

# SYSTEM OBSŁUGI PRZYCHODNI WETERYNARYJNEJ

## Spis treści

1. Use case
  - 1.1. Opis przypadku
  - 1.2. Diagram aktywności
  - 1.3. Diagram stanów
  - 1.4. Diagram sekwencji
2. Diagram klas
  - 2.1. PRI
  - 2.2. Decyzje implementacyjne
  - 2.3. Diagram implementacyjny
3. Projekt GUI przypadku użycia

### Projekt PRI

1. Dziedzina problemowa
2. Cel
3. Zakres odpowiedzialności systemu
4. Użytkownicy systemu
5. Wymagania użytkownika
6. Wymagania funkcjonalne
7. Opis struktury systemu
8. Wymagania нефunkcjonalne
9. Opis przyszłej ewolucji systemu

## 1. Use case

Scenariusz dla przypadku użycia: Zarejestruj wizytę

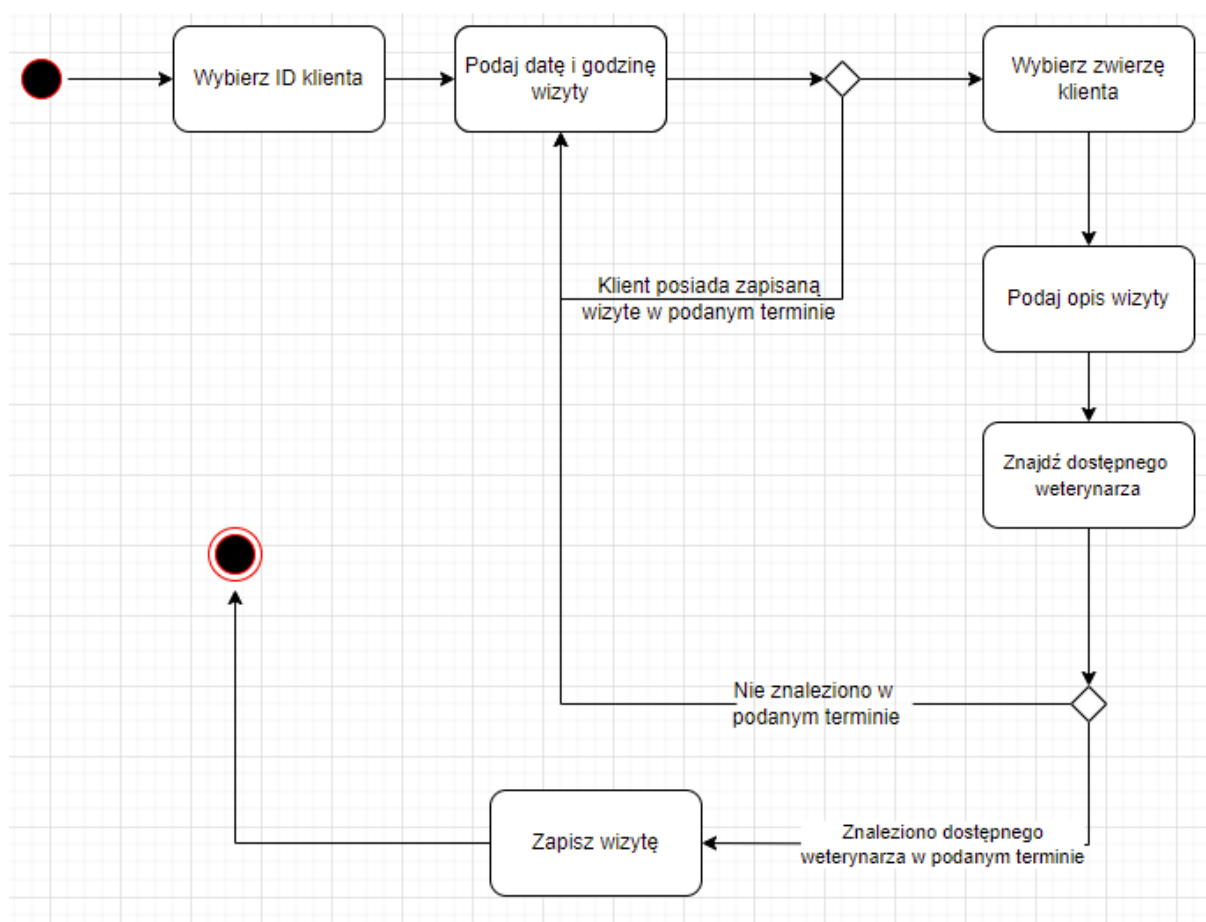
### 1.1. Opis przypadku

Use diagram – Zarejestruj wizytę

Nazwa przypadku użycia	Zarejestruj wizytę
Warunek początkowy	Klient jest zarejestrowany i posiada co najmniej jedno zarejestrowane zwierzę
Główny przepływ zdarzeń	1. Aktor Recepcjonista uruchamia przypadek użycia. 2. Aktor wybiera ID klienta 3. Aktor wybiera datę i godzinę wizyty, 4. Aktor wybiera zwierzę klienta, którego dotyczy wizyta 5. Aktor wypełnia opisu problemu (informacje od klienta) 6. Aktor klika przycisk 'znajdź weterynarza' - system przypisuje pierwszego weterynarza, który jest dostępny w podanym terminie 7. Aktor recepcjonista klika przycisk 'zapisz'
Alternatywne przepływy zdarzeń	3a. System informuje o istniejącej wizycie u wybranego klienta w podanej dacie/godzinie 6a. System informuje o braku dostępnego weterynarza w podanym terminie
Zakończenie	Gdy wizyta zostaje zapisana / proces anulowany
Warunek końcowy	brak

