rower_Id FROM Usterka WHERE Id = @usterka)
    DELETE FROM Usterka WHERE Id = @usterka;
END
GO

CREATE PROCEDURE [dbo].[weryfikuj_usterke] @usterka int
AS
BEGIN
    UPDATE Usterka
    SET Zweryfikowana = 1, Pracownik_Id = (SELECT Id FROM Pracownik WHERE Nazwa =
    USER_NAME())
    WHERE Id = @usterka
END
GO
```



```

CREATE PROCEDURE [dbo].[wypożycz_rower] @rower int
AS
BEGIN
    IF (SELECT COUNT(klient_view_wypożyczenie.[Koniec wypożyczenia]) from
    klient_view_wypożyczenie WHERE klient_view_wypożyczenie.[Koniec wypożyczenia] is NULL) > 1
        BEGIN
            SELECT 'Nie można wypożyczyć więcej niż jeden rower' AS Komunikat
        END
    ELSE IF (SELECT Czy_funkcjonalny FROM Rower WHERE Id = @rower) = 1
        BEGIN
            INSERT INTO Wypożyczenie(Klient_Id, Rower_Id, Start_wypożyczenia)
            VALUES((SELECT ID FROM Klient WHERE Nazwa = USER_NAME()), @rower,
GETDATE());

            UPDATE Stacja
            SET Rowery = Rowery - 1
            WHERE Id = (SELECT Stacja_Id FROM Rower
                WHERE Rower.Id = @rower)

            UPDATE Rower
            SET Stacja_Id = NULL
            WHERE Id = @rower
        END
    END
GO

CREATE PROCEDURE [dbo].[wyswietl_dane]
AS
BEGIN
    SELECT * FROM klient_view_customer
END
GO

```

```
CREATE PROCEDURE [dbo].[wyswietl_liste_usterek_dla_pracownika] @pracownik int
```

```
AS
```

```
BEGIN
```

```
    SELECT Zweryfikowana, Rower.Model, Pracownik.Nazwa, Pracownik.Imie,  
    Pracownik.Nazwisko FROM Usterka
```

```
    JOIN Pracownik ON Pracownik.Id = Usterka.Pracownik_Id
```

```
    JOIN Rower ON Rower.Id = Rower_Id
```

```
    WHERE Pracownik_Id = @pracownik
```

```
END
```

```
GO
```

```
CREATE PROCEDURE [dbo].[wyswietl_moja_historie_wypozycczen]
```

```
AS
```

```
BEGIN
```

```
    SELECT * FROM klient_view_wypozycczenie
```

```
END
```

```
GO
```

```
CREATE PROCEDURE [dbo].[wyswietl_oplate_dla_wypozycczenia] @wypozycczenie int
```

```
AS
```

```
BEGIN
```

```
    SELECT do_zaplaty AS 'Do zapłaty', status_zaplaty AS 'Status opłaty' FROM Opłata
```

```
    JOIN klient_view_wypozycczenie ON klient_view_wypozycczenie.Id = Opłata.Wypozycczenie_Id
```

```
    WHERE klient_view_wypozycczenie.Id = @wypozycczenie
```

```
END
```

```
GO
```

```

CREATE PROCEDURE [dbo].[wyswietl_rowery_na_stacji] @stacja int
AS
BEGIN
    SELECT Typ_roweru.Nazwa AS 'Rodzaj roweru', Model, Czy_funkcjonalny AS 'Czy jest
funkcjonalny' FROM Rower
    JOIN Typ_roweru ON Typ_roweru.Id = Typ_roweru_Id
    JOIN Stacja ON Stacja.Id = Stacja_Id
    WHERE Rower.Stacja_Id = @stacja
END;
GO

CREATE PROCEDURE [dbo].[wyswietl_stacje]
AS
BEGIN
    SELECT Adres.Ulica, Adres.Kod_pocztowy, Miejsca, Rowery FROM Stacja
    JOIN Adres ON Adres.Id = Stacja.Adres_Id;
END
GO

CREATE PROCEDURE [dbo].[wyswietl_usterki]
AS
BEGIN
    SELECT Zweryfikowana, Rower.Model, Pracownik.Nazwa, Pracownik.Imie,
Pracownik.Nazwisko FROM Usterka
    LEFT JOIN Pracownik ON Pracownik.Id = Usterka.Pracownik_Id
    JOIN Rower ON Rower.Id = Rower_Id
END
GO

```

```
CREATE PROCEDURE [dbo].[zaktualizuj_bilans] @kwota float
```

```
AS
```

```
BEGIN
```

```
    UPDATE Klient
```

```
    SET Balans += @kwota
```

```
    WHERE Nazwa = USER_NAME();
```

```
    RETURN
```

```
END
```

```
GO
```

```
CREATE PROCEDURE [dbo].[zglos_usterke] @rower int
```

```
AS
```

```
BEGIN
```

```
    IF(SELECT COUNT(Rower_Id) FROM Usterka WHERE Rower_Id = @rower) > 0
```

```
        SELECT 'Ten rower ma juz zgloszona usterke' AS Komunikat
```

```
    ELSE
```

```
        INSERT INTO Usterka(Rower_Id, Zweryfikowana)
```

```
        VALUES(@rower, 0)
```

```
END
```

```
GO
```

```
CREATE PROCEDURE [dbo].[zwroc_rower] @rower int, @stacja int
AS
BEGIN
IF((SELECT Rowery FROM Stacja WHERE Id = @stacja) = (SELECT Miejsca FROM Stacja WHERE Id =
@stacja))
SELECT 'Nie ma wolnych miejsc' AS Komunikat
ELSE
    UPDATE Wypożyczenie
    SET Koniec_wypozyczenia = GETDATE()
    WHERE Rower_Id = @rower and (SELECT Nazwa FROM klient_view_customer) =
USER_NAME() and Koniec_wypozyczenia is NULL;
    UPDATE Stacja
    SET Rowery = Rowery + 1
    WHERE Id = @stacja;
    UPDATE Rower
    SET Stacja_Id = @stacja
    WHERE Id = @rower
END
GO
```

```

CREATE TRIGGER [dbo].[payment_when_update]
    ON [dbo].[Opłata]
    AFTER UPDATE
AS
BEGIN
    UPDATE Klient
    SET Balans = Balans - (SELECT TOP 1 do_zapłaty FROM Opłata ORDER BY Id DESC)
    WHERE Klient.Id = (SELECT Id FROM klient_view_customer WHERE Nazwa = USER_NAME());

    IF (SELECT Balans FROM klient_view_customer WHERE Nazwa = USER_NAME()) < 0
        SELECT 'Bilans konta na minusie! Doladuj konto!' AS Komunikat
END
GO

