

Wrocław, 06.06.2018

PROGRAMOWANIE OBIEKTOWE

INEW0003P

Projekt

Wydział Elektroniki	Kierunek: Informatyka
Grupa zajęciowa: Wt. 17:05	Semestr: 2017/18 LATO
Nazwisko i Imię: Tobiasz Puślecki	Nr indeksu: 241354
Nazwisko i Imię: Milena Korusiewicz	Nr indeksu: 241327
Nr. grupy projektowej:	
Prowadzący: mgr inż. Jakub Zgraja	

TEMAT:

*Shelter - program do obsługi schroniska dla
bezdomnych zwierząt*

OCENA:

PUNKTY:

Data:

1. Założenia i opis funkcjonalny programu

Program wspomagający pracę schroniska dla bezdomnych zwierząt.

Pozwala ewidencjonować psy i koty.

Przechowywane dane o podopiecznym:

- Numer chipu (CID)
- Typ zwierzęcia
- Imię zwierzęcia
- Płeć
- Rasa
- Numer boxu (lokalizacja na terenie schroniska)
- Datę i miejsce odnalezienia
- Informacje nt. ewentualnych chorób
- Kolor sierści i kolor oczu
- Czy został adoptowany? (jeśli tak to można sprawdzić dane właściciela, tzn. jego imię, nazwisko i numer telefonu)

Inne funkcjonalności to:

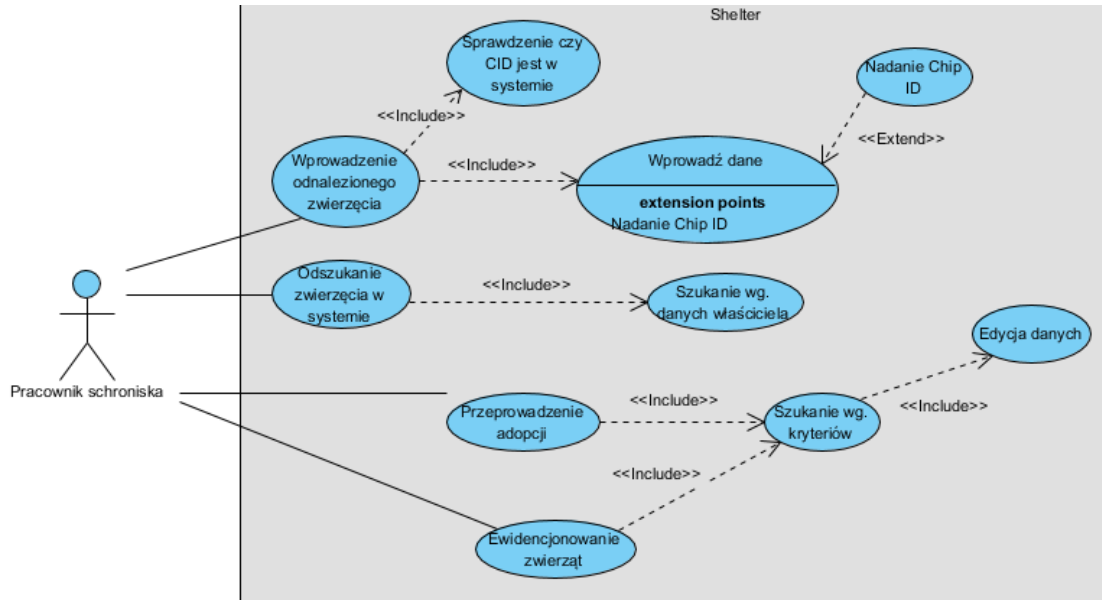
- Moduł odpowiedzialny za adopcje
- Katalog numerów chipów odnalezionych zwierząt (po CID można odnaleźć właściciela w spisie)

Scenariusze użycia:

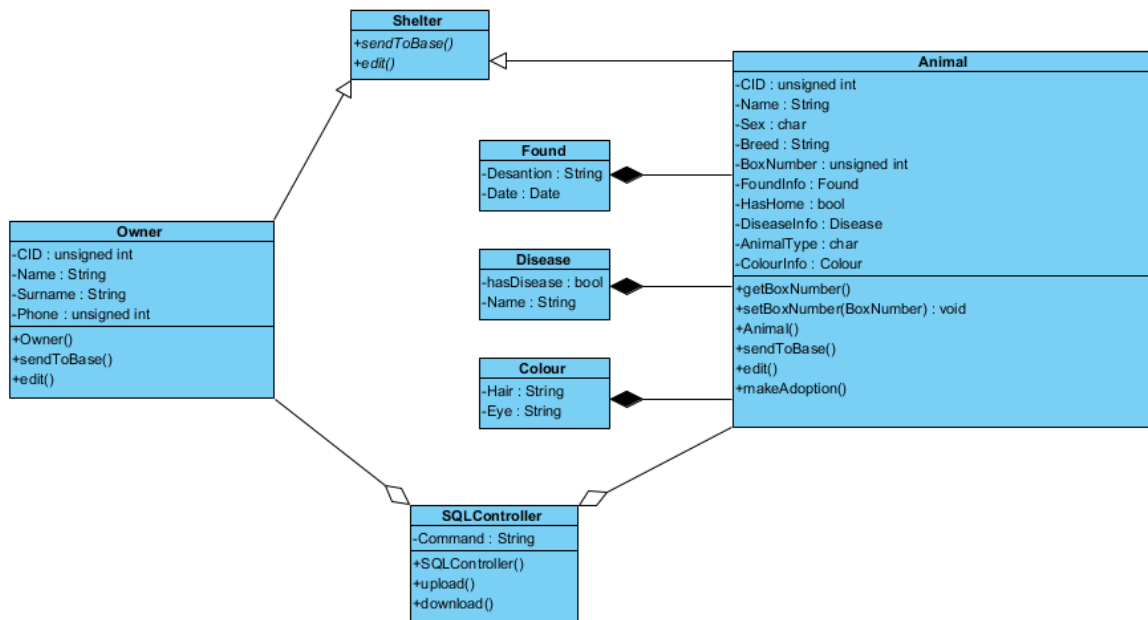
1. Odnalezione zwierzę trafia do schroniska, gdzie sprawdza się czy jest zachipowane. Jeśli tak, to w module „Znajdź właściciela” odszukuje się dane właściciela. W przeciwnym wypadku nadawany jest unikalny numer CID, a zwierzę dodawane do bazy w module „Znalezione”.
2. Właściciel przychodzi do schroniska po zgubione zwierzę i jeśli jest zarejestrowany w systemie, to na podstawie danych odszukuje się jego psa lub kota. Gdy właściciel nie był nigdy rejestrowany, należy szukać zwierzęcia przez moduły „Przeglądaj” lub „Tryb ręczny”, uwzględniając podane przez petenta cechy.
3. Przeprowadzanie adopcji, które polega na powiązaniu CID zwierzęcia z nowowprowadzonymi danymi nowego właściciela.
4. Możliwość szukania według zadanych kryteriów zwierząt, które je spełniają. Przeglądanie profili pozwala odnaleźć podopiecznych według np. daty przyjęcia, choroby.

2. Diagramy UML

a) Przypadków użycia



b) Klas



3. Kod klas C++ wraz z opisem

```
1  #ifndef LOGINWINDOW_H
2  #define LOGINWINDOW_H
3
4  #include <QDialog>
5  #include "ui_loginwindow.h"
6  #include "mainwindow.h"
7  #include <QApplication>
8  #include <QMessageBox>
9
10
11 struct LoginData
12 {
13     QString login;
14     QString password;
15
16     bool operator ==(const LoginData & ld)
17     {
18         if( ( password == ld.password ) &&( login == ld.login ) )
19             return true;
20         else
21             return false;
22     }
23 };
24
25 namespace Ui {
26     class loginWindow;
27 }
28
29 class loginWindow : public QDialog
30 {
31     Q_OBJECT
32
33 public:
34     explicit loginWindow(QWidget *parent = 0);
35     ~loginWindow();
36
37 private slots:
38     void on_pushButtonLogin_clicked();
39     void on_pushButtonExit_clicked();
40
41 private:
42     Ui::loginWindow *ui;
43
44     LoginData correct;
45     LoginData toCheck;
46
47 };
48
49 #endif // LOGINWINDOW_H
50
51
```

Struktura LoginData służy do przechowywania danych logowania: loginu i hasła. Zawiera też przeciążony operator == do sprawdzania zgodności danych. Klasa loginWindow zawiera dwa sloty obsługujące zdarzenie naciśnięcia przycisku oraz prywatne pola przechowujące podaną oraz poprawną wersję danych logowania.

```

#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QDialog>
#include "sqlcontroller.h"
#include "ui_mainwindow.h"
#include "newanimalwindow.h"
#include "adoptionwindow.h"
#include "filtersearchwindow.h"
#include "manualmodewindow.h"
#include "ownerwindow.h"

namespace Ui {
class mainWindow;
}

class mainWindow : public QDialog
{
    Q_OBJECT

public:
    explicit mainWindow(QWidget *parent = 0);
    ~mainWindow();

    void disabledAllButtons();

private slots:
    void on_pushButtonExit_clicked();

    void on_pushButtonFound_clicked();

    void on_pushButtonAdoption_clicked();

    void on_pushButtonFilterSearch_clicked();

    void on_pushButton_clicked();

    void on_pushButtonOwner_clicked();

private:
    Ui::mainWindow *ui;

};

#endif // MAINWINDOW_H

```

Klasa Main Window reprezentuje główne okno programu, zawiera funkcję odpowiadającą za deaktywowanie przycisków, pięć slotów połączonych z sygnałami przycisków odpowiadających za wybór odpowiedniej opcji. Konstruktor zawiera kod odpowiedzialny za układ graficzny oraz wyświetlanie ilości psów i kotów.

```

#ifndef NEWANIMALWINDOW_H
#define NEWANIMALWINDOW_H

#include "sqlcontroller.h"
#include <QDialog>
#include "ui_newanimalwindow.h"
#include "sqlcontroller.h"
#include <QMessageBox>

namespace Ui {
class newAnimalWindow;
}

class newAnimalWindow : public QDialog
{
    Q_OBJECT

public:
    explicit newAnimalWindow(QWidget *parent = 0);
    ~newAnimalWindow();

private slots:

    void on_pushButtonCancel_clicked();

    void on_pushButtonConfirm_clicked();

private:
    Ui::newAnimalWindow *ui;

};

#endif // NEWANIMALWINDOW_H

