Obraz zawierający tekst

Opis wygenerowany automatycznie

**Wydział Informatyki**

**Katedra Baz Danych**

Bazy danych

**Aleksander Waligóra**

18547

**Analiza I implementacja aplikacji do zarządzania procesami zachodzącymi w klinice weterynaryjnej**

Praca inżynierska

Piotr Gago

Warszawa, Lipiec, 2023

# Streszczenie

W niniejszej pracy zaprezentowano projekt aplikacji działającej w przeglądarce internetowej. Celem aplikacji jest usprawnienie zarządzania kliniki weterynaryjnej i obsługi jej klientów poprzez m.in przechowywanie informacji związanych z jej pacjentami oraz tworzenie rezerwacji wizyt do kliniki. Aplikacja została napisana w języku TypeScript z użyciem technologii Express.js, React.js oraz wykorzystano system zarządzania bazą danych Microsoft SQL Server.

**Słowa kluczowe**: aplikacja, baza danych, Internet, klient, klinika, rezerwacja, serwer, weterynarz, zarządzanie, zdrowie, zwierzę.

# Przedmowa

Opisywana praca została stworzona jak praca licencjacka studenta pierwszego stopnia kierunku Informatyki i specjalizacji Bazy danych. Została ona zaprojekowana i zaimplementowana samodzielnie przez pojedynczego autora na przestrzeni kilku semestrów.

# Spis treści

[1 Streszczenie 2](#_Toc138056215)

[2 Przedmowa 3](#_Toc138056216)

[3 Spis treści 4](#_Toc138056217)

[4 Analiza 6](#_Toc138056218)

[4.1 Wstęp 6](#_Toc138056219)

[4.2 Cele i założenia 7](#_Toc138056220)

[4.3 Kontekst 8](#_Toc138056221)

[4.4 Wymagania funkcjonalne 9](#_Toc138056222)

[4.5 Aktorzy systemu 11](#_Toc138056223)

[4.6 Przypadki użycia 12](#_Toc138056224)

[4.6.1 Diagram przypadków użycia 12](#_Toc138056225)

[4.6.2 Scenariusze przypadków użycia 13](#_Toc138056226)

[4.7 Wymagania niefunkcjonalne 32](#_Toc138056227)

[4.8 Opis prototypów 33](#_Toc138056228)

[4.8.1 Strona startowa 33](#_Toc138056229)

[4.8.2 Strona profilowa 33](#_Toc138056230)

[4.8.3 Nawigacja 34](#_Toc138056231)

[4.8.4 Lista elementów 35](#_Toc138056232)

[4.8.5 Profil zwierzęcia 36](#_Toc138056233)

[4.8.6 Tworzenie rezerwacji 38](#_Toc138056234)

[4.8.7 Rejestracja wizyty 39](#_Toc138056235)

[5 Projekt 41](#_Toc138056236)

[5.1 Ogólny opis architektury 41](#_Toc138056237)

[5.2 Model C4 42](#_Toc138056238)

[5.2.1 Poziom 1 42](#_Toc138056239)

[5.2.2 Poziom 2 43](#_Toc138056240)

[5.3 Aplikacja kliencka 44](#_Toc138056241)

[5.3.1 Wykorzystane biblioteki 44](#_Toc138056242)

[5.3.2 Opis architektury 45](#_Toc138056243)

[5.4 Aplikacja serwerowa 50](#_Toc138056244)

[5.4.1 Wykorzystanie biblioteki 50](#_Toc138056245)

[5.4.2 Opis architektury 51](#_Toc138056246)

[5.5 Warstwa trwałości 57](#_Toc138056247)

[5.5.1 Baza danych 57](#_Toc138056248)

[5.5.2 Diagram encji 57](#_Toc138056249)

[5.5.3 Indeksy 64](#_Toc138056250)

[6 Testy i wdrożenia 65](#_Toc138056251)

[6.1 Testy aplikacji serwerowej 65](#_Toc138056252)