ALTER TABLE [dbo].[Opłata] ENABLE TRIGGER [payment_when_update]
GO

CREATE TRIGGER [dbo].[usterka_when_insert]
    ON [dbo].[Usterka]
    AFTER INSERT
AS
BEGIN
    UPDATE Rower
    SET Czy_funkcjonalny = 0
    WHERE Id = (SELECT Rower_Id from Usterka WHERE Id = (SELECT TOP 1 Id FROM Usterka
ORDER BY Id DESC))

END
GO

ALTER TABLE [dbo].[Usterka] ENABLE TRIGGER [usterka_when_insert]
GO

```

```
CREATE TRIGGER [dbo].[wypozyczenie_when_delete]
    ON [dbo].[Wypożyczenie]
    FOR DELETE
AS
BEGIN
    UPDATE Opłata
    SET Wypozyczenie_Id = NULL
    WHERE Wypozyczenie_Id IN(SELECT deleted.id FROM deleted)

END
GO
ALTER TABLE [dbo].[Wypożyczenie] ENABLE TRIGGER [wypozyczenie_when_delete]
GO
CREATE TRIGGER [dbo].[wypozyczenie_when_insert]
    ON [dbo].[Wypożyczenie]
    AFTER INSERT
AS
BEGIN
    INSERT INTO Opłata VALUES((SELECT TOP 1 Id FROM klient_view_wypozyczenie ORDER BY Id
DESC), 0, 0);

END
GO
ALTER TABLE [dbo].[Wypożyczenie] ENABLE TRIGGER [wypozyczenie_when_insert]
GO
```