```

Klasa newAnimalWindow zawiera sloty obsługujące naciśnięcia przycisków potwierdzenia i anulowania.

```

#ifndef OWNERWINDOW_H
#define OWNERWINDOW_H

#include <QDialog>
#include "ui_ownerwindow.h"
#include "sqlcontroller.h"

namespace Ui {
class ownerWindow;
}

class ownerWindow : public QDialog
{
    Q_OBJECT

public:
    explicit ownerWindow(QWidget *parent = 0);
    ~ownerWindow();

private slots:
    void on_pushButtonCancel_clicked();

    void on_pushButtonConfirm_clicked();

private:
    Ui::ownerWindow *ui;
};

#endif // OWNERWINDOW_H

```

Klasa ownerWindow obsługuje opcję znajdowania właściciela zwierzęcia, zawiera sloty obsługujące przyciski.

```

#ifndef ADOPTIONWINDOW_H
#define ADOPTIONWINDOW_H

#include <QDialog>
#include "ui_adoptionwindow.h"
#include "sqlcontroller.h"
#include <QMessageBox>

namespace Ui {
class adoptionWindow;
}

class adoptionWindow : public QDialog
{
    Q_OBJECT

public:
    explicit adoptionWindow(QWidget *parent = 0);
    ~adoptionWindow();

private slots:
    void on_pushButtonCancel_clicked();

    void on_pushButtonConfirm_clicked();

private:
    Ui::adoptionWindow *ui;
};

#endif // ADOPTIONWINDOW_H

```

Klasa adoptionWindow odpowiada za obsługę adoptowania zwierząt, zawiera sloty połączone z przyciskami.


```

#ifndef FILTERSEARCHWINDOW_H
#define FILTERSEARCHWINDOW_H

#include <QDialog>
#include "ui_filtersearchwindow.h"
#include "sqlcontroller.h"
#include "shelter.h"
#include <QMessageBox>

namespace Ui {
class filterSearchWindow;
}

class filterSearchWindow : public QDialog
{
    Q_OBJECT

public:
    explicit filterSearchWindow(QWidget *parent = 0);
    ~filterSearchWindow();

private slots:

    void on_pushButtonCancel_clicked();

    void on_pushButtonLeftArrow_clicked();

    void on_pushButtonRightArrow_clicked();

    void on_pushButtonSearch_clicked();

private:
    Ui::filterSearchWindow *ui;
};

#endif // FILTERSEARCHWINDOW_H

```

```

#ifndef SHELTER_H
#define SHELTER_H

#include <QString>

class Shelter
{
public:
    Shelter();
    virtual void sendToBase() = 0;
    //virtual void edit() = 0;
    QString cid;
    QString name;
};

class Animal : public Shelter
{
public:
    QString sex;
    QString breed;
    QString boxNumber;
    QString foundStreet;
    QString foundDate;
    QString hasHome;
    QString hasDisease;
    QString diseaseName;
    QString eyeColour;
    QString hairColour;
    QString animalType;

    virtual void sendToBase();
};

class Owner : public Shelter
{
public:
    Owner(QString cid, QString name, QString surname, QString phone);
    Owner();

    virtual void sendToBase();

    QString surname, phone;
};