[7 Podsumowanie 66](#_Toc138056253)

[7.1 Efekty końcowe pracy 66](#_Toc138056254)

[7.2 Możliwości kierunku rozwoju aplikacji 66](#_Toc138056255)

[8 Bibliografia 67](#_Toc138056256)

[9 Spis ilustracji 69](#_Toc138056257)

[10 Spis tabel 70](#_Toc138056258)

# Analiza

## Wstęp

Człowiek pierwszy raz udomowił zwierzę ok.10 tysięcy lat temu [1]. Od tego czasu stały się istotną naszego życia, zarówno jako źródło pożywienia lub różnego rodzaju zasobów oraz jako towarzysze. Posiadanie zwierząt domowego stwarza jednak wymóg odpowiedniej ich opieki. Jednym z tych wymogów jest zapewnienie opieki weterynaryjnej.

Medycyna weterynaryjna jest prawie tak stara jak i samo udamawianie zwierząt. Pierwsze dowody procedur medycznych na zwierzętach datuje się na 3 tysiące lat przed naszą erą [2]. Pierwsza uczelnia weterynaryjna została założona w XVIII wieku we Francji, aby pomóc w zwalczeniu pomoru bydła. Niedługo później dziedzina ta otrzymała międzynarodowe uznanie [3]. W Polsce pierwsza Katedra Weterynarii została powołana w 1804 roku w Krakowie na Uniwersytecie Jagiellońskim. Współcześnie weterynaria jest szeroko rozpowszechnioną dziedziną medycyny.

W XXI wieku nadeszła ciągle postępująca epoka silnej informatyzacji wielu aspektów naszego życia, w tym w sposobie działania wielu biznesów. Większość przedsiębiorstw, w ramach swojej organizacji, zaczęła stosować systemy informatyczne. W dużym stopniu ta transformacja usprawniła szybkość, zwiększyła wydajność oraz uprościła komunikację między systemami przedsiębiorstw, a ich pracownikami i klientami.

Jednym z beneficjentów tychże transformacji zostały też usługi weterynaryjne. Poza wspomnianymi usprawnieniami wiążącymi się z komputeryzacją, znacząco zyskali także klienci, którzy dzięki usprawnionej organizacji mogą częściej i sprawniej korzystać z opieki dostarczonej przez kliniki weterynaryjne.

## Cele i założenia

Celem aplikacji jest wsparcie poszczególnych aspektów w procesie dostarczania opieki zdrowotnej zwierzętom domowym przez kliniki weterynaryjne poprzez implementację aplikacji webowej.

Do wspieranych procesów podchodzą między innymi wprowadzanie, magazynowanie i pozyskiwanie danych związanymi z obsługą pacjentów kliniki, takich jak prowadzenie profilu pacjenta włączenie z przechowywaniem wszelkimi informacji zdrowotnych, rejestracja odbytych wizyt i zabiegów w systemie kliniki. Dodatkowo zostanie wsparta organizacja rezerwacji wizyt i zabiegów

Poprzez wykorzystanie do tego celu aplikacji internetowej, pozwoli to na możliwość przekazanie części odpowiedzialności klinki, samym klientom. Rezultatem tego, byłoby pozwolenie weterynarzom na skupieniu się nad wykonywaniem głównego zadania ich zawodu, czyli udzielania opieki medycznej pacjentom.

## Kontekst

Zdecydowałem się na obranie opisywanej problematyki z szeregu powodów, związanych zarówno z kliniką weterynaryjną jak i aplikacjami webowymi.