```
CREATE TRIGGER [dbo].[wypozyczenie_when_update]
ON [dbo].[Wypożyczenie]
AFTER UPDATE
AS
BEGIN
    UPDATE Opłata
    SET do_zapłaty =
        (SELECT DATEDIFF(MINUTE, Start_wypozyczenia, Koniec_wypozyczenia) *
        Typ_roweru.Cena_minuta FROM Wypożyczenie
        JOIN Rower ON Rower_Id = Rower.Id
        JOIN Typ_roweru ON Typ_roweru_Id = Typ_roweru.Id
        WHERE Wypożyczenie.Id = Wypozyczenie_Id), status_zapłaty = 1
END
GO
ALTER TABLE [dbo].[Wypożyczenie] ENABLE TRIGGER [wypozyczenie_when_update]
```


Projekt interfejsu użytkownika

Id	Ulica	Kod pocztowy	Liczba miejsc	Liczba rowerów
<div>Wyświetl rowery na stacji</div>				

Lista stacji

Ekran główny użytkownika

Wybór stacji, gdzie oddać wypożyczony rower

Historia wypożyczeń

Zmiana danych osobowych

Lista rowerów na stacji

Wyświetl klientów

Wyświetl stacje

Wyświetl usterki

Wyświetl typy rowerów i ceny

Konto

Widok główny pracownika

Nick	Imię	Nazwisko	Telefon	Email	Data urodzenia	Stan konta
------	------	----------	---------	-------	----------------	------------

Wypożyczenia klienta

Lista użytkowników

Nr usterki	Id roweru	Model	Stan weryfikacji
------------	-----------	-------	------------------

Zweryfikuj

Odrzuć

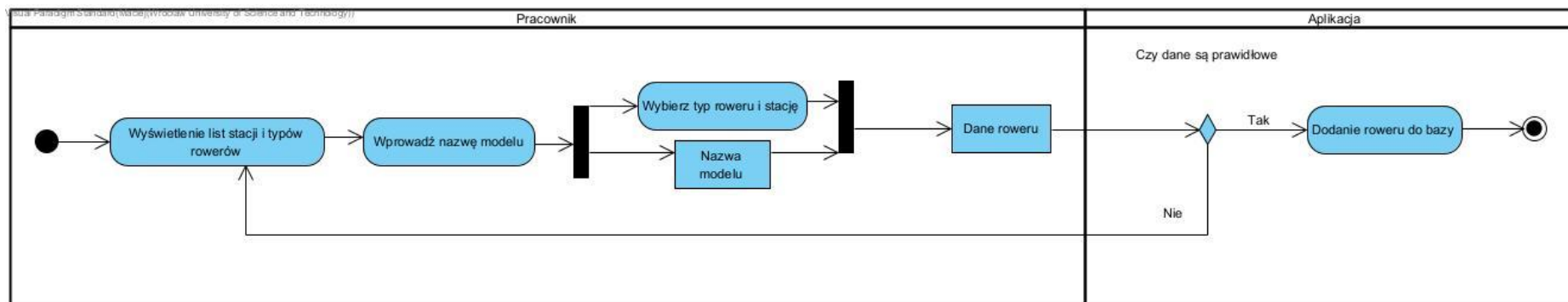
Lista usterek

Nazwa modelu	Cena
--------------	------

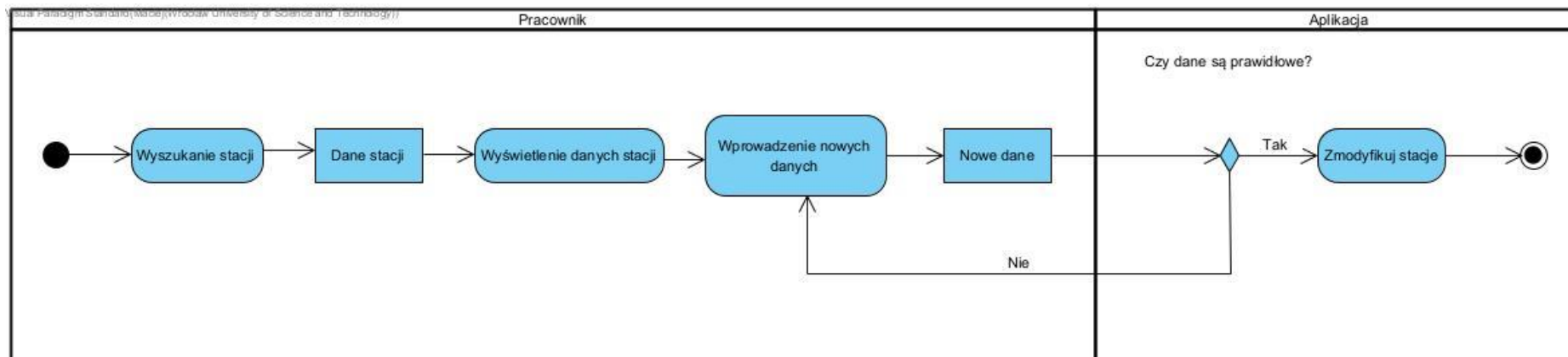
Lista rodzajów rowerów

Diagramy aktywności:

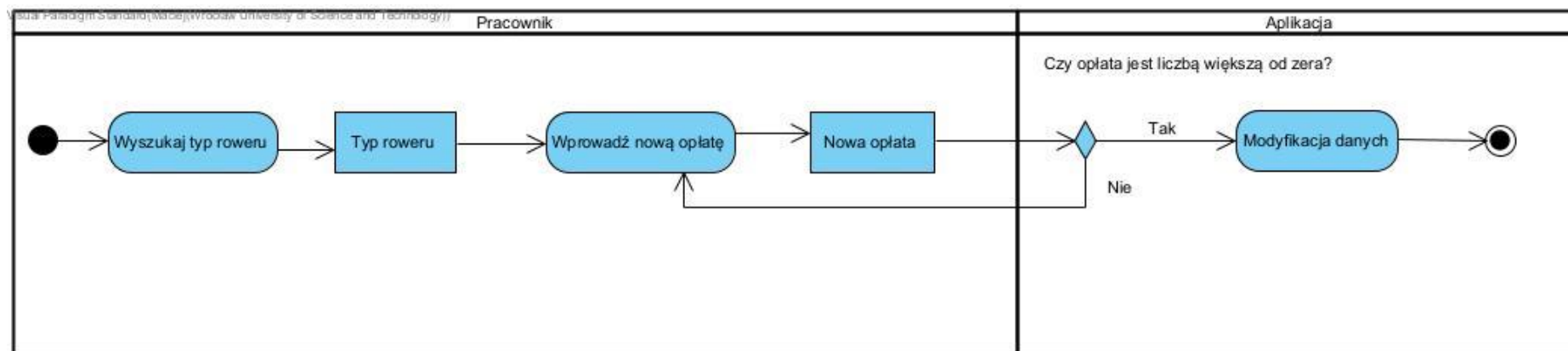
Dodanie nowego roweru do bazy



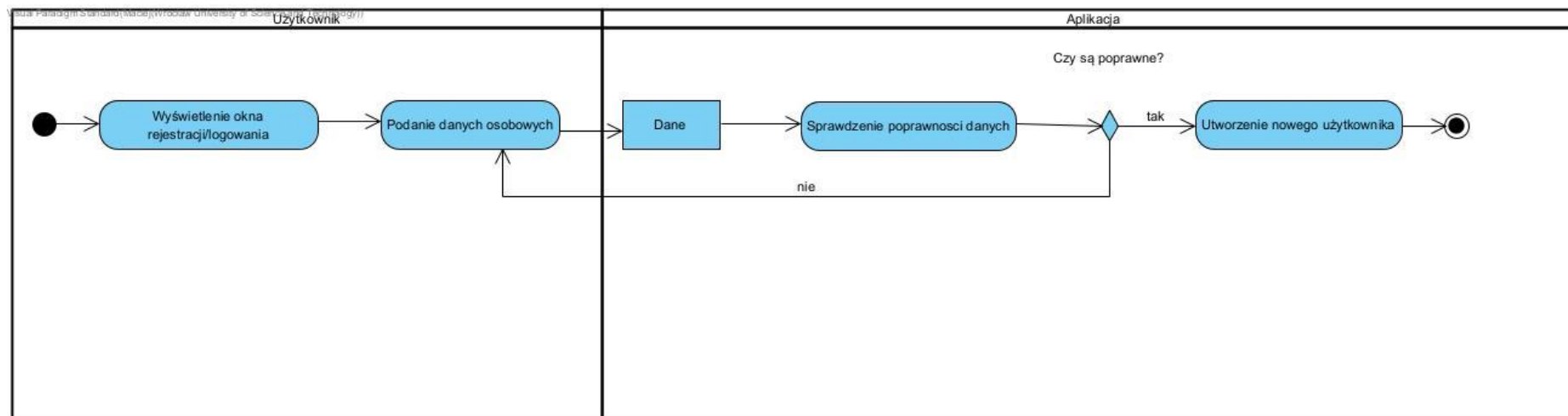
Modyfikacja danych stacji



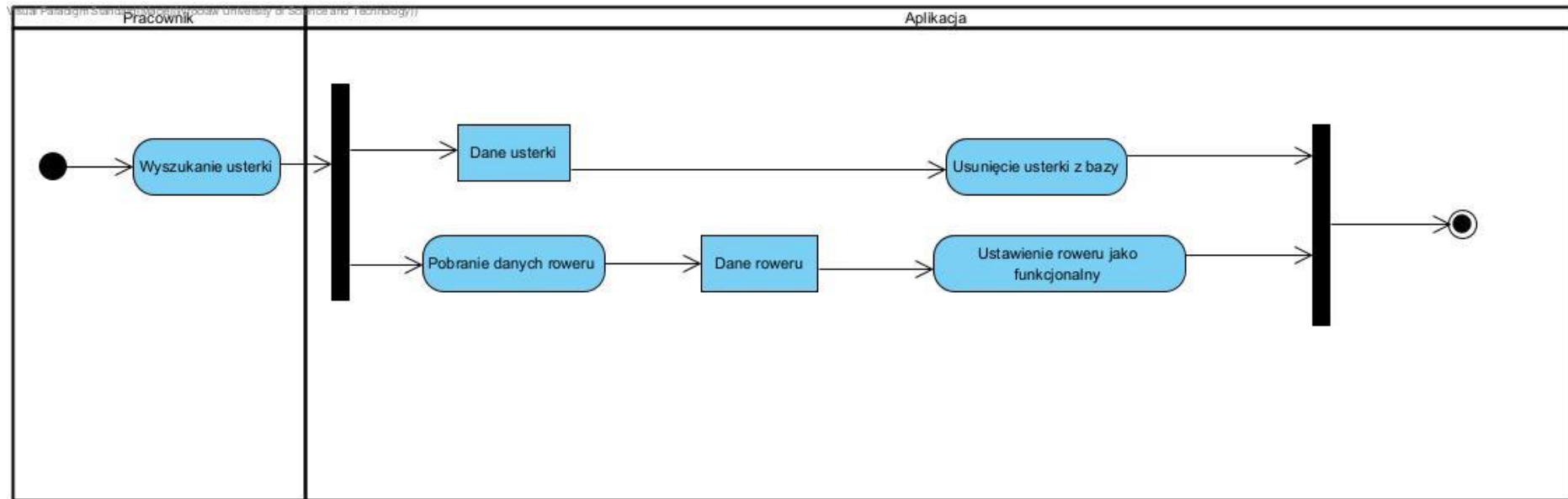