```

Klasa Shelter jest nadklasą klas Animal i Owner. Klasa Animal zawiera pola przechowujące informacje o cechach zwierzęcia. Metoda sendToBase wysyła informacje do bazy danych. Konstruktor klasy Owner ustawia wartości odpowiadające informacjom o właścicielu.

```

#ifndef MANUALMODEWINDOW_H
#define MANUALMODEWINDOW_H

#include <QDialog>
#include "ui_manualmodewindow.h"
#include "sqlcontroller.h"

namespace Ui {
class manualModeWindow;
}

class manualModeWindow : public QDialog
{
    Q_OBJECT

public:
    explicit manualModeWindow(QWidget *parent = 0);
    ~manualModeWindow();

private slots:
    void on_pushButtonExit_clicked();

    void on_pushButtonConfirm_clicked();

private:
    Ui::manualModeWindow *ui;
};

#endif // MANUALMODEWINDOW_H

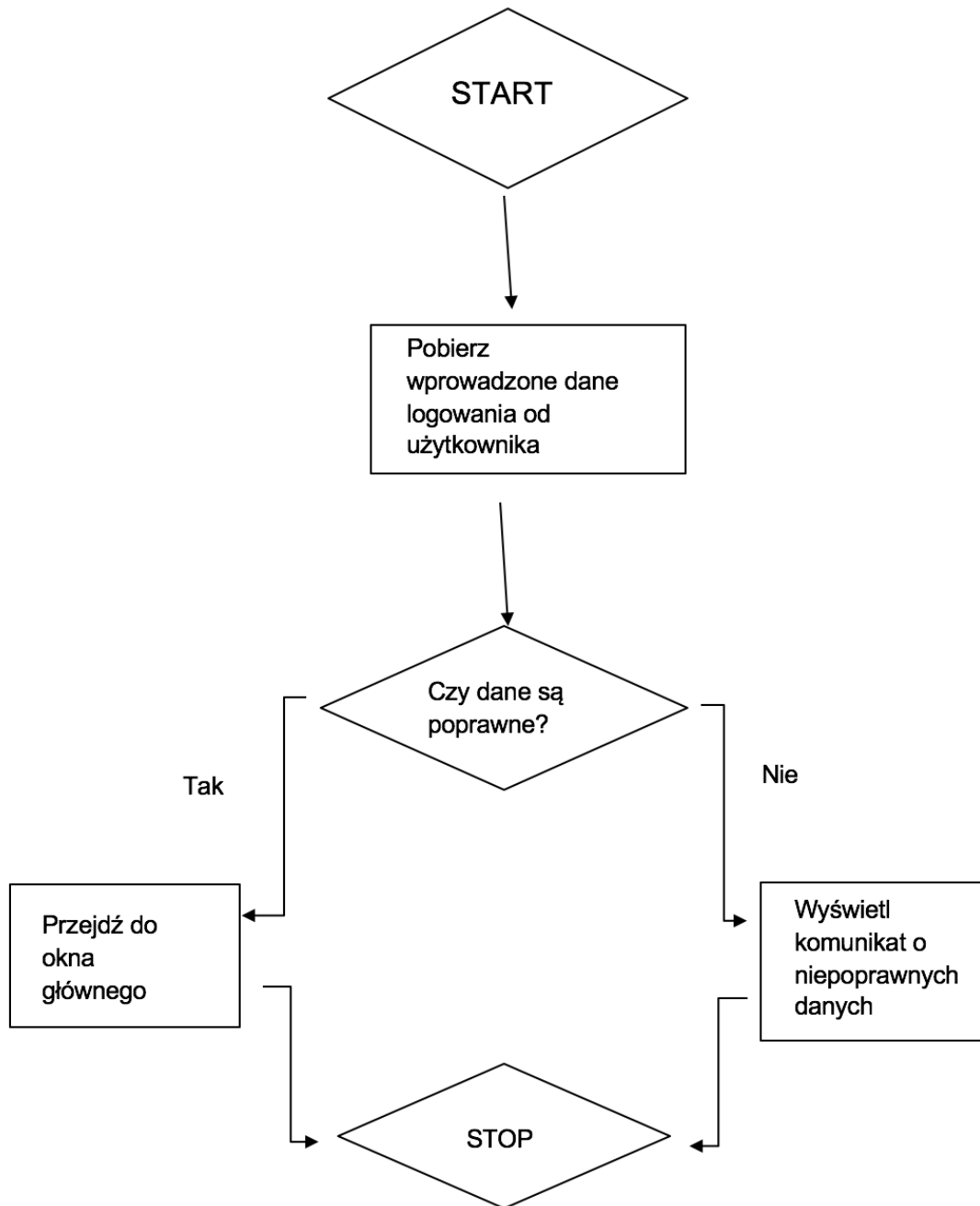
```

Klasa manualModeWindow odpowiada za obsługę trybu manualnego.

4. Schematy blokowe oraz kod własnych funkcji C++ / C#

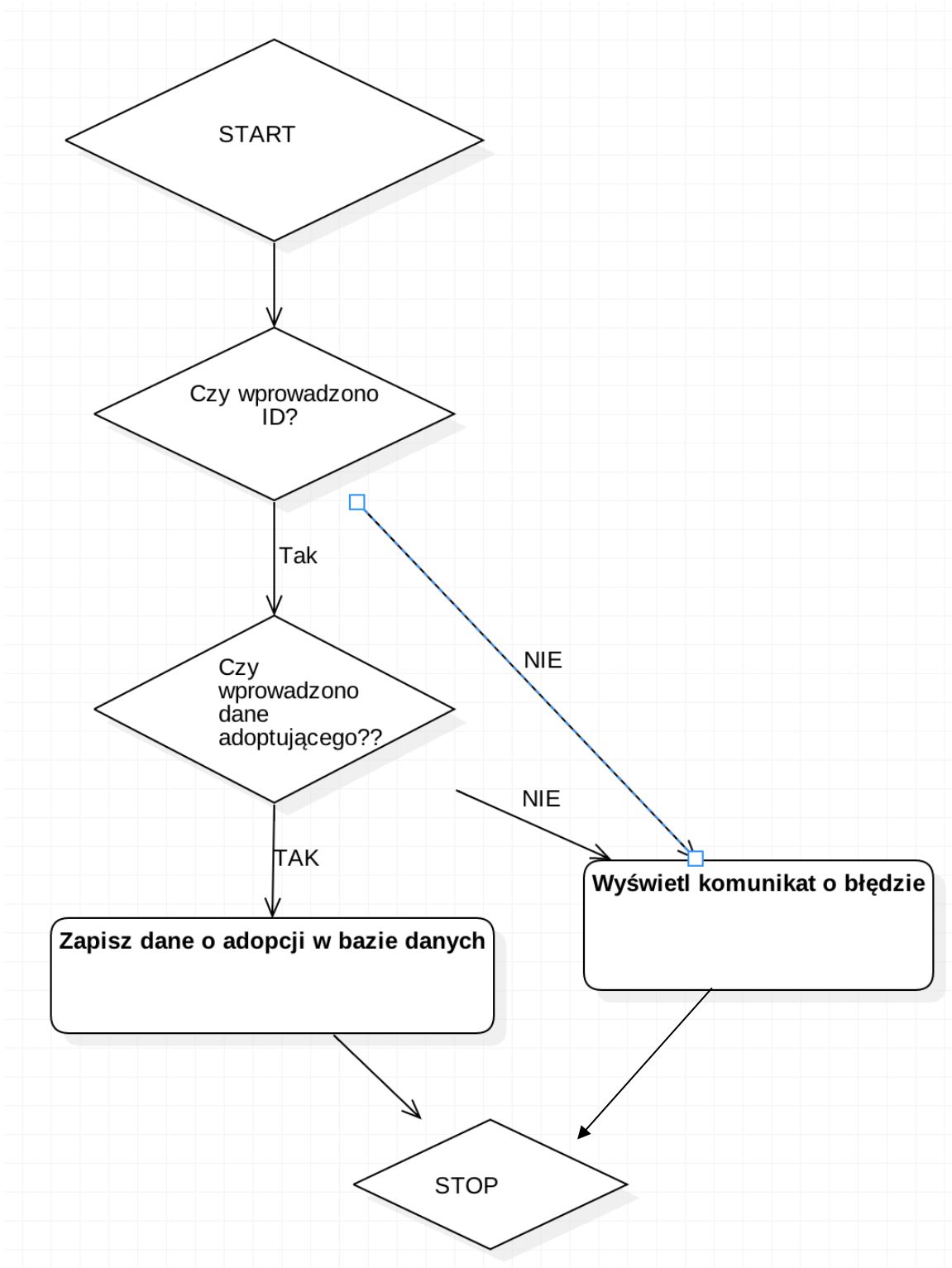
Funkcja loginWindow::on_pushButtonLogin_clicked:

Zadaniem funkcji jest sprawdzenie poprawności danych logowania poprzez porównanie ich z danymi prawidłowymi. Robi to poprzez ich porównanie za pomocą przeciążonego operatora ==.

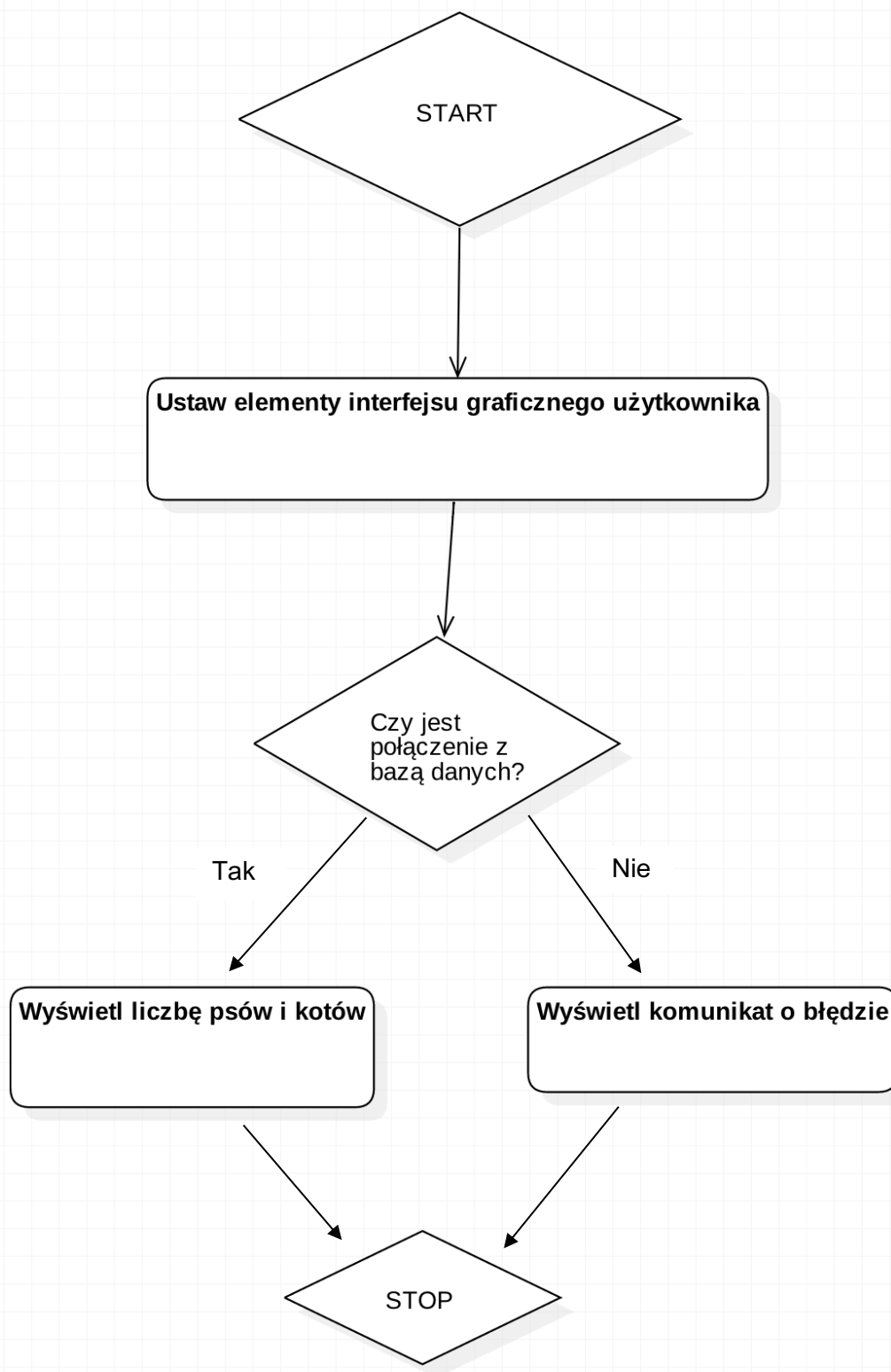


Funkcja void adoptionWindow::on_pushButtonConfirm_clicked():

Funkcja ta odpowiada za sprawdzenie poprawności danych dla adopcji oraz obsługuje jej proces.