Coraz bardziej rośnie popularność wszelkiego rodzaju aplikacji, w tym aplikacji webowych. Wynika to z posiadanych przez nie szeregu zalet. Do ich eksploatacji wymagany jest jedynie Internet. Jest to zaletą z dwóch względów. Po pierwsze nie jest wymagana instalacja odpowiedniego oprogramowania, więc aplikacja może być eksploatowana na dowolnym komputerze, o ile jest podłączony do sieci. Druga zaletą jest powszechność samego Internetu, co za tym idzie szeroki dostęp do aplikacji. Zgodnie z raportem z portalu *wirtualnemedia.pl*, 92 % gospodarstw domowych w Polsce ma dostęp do Internetu [4], co naturalnie daje szeroki dostęp do takiejże aplikacji. Z tych powodów coraz więcej różnych przedsiębiorstw decyduje na własne ich implementacje, które wesprą zarówno pracowników jak i klientów. Jednym z typów przedsiębiorstw, które by mogły skorzystać na posiadaniu takiej aplikacji są kliniki weterynaryjne.

Zgodnie z raportem FEDIAF o zwierzętach domowych z 2020r pand 45 % gospodarstw na terenie Polski posiada zwierzę domowe. [5] Taka ewenement stwarza wysokie zapotrzebowanie na wszelkie usługi związanie z ich, że utrzymaniem, w tym zapewnienie im opieki medycznej.

Aplikacje webowe przeznaczone dla klinik weterynaryjnych nie są jeszcze tak popularne w porównaniu do ich odpowiedników przeznaczonych dla ludzi. Istnieją za to takie aplikacje, które swoim działaniem mogłyby by takową przypominać. Taka aplikacja swoim funkcjonowaniem pokrywałaby się z przeróżnymi aplikacjami dającymi użytkownikowi możliwość ręcznej rezerwacji. Przykładem może aplikacja webowa <https://www.halodoctor.pl/> która pozwala klientom oraz samym lekarzom na rejestrację oraz samodzielną rezerwację na określoną godzinę i wybranego przez siebie lekarza. Zaletą tej aplikacji jest również możliwość umawiania się do określonego typu weterynarza przez klienta oraz przeglądanie odbytych wizyt. Wadami są za to braki wirtualnej historii chorób i książeczki zdrowotnej. Dodatkowo opisywane funkcje są dostępne jedynie dla klientów, a nie również dla pracowników kliniki.

## Wymagania funkcjonalne

Aplikacja posiada dwa główne tryby: niezalogowany i zalogowany. W trybie niezalogowanym użytkownik ma możliwość zalogowania się do swojego konta kliniki, zarejestrowanie się w systemie jako klient oraz przejrzenia godzin otwarcia kliniki.

Po zalogowaniu użytkownikowi udostępniony jest szereg funkcjonalności, podzielonych ze względu na następujące typy użytkownika: właściciela, weterynarza i zarządcę, gdzie zarządca jest weterynarzem z nieograniczonym dostępem do wszelkich funkcjonalności w systemie. Każdy zalogowany użytkownik posiada własny profil. Dodatkowo weterynarz posiada tygodniowy harmonogram. Zarządca ma możliwość edytowania profili weterynarzy.

Zalogowany użytkownik mają dostęp do modułu zarządzania zwierzętami, w którego skład wchodzi rejestracja zwierząt oraz przeglądanie zarejestrowanych. Właściciel ma dostęp swoich zwierzęta, a weterynarza do wszystkie w bazie danych. Weterynarz może również wyszukać zwierzę po kliencie identyfikowanego po jego adresie email. Profil zwierzęcia składa się z 4 zakładek. Profil zwierząt zawiera ich podstawowe informacje, takie jak imię, data urodzenia czy rasa. Każdy użytkownik ma możliwość edycji tych danych. Zakładka zdrowie zawiera bieżące informacje zdrowotne zwierzęcia pogrupowanie ze względu na kategorię zdrowotną, którą weterynarz może edytować. Zakładka choroby przedstawia tabelę historię chorób zwierzęcia zarejestrowanych w klinice. Posiada ona rubryki z datą wykrycia choroby oraz ich wyleczenia. Zakładka szczepienia posiada dwie listy: listę odbytych oraz wymaganych szczepień.