Modyfikacja opłaty



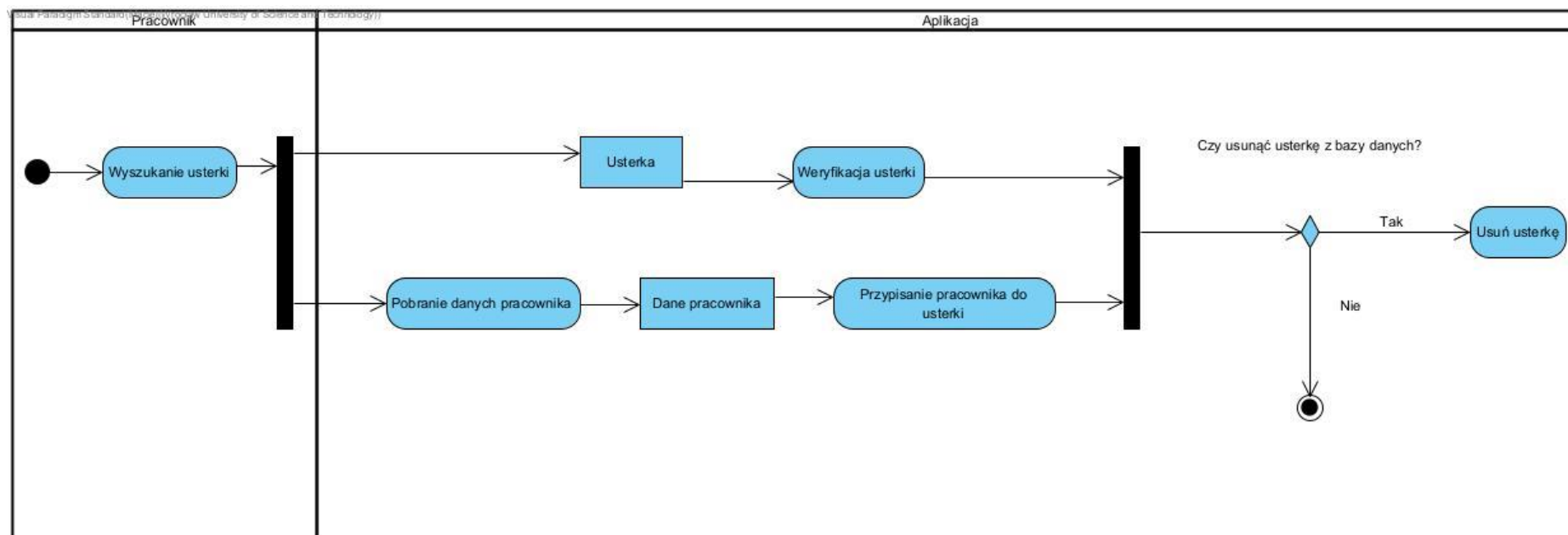
Rejestracja



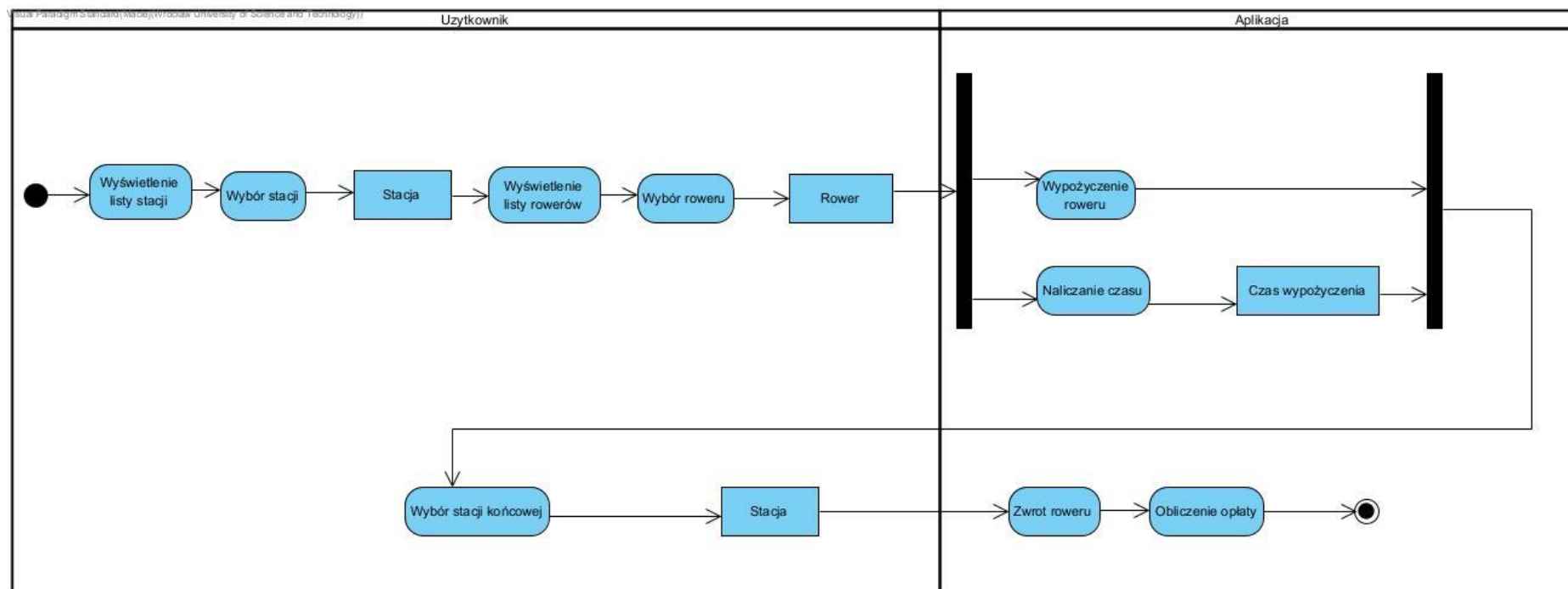
Usunięcie usterki



Weryfikacja usterki



Wypożyczenie roweru



Zgłaszanie usterki

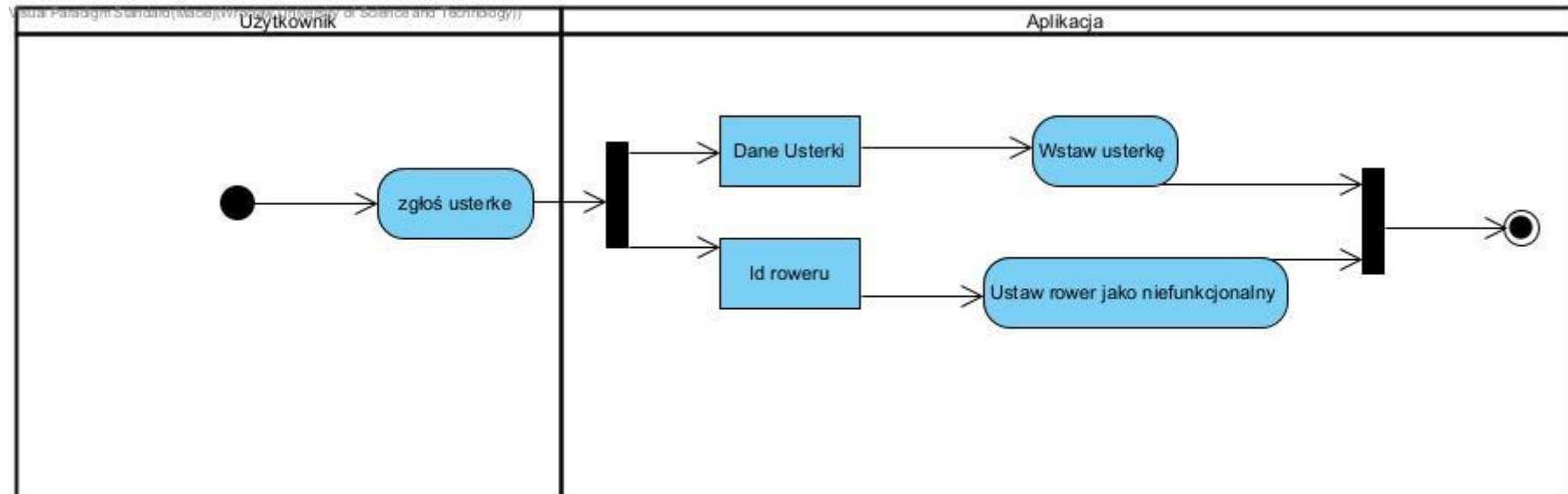
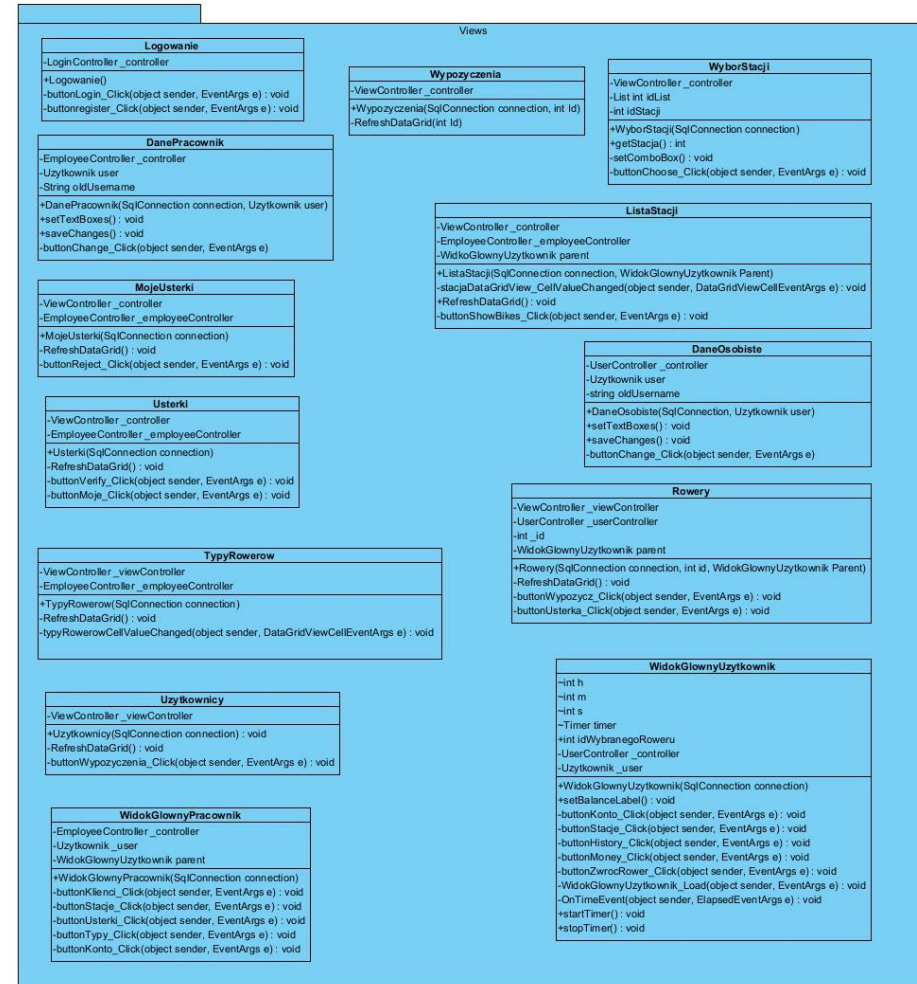
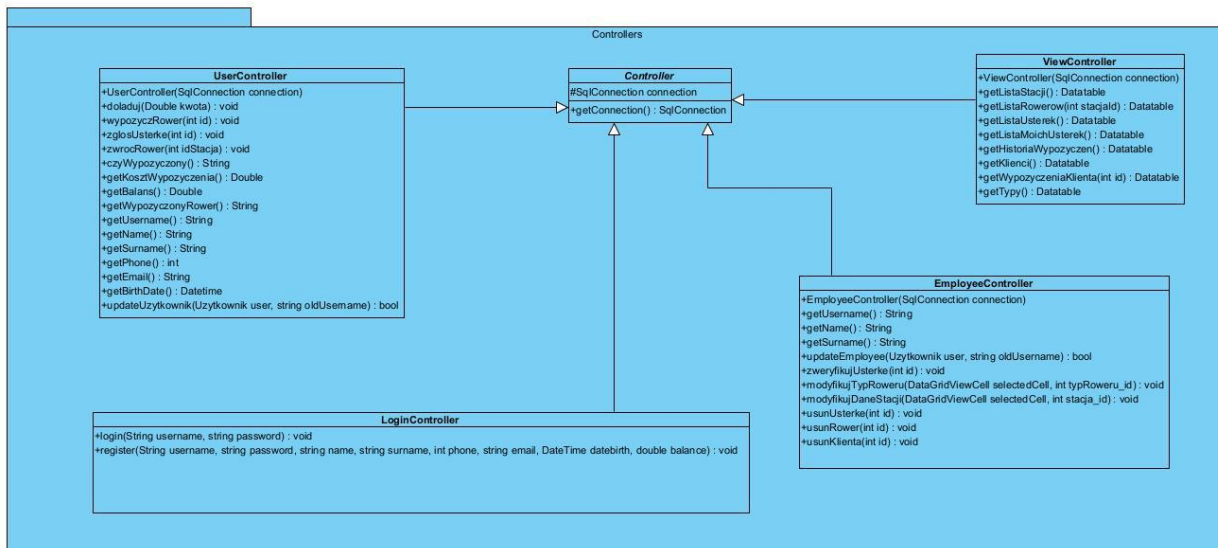
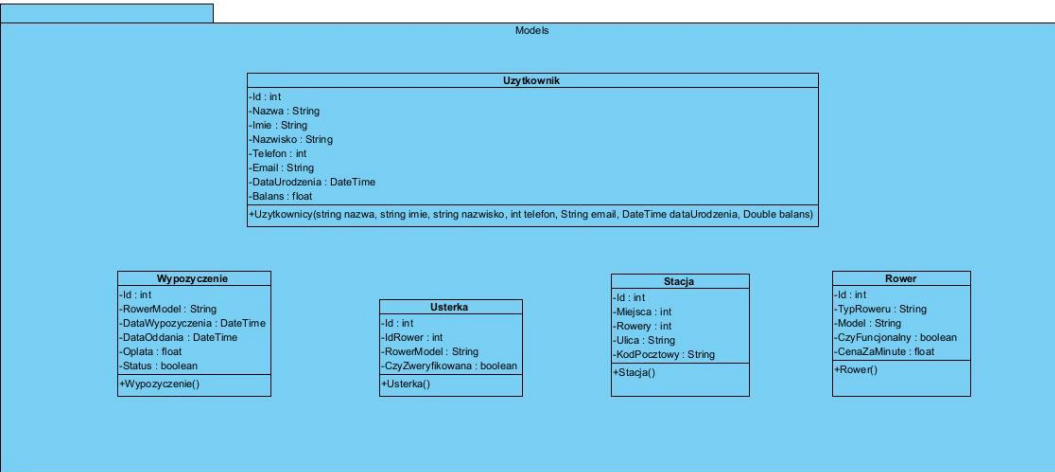


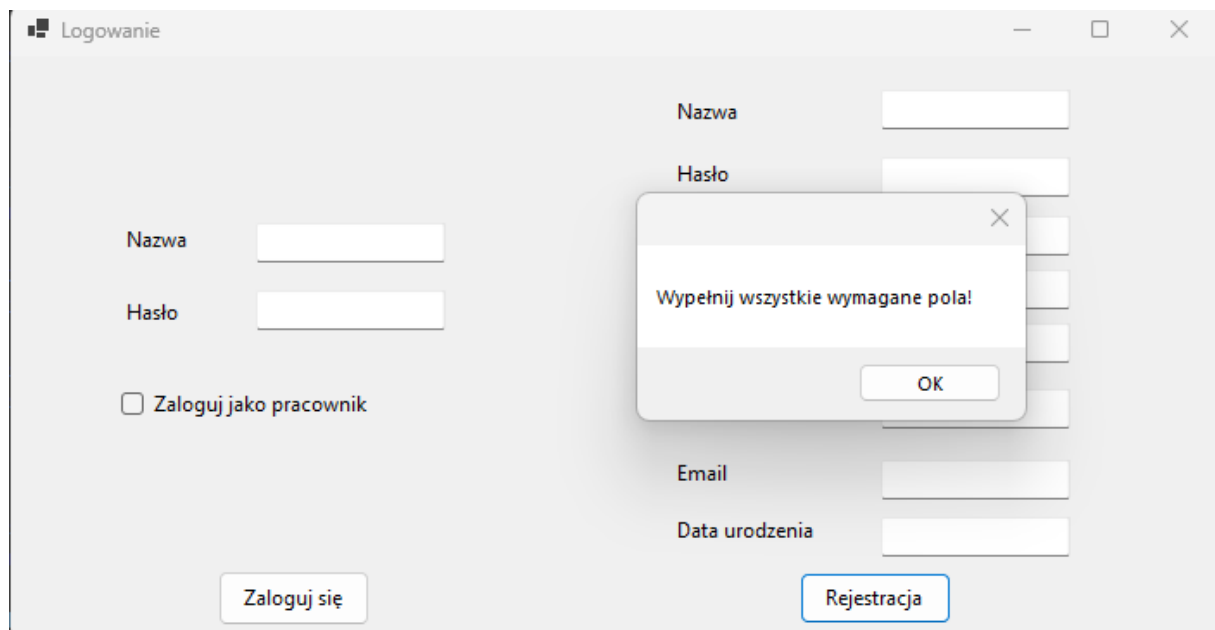
Diagram klas

Visual Paradigm Standard (Model) (Wrocław University of Science and Technology)

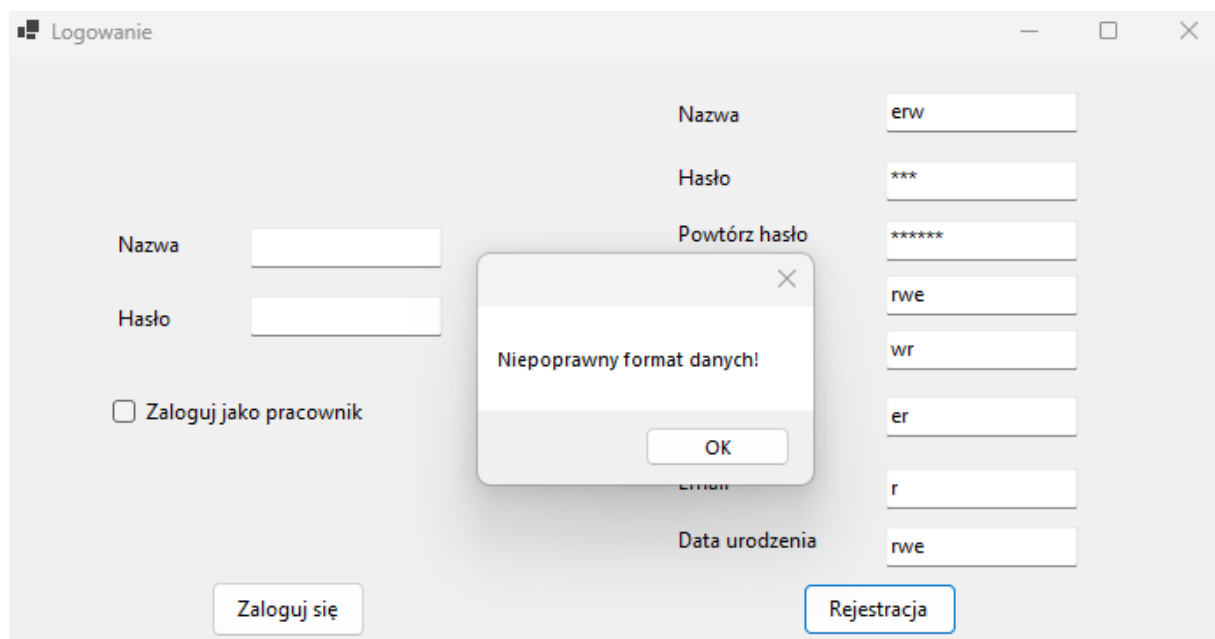


Testy:

- Rejestracja użytkownika:



The screenshot shows a window titled "Logowanie" (Login). On the left, there is a login section with fields for "Nazwa" (Name) and "Hasło" (Password), a checkbox labeled "Zaloguj jako pracownik" (Login as employee), and a "Zaloguj się" (Login) button. On the right, there is a registration section with fields for "Nazwa", "Hasło", "Email", and "Data urodzenia" (Date of birth), and a "Rejestracja" (Registration) button. A modal dialog box is centered over the registration fields, displaying the message "Wypełnij wszystkie wymagane pola!" (Fill in all required fields!) and an "OK" button.



The screenshot shows the same "Logowanie" window. The registration fields are now filled with data: "Nazwa" is "erw", "Hasło" is "***", "Powtórz hasło" (Repeat password) is "*****", "Email" is "rwe", and "Data urodzenia" is "wr". A modal dialog box is centered over the registration fields, displaying the message "Niepoprawny format danych!" (Incorrect data format!) and an "OK" button.