Konstruktor `mainWindow::mainWindow(QWidget *parent)` odpowiada za wyświetlenie interfejsu użytkownika dla głównego okna i wczytanie danych o aktualnej liczbie psów i kotów.



Funkcja void newAnimalWindow::on_pushButtonConfirm_clicked():

Funkcja zatwierdzająca dodanie nowego zwierzęcia i sprawdzająca poprawność wprowadzonych danych. W przypadku nieprawidłowych danych zgłasza błąd.

Konstruktor Owner::Owner(QString cid, QString name, QString surname, QString phone):

Wprowadza dane do pól klasy Owner.

Funkcja void Animal::sendToBase():

Wysyła do bazy danych dane o zwierzęciu.

Funkcja int SQLController::getNumberOf(QString type, int cid):

Zwraca liczbę zwierząt danego gatunku.

void SQLController::makeAdoption(QString cid, QString name, QString surname, QString phone):

Aktualizuje bazę danych po przeprowadzeniu adopcji.

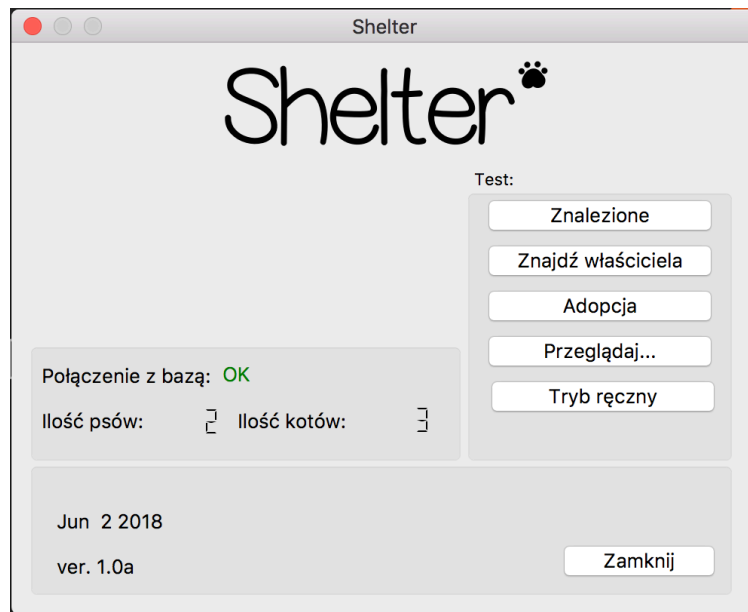
5. Opis użytkowy programu C++ / C#

• LOGOWANIE:



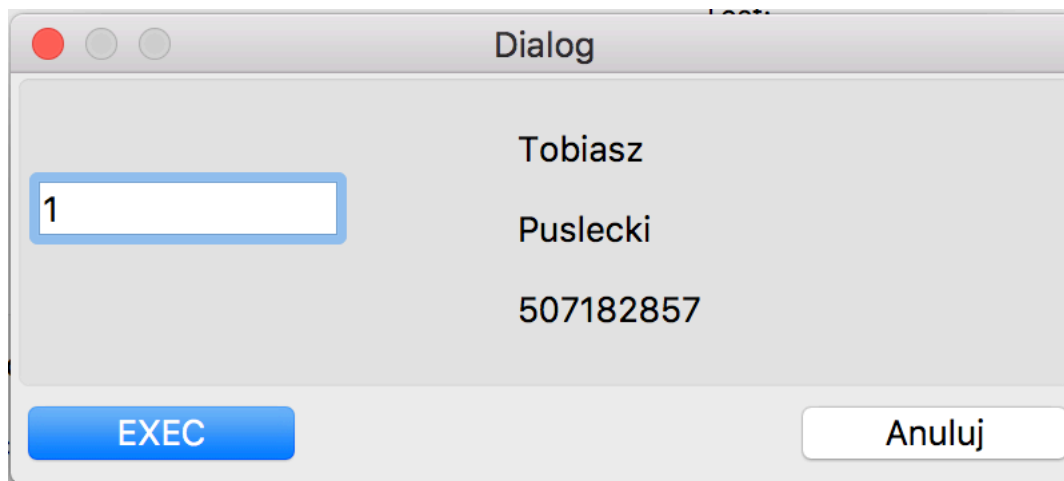
Jest to okno logowania. Istnieje możliwość zalogowania się do systemu poprzez wpisanie loginu i hasła ustalonego przez użytkownika programu schroniska. Domyślne dane logowania to login: root oraz hasło: root

- **MENU GŁÓWNE**



Po zalogowaniu do systemu ukazuje się okno główne, zawiera ono informacje o stanie połączenia z bazą oraz liczbie i gatunku podopiecznych schroniska. Dostępne są też opcje dodania znalezionego zwierzęcia za pomocą przycisku „znalezione”, opcja poszukiwania właściciela danego zwierzęcia, menu adopcji oraz przeglądanie cech i parametrów wybranego zwierzęcia, można też użyć trybu ręcznego. Na dole okna widnieje przycisk „Zamknij” służący do opuszczenia danego okna.