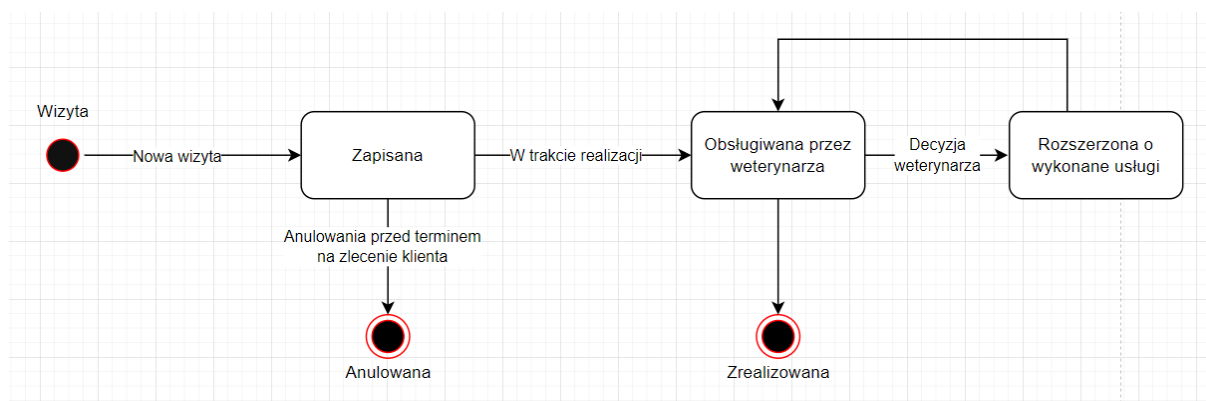
## 1.2. Diagram aktywności

Activity diagram – Zarejestruj wizytę



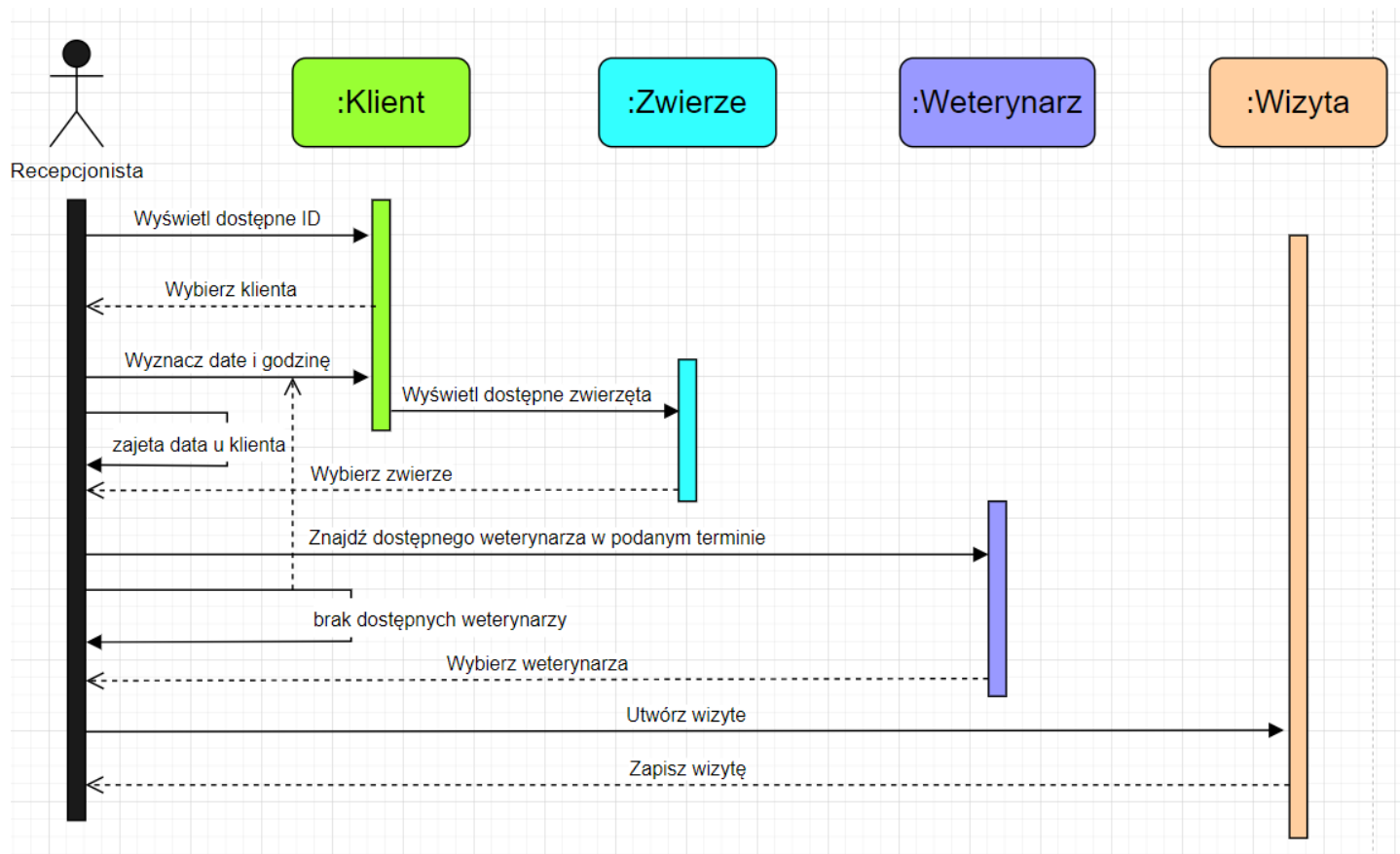
## 1.3. Diagram stanów

State machine – Wizyta



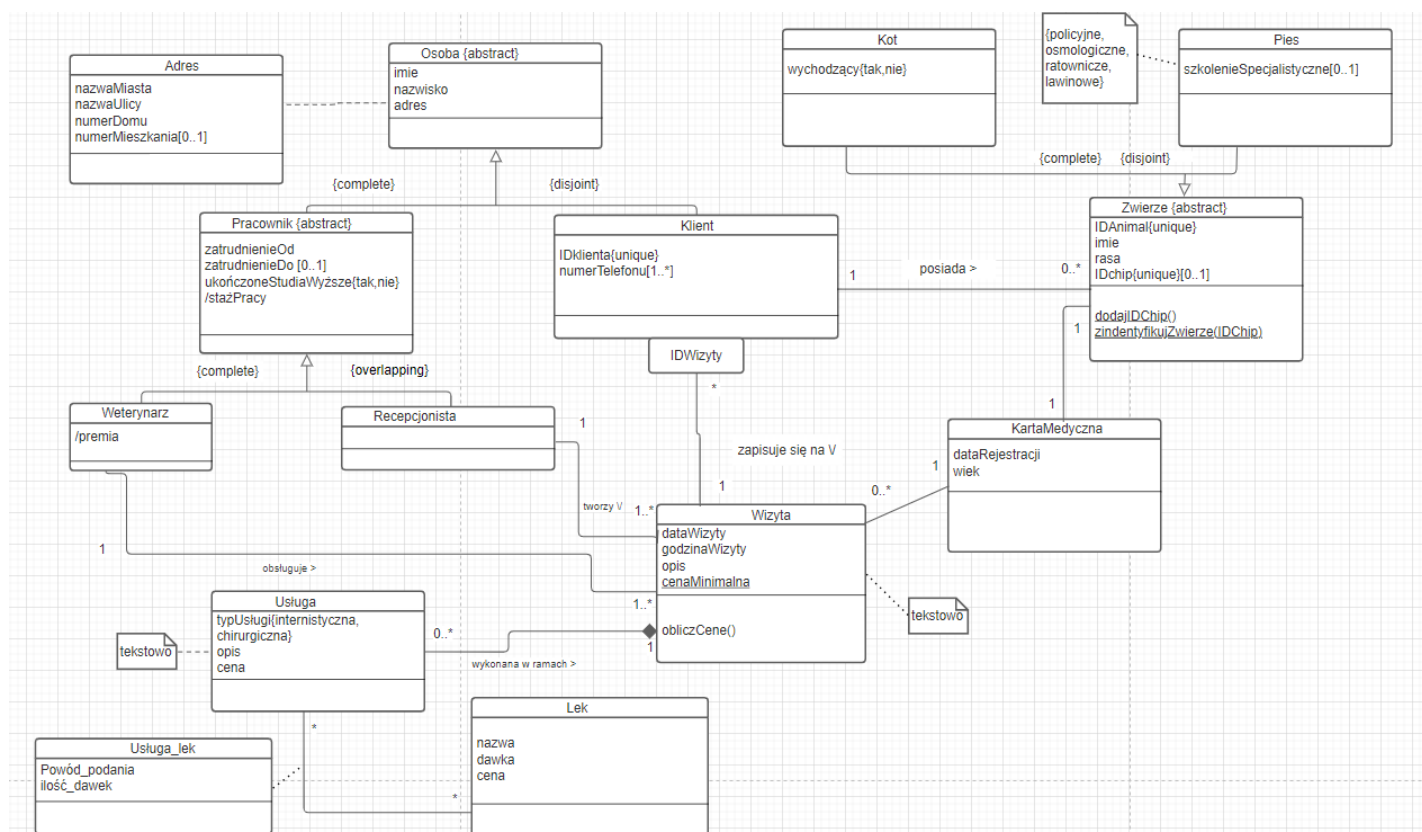
## 1.4. Diagram sekwencji

Sequence diagram – Zarejestruj wizytę

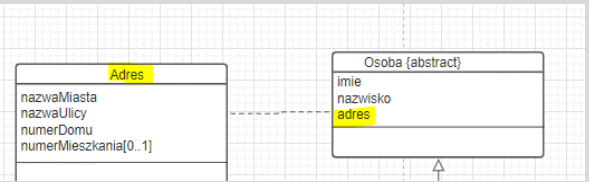
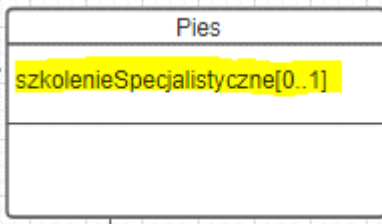


## 2. Diagram klas

### 2.1. PRI



Opis wymaganych elementów:

WYMAGANE KONSTRUKCJE			
Lp	Nazwa	Opis	Screenshot
1	Atrybut złożony	Klasa: Osoba Atrybut: adres Adres - oddzielna klasa	
2	Atrybut opcjonalny	Klasa: Pies Atrybut: szkolenieSpecial	