Program sprawdza, czy użytkownik wypełnił wszystkie pola do rejestracji oraz czy są one poprawne.

- Logowanie

The screenshot shows a login window titled "Logowanie". On the left, there are input fields for "Nazwa" (Name) and "Hasło" (Password), a checkbox labeled "Zaloguj jako pracownik" (Login as employee), and a "Zaloguj się" (Login) button. On the right, there are input fields for "Nazwa", "Hasło", "Powtórz hasło" (Repeat password), and "Data urodzenia" (Date of birth), along with a "Rejestracja" (Registration) button. A modal dialog box titled "Błąd logowania" (Login error) is displayed in the center, featuring a red circle with a white 'X' icon and the text "Nieprawidłowe dane!" (Invalid data!). The dialog has an "OK" button at the bottom.

W przypadku błędnych danych aplikacja wyświetla komunikat i prosi o ponowne wpisanie danych.

- Doładowanie konta

WidokGlownyUzytkownik

Stan konta: 200,2 zł

Wypożyczony rower: Brak

Cena za minutę: 0

Czas wypożyczenia: 00:00:00

Doładuj konto

Zwróć rower

Historia wypożyczeń

Wyświetl stacje

Konto

WidokGlownyUzytkownik

Stan konta: 240,2 zł

Wypożyczony rower: Brak

Cena za minutę: 0

Czas wypożyczenia: 00:00:00

Doładuj konto

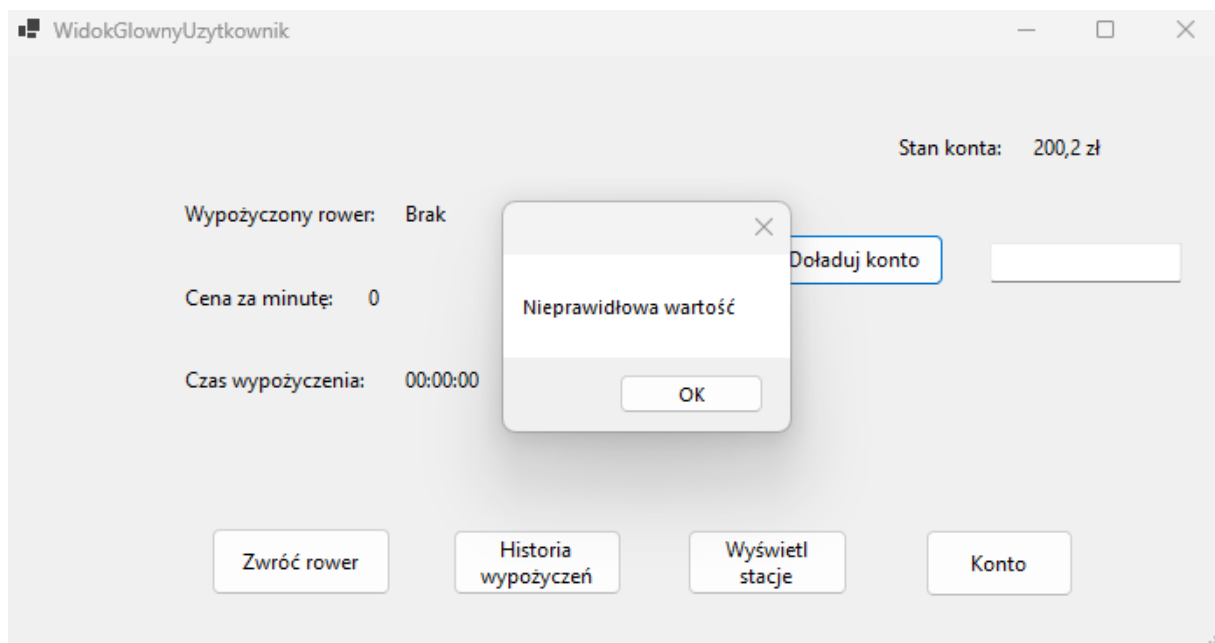
40

Zwróć rower

Historia wypożyczeń

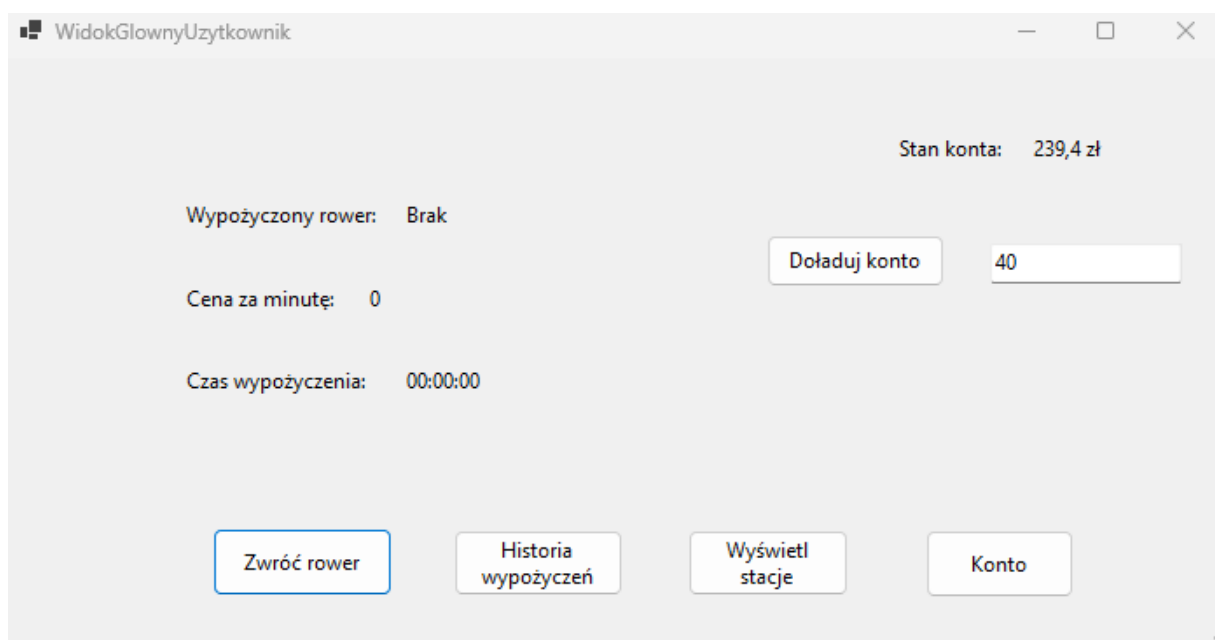
Wyświetl stacje

Konto



Aplikacja pozwala doładować konto użytkownika o zadaną kwotę. Sprawdza również czy została podana liczba zmiennoprzecinkowa.

- Wypożyczenie roweru



ListaStacji

	Id	Ulica	Kod pocztowy	Miejsca	Liczba Rowerów
▶	1	Śliczna 21	51-211	60	3