- **ZNAJDŹ WŁAŚCICIELA**



Okno znajdowania właściciela zawiera miejsce na wpisanie numeru identyfikującego zwierzę w celu odnalezienia właściciela. Po prawej stronie okna ukazują się jego dane.

- **ZNALEZIONE**

Dodawanie nowego zwi...

☒ Pies ☐ Kot

☐ Samiec ☒ Samica

Rex

12

☒ Choroba? FELV

Dog

2018-06-06

Grunwaldzka

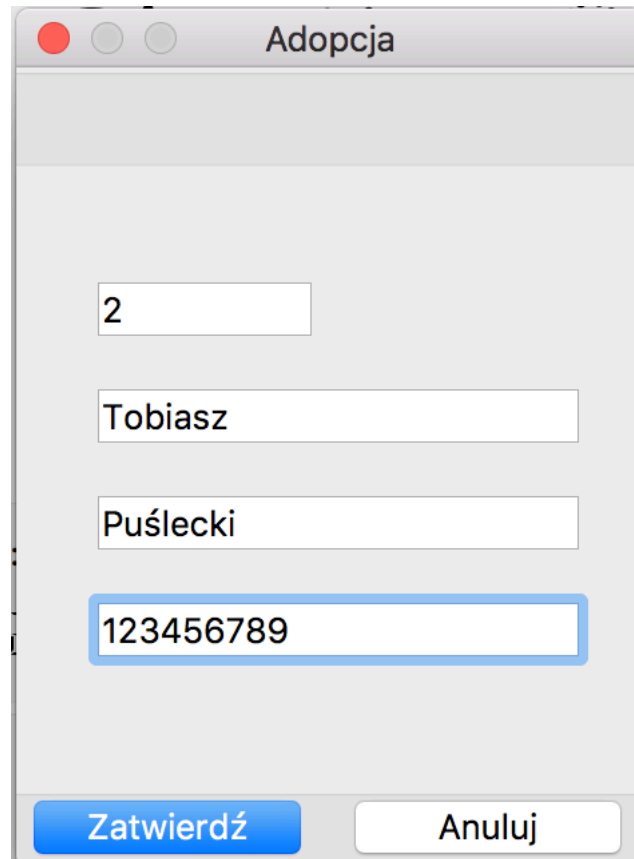
Czarny

Zielone

Dodaj Anuluj

Wchodząc w opcję „znalezione” mamy możliwość dodania nowego zwierzęcia i wprowadzenia jego danych.

- **ADOPCJA**



Adopcja

2

Tobiasz

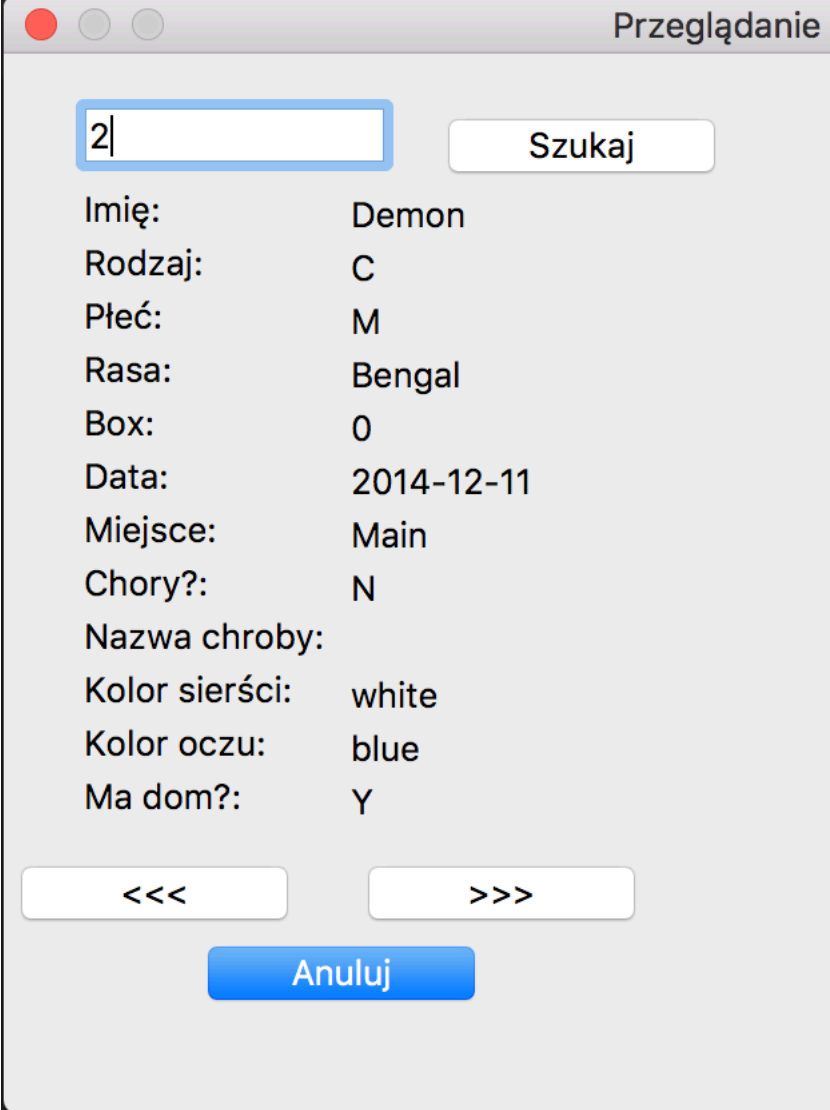
Puślecki

123456789

Zatwierdź Anuluj

Opcja „Adopcja” umożliwia zarejestrowanie zwierzęcia jako zaadoptowanego. W pierwszej linii należy wpisać numer zwierzęcia, które ma zostać zaadoptowane. W następnych liniach dane osoby planującej adopcję, odpowiednio: imię, nazwisko oraz numer telefonu. Adopcji dokonujemy przyciskiem „Zatwierdź”.