3	Atrybut powtarzalny	Klasa: Klient Atrybut: numerTelefonu (String)	<pre> classDiagram     class Klient {         IDklienta(unique)         numerTelefonu[1..*]     } </pre>
4	Atrybut klasowy	Klasa: Wizyta Atrybut: cenaMinimalna	<pre> classDiagram     class Wizyta {         dataWizyty         godzinaWizyty         opis         cenaMinimalna     } </pre>
5	Atrybut pochodny	Klasa: Pracownik Atrybut: stażPracy (DateNow - zatrudnienieOd w latach)	<pre> classDiagram     class Pracownik {         &lt;&lt;abstract&gt;&gt;         zatrudnienieOd         zatrudnienieDo [0..1]         ukończoneStudiaWyzsze{tak,nie}         stażPracy     } </pre>
6	Asocjacja zwykła	Zwierze - KartaMedyczna	<pre> classDiagram     class Zwierze {         &lt;&lt;abstract&gt;&gt;         IDAnimal(unique)         imie         rasa         IDchip(unique)[0..1]         +dodajIDChip()         +zidentyfikujZwierze(IDChip)     }     class KartaMedyczna {         dataRejestracji         wiek     }     Zwierze "0..*" -- "1" KartaMedyczna : posiada &gt; </pre>
7	Asocjacja z atrybutem	Usługa-Usługa_lek-Lek	<pre> classDiagram     class Usługa {         typUsługi(internistyczna, chirurgiczna)         opis         cena     }     class Usługa_lek {         Powód_podania         ilość_dawek     }     class Lek {         nazwa         dawka         cena     }     Usługa "0..*" -- "*" Usługa_lek : wykonana w ramach &gt; </pre>

8	Asocjacja kwalifikowana	Klient - [IdWizyty] Wizyta	
9	Kompozycja	Wizyta - Usługa	
10	Klasa abstrakcyjna	Zwierze	
11	Dziedziczenie overlapping	Pracownik - Weterynarz / Recepcjonista	

## 2.2. Decyzje implementacyjne

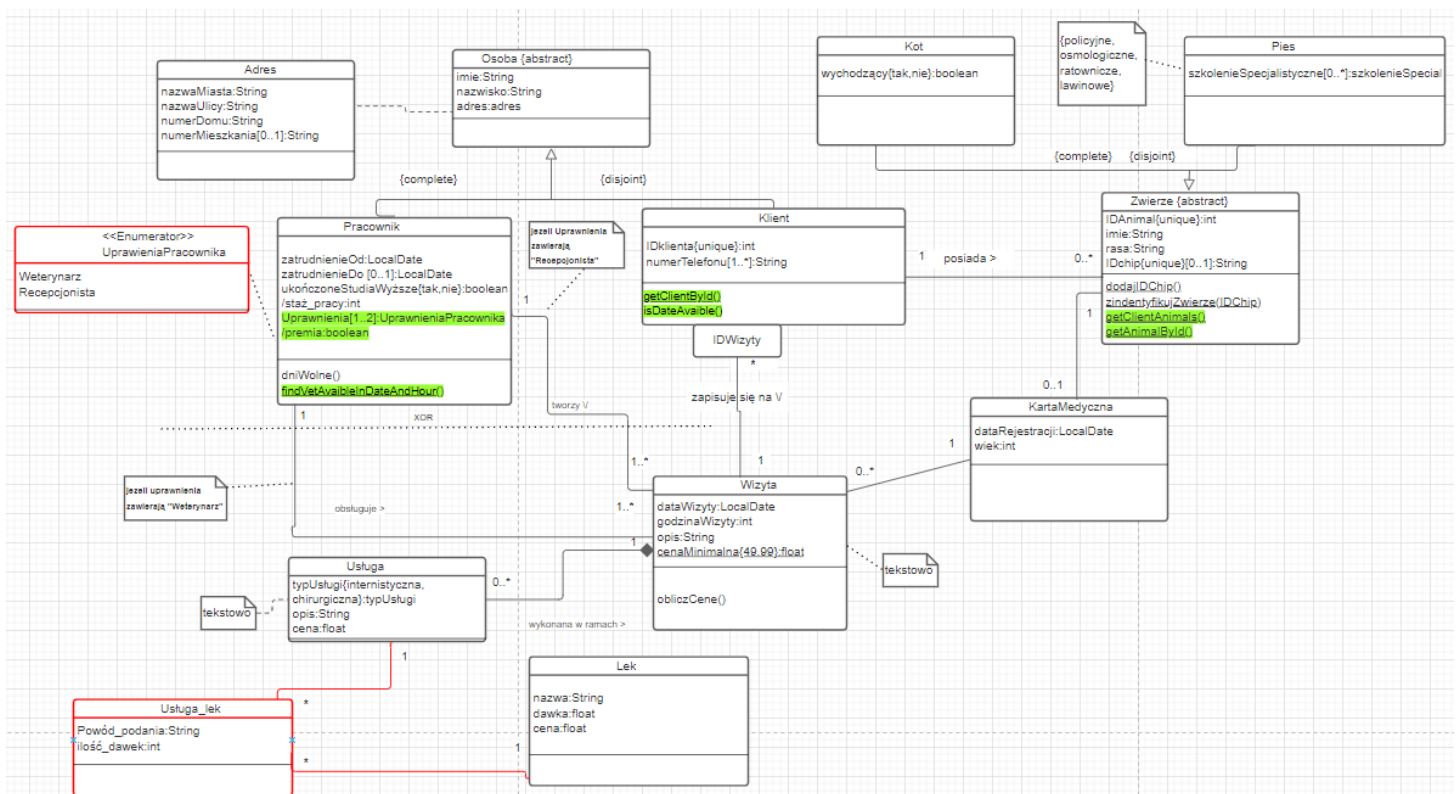
Uwzględniając możliwości implementacyjne języka JAVA oraz wynik analizy wybranego przypadku diagram implementacyjny posiada następujące zmiany:

- Spłaszczenie struktury [Weterynarz->Pracownik<-Recepcjonista] oraz utworzenia nowego typu wyliczeniowego [UprawnieniaPracownika]
- Klasa [Pracownik] nie jest już abstrakcyjna

Nowo powstałe klasy zostały oznaczone na diagramie implementacyjnym na **zielono**, za to nowe metody – na **czerwono**.