	2	Wroblewskiego...	51-867	40	2

Wyświetl rowery na stacji

Rowery

	Id	Rodzaj roweru	Model	Czy jest funkcjonalny
▶	5	SuperTyp	Rower 2	<input checked="" type="checkbox"/>
	11	SuperTyp	Rower 5	<input checked="" type="checkbox"/>
	12	DrogiTyp	Rower 4	<input checked="" type="checkbox"/>

Wypożycz

Zgłoś usterkę

Program poprawnie wyświetla listę stacji i listę dostępnych w na niej rowerów.

Rowery

	Id	Rodzaj roweru	Model	Czy jest funkcjonalny
▶	11	SuperTyp	Rower 5	<input checked="" type="checkbox"/>
	12	DrogiTyp	Rower 4	<input checked="" type="checkbox"/>

Wypożycz

Zgłoś usterkę

Po wybraniu opcji wypożycz rower zostaje usunięty ze stacji.

WidokGłównyUzytkownik

Stan konta: 240,2 zł

Wypożyczony rower: Rower 2

Cena za minutę: 0,8

Czas wypożyczenia: 00:01:02

Doładuj konto

40

Zwróć rower

Historia wypożyczeń

Wyświetl stacje

Konto

Użytkownik na stronie głównej ma wyświetlone nazwę roweru, cenę jego wypożyczenia oraz czas.

- Zwrot roweru

WyborStacji

Śliczna 21

Wybierz

Rowery

	Id	Rodzaj roweru	Model	Czy jest funkcjonalny
▶	5	SuperTyp	Rower 2	<input checked="" type="checkbox"/>
	11	SuperTyp	Rower 5	<input checked="" type="checkbox"/>
	12	DrogiTyp	Rower 4	<input checked="" type="checkbox"/>

Wypożycz Zgłoś usterkę

The screenshot shows a window titled "WidokGlownyUzytkownik". In the top right corner, it displays "Stan konta: 239,4 zł". Below this, on the left, are three labels: "Wypożyczony rower: Brak", "Cena za minutę: 0", and "Czas wypożyczenia: 00:00:00". To the right of these labels is a button labeled "Doładuj konto" and a text input field containing the number "40". At the bottom of the window, there are four buttons: "Zwróć rower" (highlighted with a blue border), "Historia wypożyczeń", "Wyświetl stacje", and "Konto".