- **PRZEGLĄDANIE**

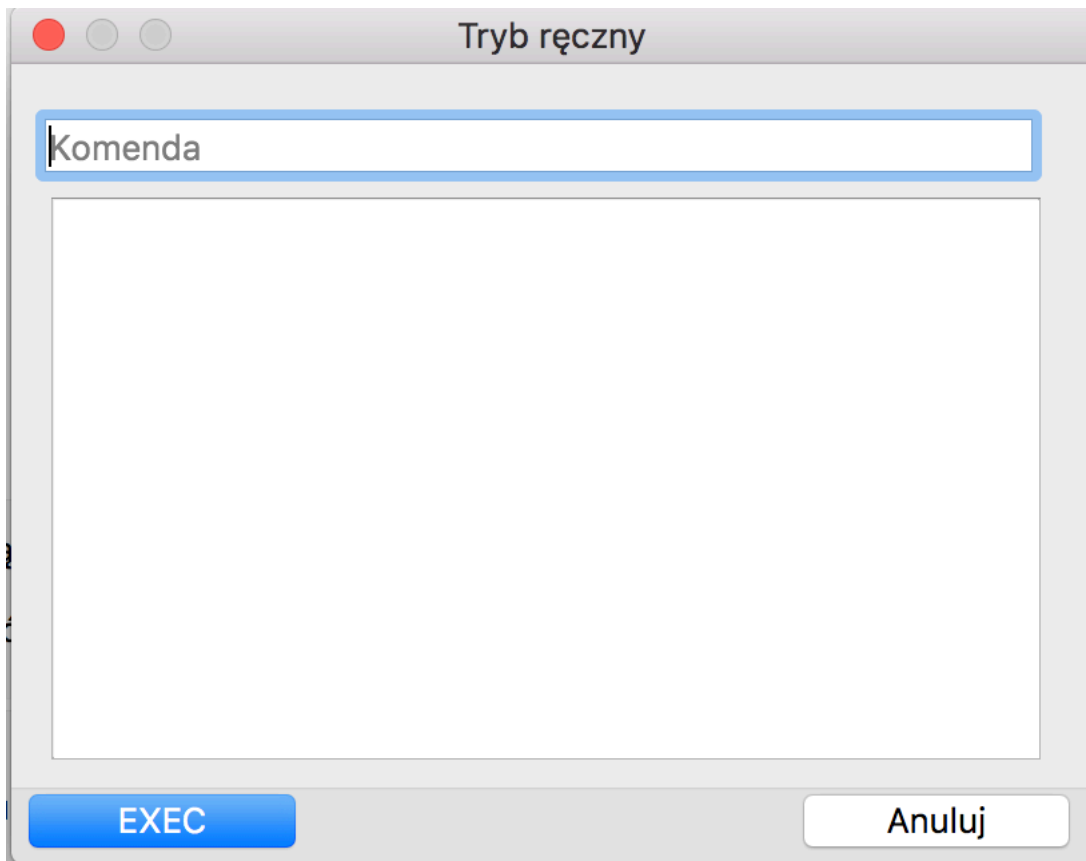


The screenshot shows a window titled "Przeglądanie" (Browse) with a search bar containing the number "2" and a "Szukaj" (Search) button. Below the search bar, a list of attributes for a specific animal is displayed. At the bottom, there are navigation buttons: "<<<", ">>>", and a blue "Anuluj" (Cancel) button.

Imię:	Demon
Rodzaj:	C
Płeć:	M
Rasa:	Bengal
Box:	0
Data:	2014-12-11
Miejsce:	Main
Chory?:	N
Nazwa choroby:	
Kolor sierści:	white
Kolor oczu:	blue
Ma dom?:	Y

Powyższe okno przedstawia wybraną opcję „Przeglądaj”, dzięki której poprzez wpisanie numeru identyfikującego zwierzę możemy poznać jego cechy charakterystyczne, na przykład płeć, choroby czy informację o tym czy ma dom (to znaczy czy został już zaadoptowany).

- **TRYB RĘCZNY**



Tryb ręczny umożliwia manualną obsługę programu poprzez wpisywanie komend w górnej linii.

6. Listing kodu C++ / C# – wraz z komentarzami.

<Wstawiono na ePortal>

7. Wnioski

Zaimplementowano podstawowe funkcjonalności i zrealizowano część początkowych założeń. Program nadaje się do użytku, jednak brakuje lepszego kreatora zapytań ręcznych, do którego obsługi nie byłaby wymagana znajomość SQL-a. Nie udało się dobrze zoptymalizować kodu oraz w niektórych miejscach sprawić, aby był on zgodny z wymaganiami. Zabrakło również sprawdzania poprawności danych przed wysłaniem ich do bazy oraz komunikacji zwrotnej z bazą (nie ma informacji czy zapytanie powiodło się).