Do utrwalania danych zastosowano wbudowany z język JAVA mechanizm serializacji obiektów a samą kontrolę i odpowiedzialność za proces utrwalania i odtwarzania przekazano do nowo powstałej klasy *ExtensionManager*.

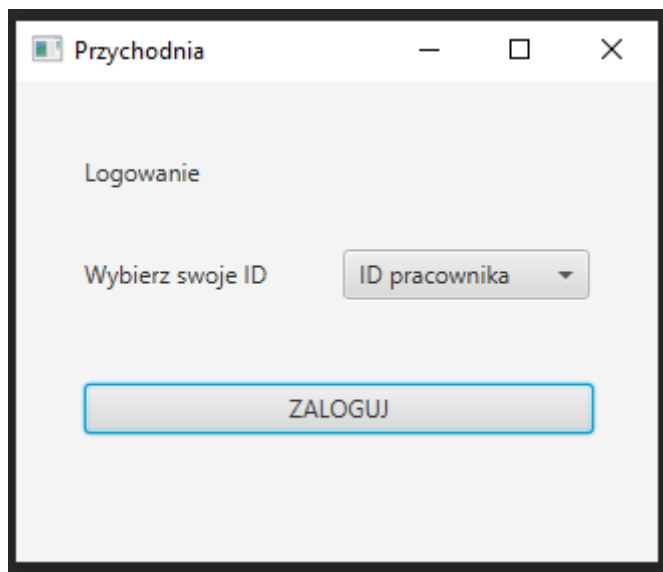
## 2.3. Diagram implementacyjny



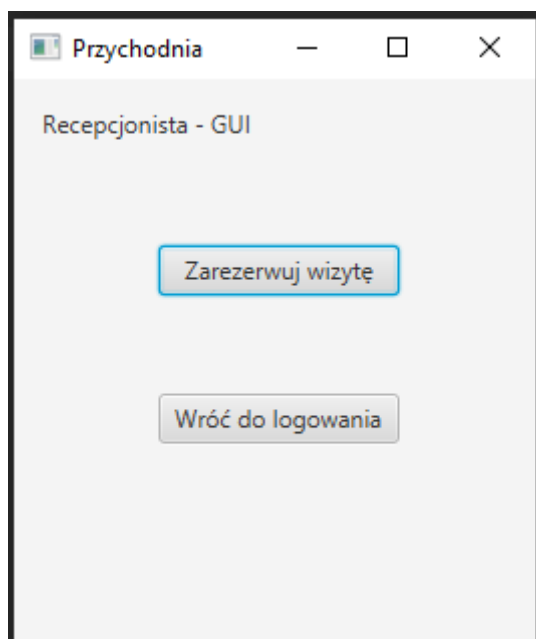


### 3. Projekt GUI przypadku użycia

Ekran początkowy – logowanie pracownika:



Ekran wyboru czynności:



Po kliknięciu przycisku „Zarezerwuj wizytę” – ekran do tworzenia nowej wizyty:

The screenshot shows a window titled "Przychodnia" with a standard Windows title bar (minimize, maximize, close buttons). Inside the window, there are four dropdown menus arranged vertically: "Wybierz ID klienta" with "ID klienta" selected, "Data wizyty" with "Data" selected, "Godzina wizyty" with "Godzina" selected, and "Wybierz ID zwierzęcia" with "ID zwierzęcia" selected. Below these is a text input field labeled "OPIS:". At the bottom of the window are two buttons: "Znajdź weterynarza" (highlighted with a blue border) and "ZAPISZ".

Po uzupełnieniu danych należy znaleźć weterynarza, który może obsłużyć wizytę (przycisk 'znajdź weterynarza').

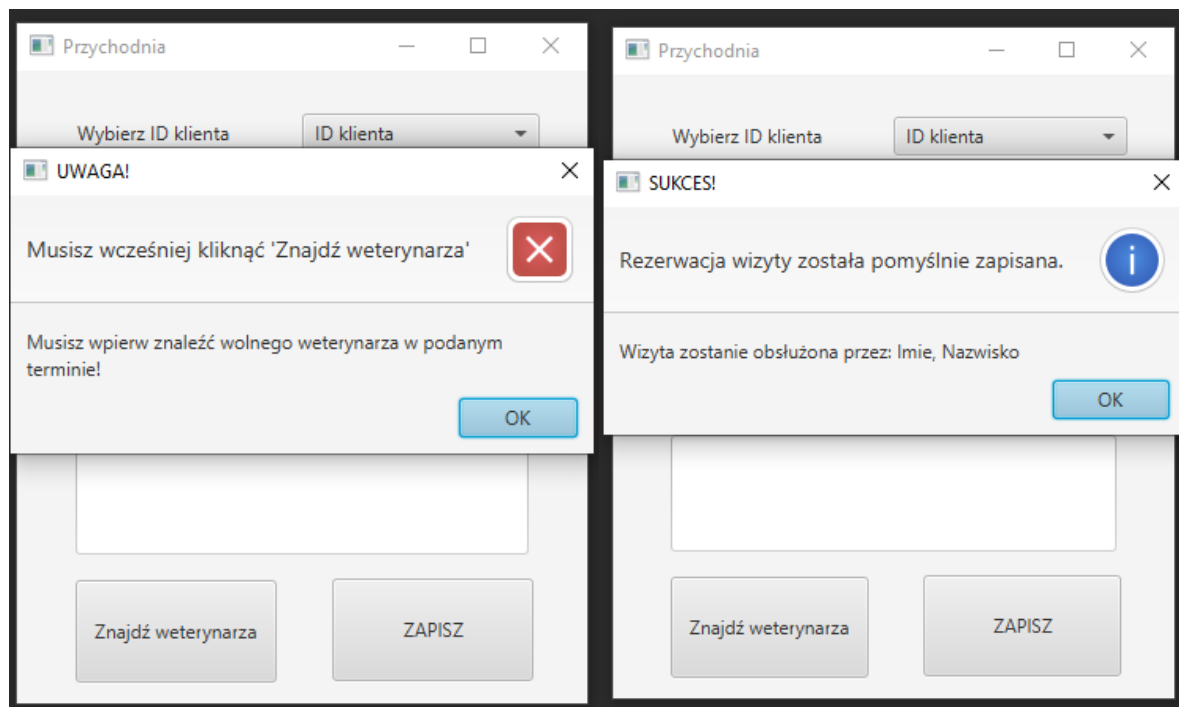
W przypadku braku wolnych weterynarzy w podanej dacie / godzinie – pokazuje się powiadomienie z odpowiednią informacją. Brak informacji oznacza powodzenie w odnalezieniu wolnego weterynarza.