Po zakończeniu wypożyczenia na wybranej stacji stan konta zostaje obniżony o odpowiednią kwotę. Rower zostaje przypisany do stacji wybranej przez użytkownika.

This screenshot shows the same "WidokGlownyUzytkownik" window as before, but with a modal dialog box in the center. The dialog box has a title bar with a close button (X) and contains the text "Nie wypożyczyłeś roweru" and an "OK" button. The background window elements are visible but slightly dimmed.

Aplikacja blokuje dostęp do okna wyboru stacji podczas zwrotu roweru jeśli użytkownik nie wypożycza aktualnie żadnego roweru.

	Id	Model	Nazwa	Start wypożyczenia	Koniec wypożyczenia
▶	1	Rower 1	Maciej123	03.12.2022 16:18	03.12.2022 16:21
	2	Rower 1	Maciej123	03.12.2022 18:52	03.12.2022 18:53
	3	Rower 1	Maciej123	03.12.2022 20:22	03.12.2022 20:24
	4	Rower 1	Maciej123	17.01.2023 14:34	17.01.2023 14:37
	5	Rower 1	Maciej123	17.01.2023 14:47	17.01.2023 14:49
	38	Rower 2	Maciej123	28.01.2023 22:43	28.01.2023 22:44

W historii wypożyczeń zostaje zapisana nowa pozycja.

- Wyjście z aplikacji

Logowanie

Nazwa

Hasło

☐ Zaloguj jako pracownik

Zaloguj się

Nazwa

Hasło

Powtórz hasło

Imię

Wyjście z programu

Czy napewno chcesz wyjść z aplikacji?

TakNie

Rejestracja

Aplikacja wyświetla komunikat chroniący przed przypadkowym jej wyłączeniem.

- Zgłoszenie usterki

The screenshot shows a window titled "Rowery". It contains a table with the following data:

	Id	Rodzaj roweru	Model	Czy jest funkcjonalny
▶	5	SuperTyp	Rower 2	<input checked="" type="checkbox"/>
	12	DrogiTyp	Rower 4	<input checked="" type="checkbox"/>

Below the table are two buttons: "Wypożycz" and "Zgłoś usterkę".

Po wybraniu opcji zgłoś usterkę parametr roweru (w tym przypadku „Rower 5”) „czy jest funkcjonalny” zostaje ustawiony na false i nie jest dalej wyświetlany użytkownikom na liście.

The screenshot shows a window titled "Usterki". It contains a table with the following data:

	Id	Zweryfikowana	Model	Nazwa	Imie	Nazwisko
▶	1	<input type="checkbox"/>	Rower 5			

Below the table are two buttons: "Moje usterki" and "Zweryfikuj".

Pracownik widzi zgłoszoną usterkę i wybierając opcję weryfikuj zostaje ona do niego przypisana i jest dostępna do podgląd w opcji „Moje usterki”.

MojeUsterki

	Id	Zweryfikowana	Model	Nazwa	Imie	Nazwisko
▶	1	<input checked="" type="checkbox"/>	Rower 5	Pracownik123	Jan	Kowal

Odrzuć

MojeUsterki

	Id	Zweryfikowana	Model	Nazwa	Imie	Nazwisko
--	----	---------------	-------	-------	------	----------

Odrzuć

Pracownik, jeśli zdecyduje, że rower jest już sprawny może usunąć usterkę i przywrócić rower jako dostępny do wypożyczenia.

Rowery

	Id	Rodzaj roweru	Model	Czy jest funkcjonalny
▶	5	SuperTyp	Rower 2	<input checked="" type="checkbox"/>
	11	SuperTyp	Rower 5	<input checked="" type="checkbox"/>
	12	DrogiTyp	Rower 4	<input checked="" type="checkbox"/>

Wypożycz Zgłoś usterkę

Status roweru „czy jest funkcjonalny” zostaje zamieniony na true i rower znów wyświetla się użytkownikom.