This screenshot shows the same "Przychodnia" window, but with an error dialog box overlaid. The dialog box is titled "UWAGA!" and contains the text "Brak dostępnych weterynarzy!" followed by a red square button with a white 'X'. Below this, it says "Podany termin jest zajęty, zmień datę / godzinę wizyty!" and has a blue "OK" button. The background window is partially visible, showing the same dropdown menus and buttons as in the previous screenshot.

Przycisk „ZAPISZ” finalizuje cały proces i zapisuje wizytę na podstawie wybranych danych oraz odnalezonego weterynarza[2]. Nie ma możliwości zapisania wizyty bez wcześniejszego wyszukania weterynarza[1]

[1]

[2]



## TREŚĆ PROJEKTU PRI

### 1. Dziedzina problemowa:

Projekt systemu miałby zastosowanie w rezerwacji wizyt w przychodniach weterynaryjnych oferujących specjalistyczną opiekę zdrowotną psów i kotów.

### 2. Cel:

Możliwość rejestracji wizyt weterynaryjnych. Szybki wgląd do karty medycznej zwierzęcia i sprawdzenie szczegółów wizyty, stosowanych obecnie/historycznie leków. Możliwa indentyfikacja zwierząt po microchipie (jeżeli zostały zarejestrowane w bazie). Uzyskiwanie informacji o liczbie zaplanowanych wizyt.

### 3. Zakres odpowiedzialności systemu:

System będzie odpowiedzialny za rejestrację klientów, tworzenie i modyfikacji kart medycznych zwierząt zawierających pełną historię leczenia wraz z listą podawanych leków w czasie usług medycznych.

#### 4. Użytkownicy systemu:

Weterynarz, Recepcjonista, Klient

#### 5. Wymagania użytkownika:

System obsługi przychodni weterynaryjnej powinien przechowywać informacje dotyczące:

- klientów,
- zatrudnionego personelu
- umówionych wizyt wraz z historią zrealizowanych,
- kart medycznych zwierząt
- dostępnych leków do podania w ramach usługi weterynaryjnej

##### *Klienci*

Dane powinny zawierać:

- imię oraz nazwisko,
- ID klienta,
- adres zamieszkania
- numery telefonu (min. 1)

##### *Osoby zatrudnione:*

###### Weterynarz

- mogący uzyskać premie jeżeli w ciągu miesiąca kalendarzowego były wykonywane zabiegi chirurgiczne z opisem „chip”.
- obsługuje wizyty

###### Recepcjonista:

- rejestruje wizyty

###### Wizyty:

Dane powinny zawierać:

- datę wizyty,
- godzinę wizyty,
- tekstowy opis.

W skład wizyty mogą wchodzić:

- usługi wykonywane przez weterynarza zawierające:
  - typ (internistyczna lub chirurgiczna),
  - tekstowy opis,
  - cena usługi.

- (opcjonalnie)leki podane podczas wykonywania danej usługi:
  - nazwa,
  - dawka,
  - cena za dawkę

Wizyty są rejestrowane przez recepcjonistę na podstawie id klienta oraz karty medycznej zwierzęcia / zwierząt i obsługiwane przez weterynarza. System musi pozwolić rejestrację nowych wizyt przez wybranych weterynarzy (wyłącznie na innych weterynarzy).

Minimalna cena wizyty wynosi 50 zł. Cena finalna wizyty jest zależna od wykonanych usług oraz podanych dawek leków w czasie wykonywania usługi.

Specjalizacja weterynarzy pozwala na leczenie wyłącznie:

Kotów:

- posiadających informację czy są wychodzące (poza dom / mieszkanie).

Psów:

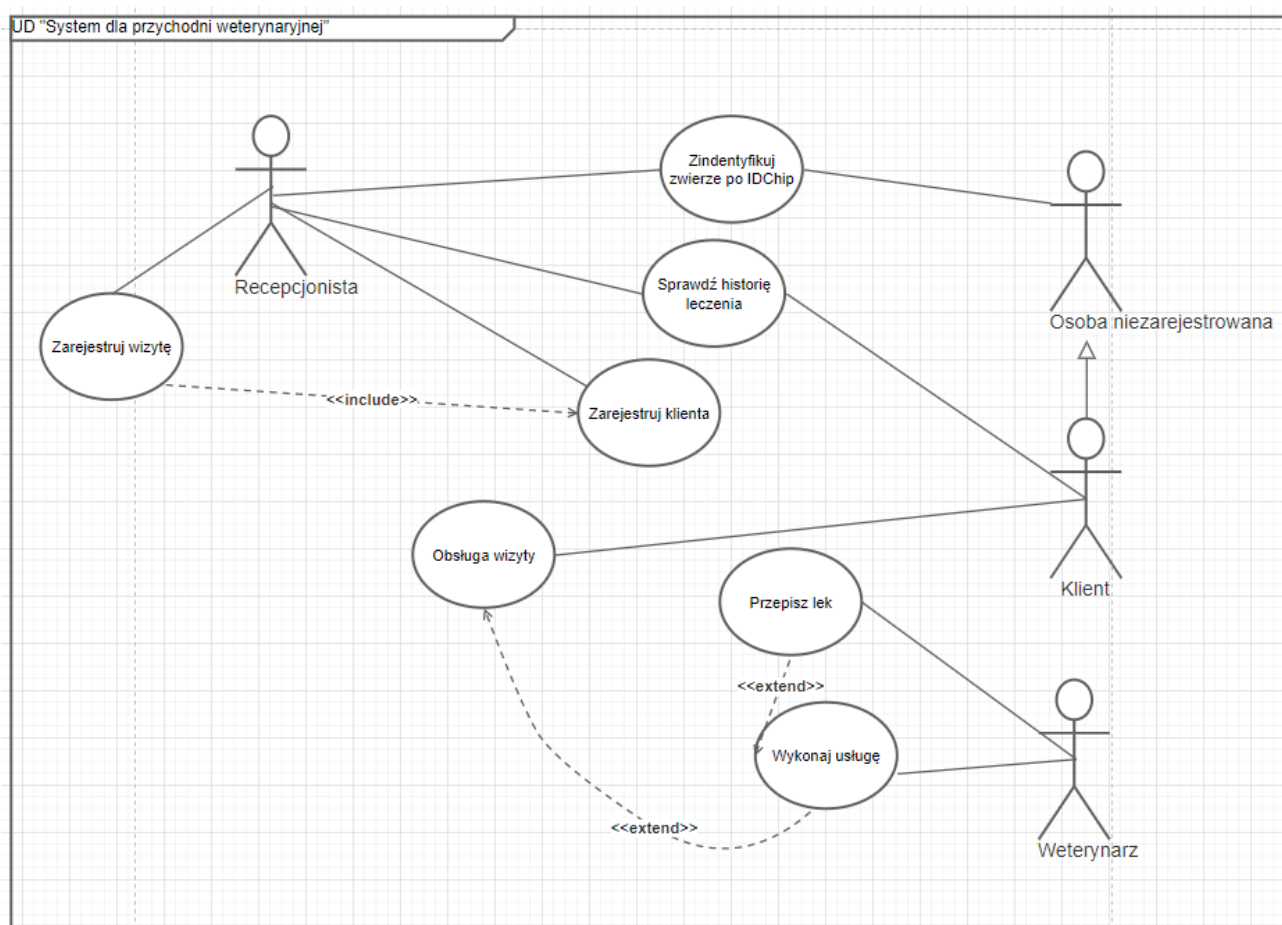
- posiadających informację dotyczącą specjalistycznego szkolenia (policyjne, osmologiczne, ratownicze, lawinowe).

Oczekuje się, że system będzie wspomagał użytkowników w sprawnym rejestrowaniu nowych klientów, umawianiu wizyt obecnie istniejącym. Dzięki niemu będzie możliwy wgląd w historię leczenia danego zwierzęcia danego klienta, jak również identyfikacja zwierzęcia jeżeli został zarejestrowany w systemie.

System powinien spełniać następujące wymagania:

- komputer z wersją systemu windows >7.

## 6. Wymagania funkcjonalne:



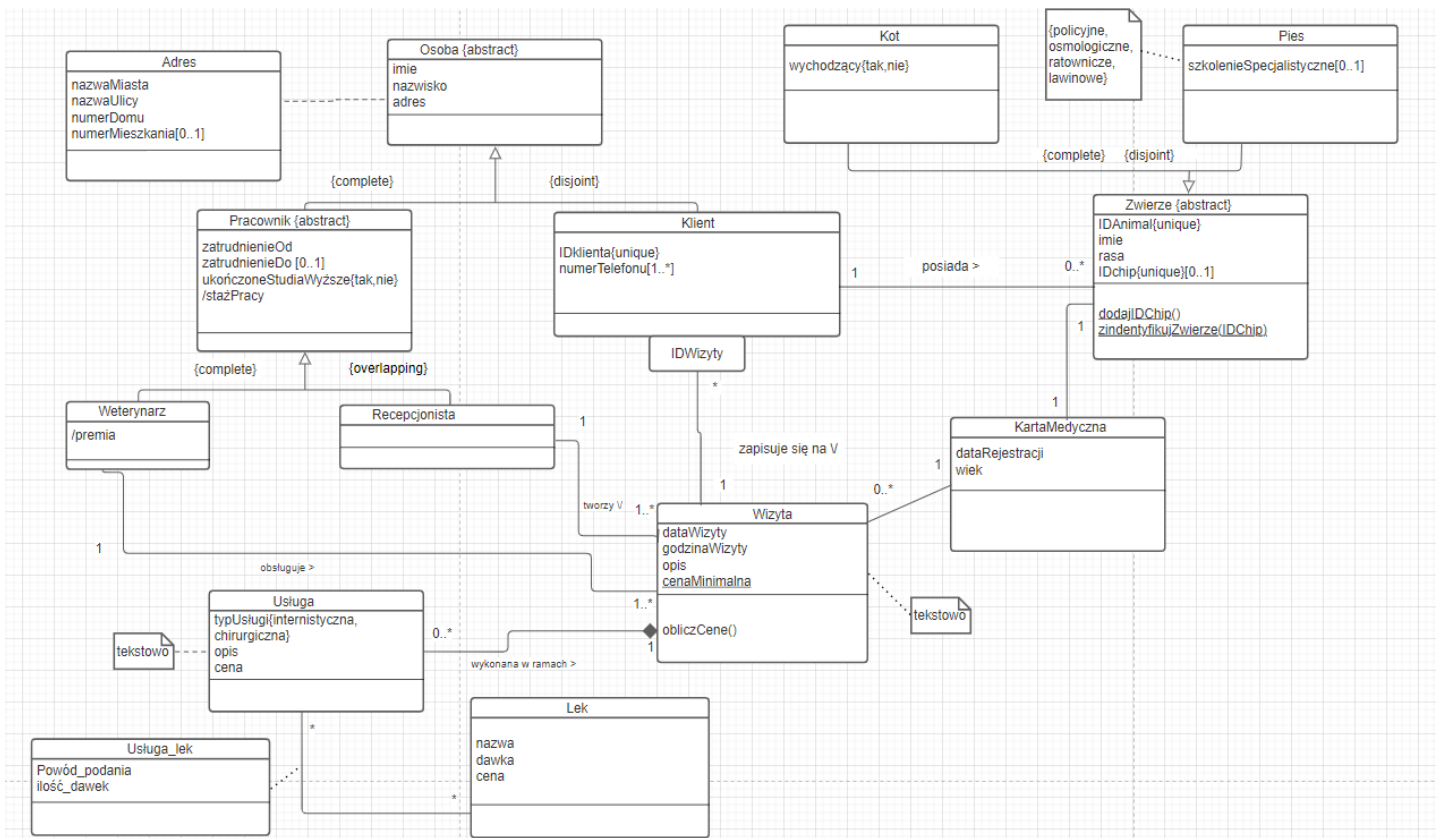
Nazwa przypadku użycia	Zidentyfikuj zwierzę po IDChip
Warunek początkowy	Osoba zarejestrowana / niezarejestrowana znalazła zwierzę i prosi o sprawdzenie możliwości identyfikacji
Główny przepływ zdarzeń	1. Aktor Recepcjonista uruchamia przypadek użycia. 2. System prosi o wpisanie ID chip 3. Aktor uzyskuje id chip od osoby i wpisuje w odpowiednie okno 4. System zwraca kartę medyczną zwierzęcia wraz z id właściciela.
Alternatywne przepływ zdarzeń	4a. System wyświetla komunikat o tym, że podany ID nie istnieje 4b. System wyświetla komunikat o tym, że podany ID jest błędny.
Zakończenie	W dowolnym momencie
Warunek końcowy	brak

Nazwa przypadku użycia	Zarejestruj wizytę
Warunek początkowy	Klient jest zarejestrowany
Główny przepływ zdarzeń	1. Aktor Recepcjonista uruchamia przypadek użycia. 2. Aktor wybiera ID klienta 3. Aktor wybiera datę i godzinę wizyty, 4. Aktor wybiera zwierzę klienta, którego dotyczy wizyta 5. Aktor wypełnia opisu problemu (informacje od klienta) 6. Aktor klika przycisk 'znajdź weterynarza' - system przypisuje pierwszego weterynarza, który jest dostępny w podanym terminie 7. Aktor recepcjonista klika przycisk 'zapisz'
Alternatywne przepływ zdarzeń	3a. System informuje o istniejącej wizycie u wybranego klienta w podanej dacie/godzinie 6a. System informuje o braku dostępnego weterynarza w podanym terminie
Zakończenie	Gdy wizyta zostaje zapisana / proces anulowany
Warunek końcowy	brak

Nazwa przypadku użycia	Zarejestruj klienta
Warunek początkowy	Klient niezarejestrowany chce umówić wizytę / zarejestrować się
Główny przepływ zdarzeń	1. Aktor Recepcjonista uruchamia przypadek użycia. 2. System prosi o wpisanie danych klienta (imie, nazwisko, adres, numer telefonu) 3. System prosi o zaakceptowanie wprowadzonych wartości 4. System generuje unikalne id klienta
Alternatywne przepływ zdarzeń	2a. System informuje o wprowadzeniu błędnego numeru telefonu 4a. System wyświetla komunikat o tym, że podane dane są błędne
Zakończenie	W dowolnym momencie
Warunek końcowy	Sukces: System zwraca id klienta

Nazwa przypadku użycia	Sprawdź historię leczenia
Warunek początkowy	Klient chce sprawdzić historię leczenia zwierzęcia
Główny przepływ zdarzeń	1. Aktor Recepcjonista uruchamia przypadek użycia. 2. System prosi o podanie id klienta 3. System wyświetla dostępne karty medyczne 4. Aktor recepcjonista wybiera kartę medyczną do wglądu
Alternatywne przepływ zdarzeń	3a. System informuje o braku kart medycznych
Zakończenie	W dowolnym momencie
Warunek końcowy	brak

## 7. Opis struktury systemu (schemat pojęciowy):



## 8. Wymagania niefunkcjonalne:

- Komputer PC lub laptop z systemem Windows ver >7

## 9. Opis przyszłej ewolucji systemu:

System w przyszłości może zostać rozbudowany o możliwość leczenia innych gatunków zwierząt, jednakże będzie to związane z zatrudnieniem dodatkowego personelu medycznego z odpowiednimi kwalifikacjami bądź przeszkoleniem obecnie pracujących weterynarzy.