Moduł zarządzania rezerwacjami składa się listy rezerwacji oraz ich tworzenia. Każdy użytkownik posiada możliwość zarejestrowania rezerwacji na określony termin. Właściciel widzi listę rezerwacji stworzonych przez siebie, weterynarz listę rezerwacji zarepetowanych do niego, a zarządca listę wszystkich rezerwacji w systemie. Z listy użytkownicy mają możliwość jej anulowania. Dodatkowo weterynarz ma możliwość realizacji rejestracja, gdzie weterynarz jest przekierowywany na stronę z rejestracją wizyty z odpowiednio częściowo wypełnionym formularzem.

Wewnątrz modułu zarządzania wizytami użytkownik klient ma możliwość przeglądania odbytych przez siebie wizyty. Lista wizyty udostępniona weterynarzowi poszerzona jest o możliwość ich filtrowania po zwierzęciu, weterynarzu i dacie. Formularz rejestracji wizyty składa się z szeregu składowych. Poza podstawowymi informacjami, formularz posiada listę z czynnościami medycznymi. Każda z nich kosztuje określoną cenę, które sumują się na opłatę za wizytę. Również dostępna jest lista dostępnych szczepień personalizowana na podstawie historii szczepieni zwierzęcia. Dodatkowo jest możliwość zarejestrowania listy chorób zdiagnozowanych podczas rejestrowanej wizyty.

Moduł zarządzania zabiegami działa podobnie do modułu wizyt. Właściciel ma możliwość przeglądania listy zabiegów dokonanych na swoich zwierzętach oraz weterynarz może je filtrować po zwierzęciu, weterynarzu i dacie. Po zarejestrowanie zabiegu rejestrujący ma możliwość jego usunięcie lub złożenie raportu.

Użytkownik zarządca posiada moduł zarządzania weterynarzami w systemie, w skład którego wchodzi przeglądanie i edytowanie profili weterynarzy i ich rejestracja.

Również zarządca ma dostęp do modułu innych, który pozwala mu na zarejestrowanie i usunięcie szczepionek oraz czynności medycznych do systemu.

## Aktorzy systemu

W aplikacji wyróżniono cztery typy aktorów:

* **Klient** /**Właściciel** – aktor, który ma dostęp do niewielkiej ilości funkcji, takich jak przeglądanie danych, założenie konta, rejestracja zwierzęcia i zarezerwowanie wizyt.
* **Weterynarz** – aktor, któremu jest udostępniona większość funkcjonalności aplikacji. Posiada wszystkie funkcje aktora Klient. Ponadto ma dostęp do rejestracji wizyt, zabiegów oraz wszelkich aspektów z nimi związanymi.
* **Zarządca** – aktor mający nieograniczony dostęp do wszystkich funkcji systemu.
* **Użytkownik niezalogowany** – aktor niebędący zalogowany do aplikacji.

## Przypadki użycia

### Diagram przypadków użycia



### Scenariusze przypadków użycia

|  |  |
| --- | --- |
| Lp. | PU-1 |
| Nazwa: | Zalogowanie się |
| Aktorzy: | Użytkownik niezalogowany |
| Warunki początkowe | Aktor jest na stronie startowej |
| Opis | Aktor chce się zalogować. |
| Ścieżka główna: | 1. Aktor wypełnia formularz logowania. Wymagane pola to „Email”, „Hasło”. 2. Aktor zatwierdza logowanie poprzez kliknięcie przycisku „Zaloguj”. 3. Aktor zostaje przekierowany na stronę z profilem użytkownika. |
| Ścieżka alternatywna | 3a. Aktor zostaje powiadomiony o niepoprawnych danych logowania. |
| Warunki końcowe | Aktor znajduje się na stronie z profilem użytkownika |

Tabela 1 PU-1

|  |  |
| --- | --- |
| Lp. | PU-2 |
| Nazwa: | Zarejestrowanie użytkownika |
| Aktorzy: | Użytkownik niezalogowany |
| Warunki początkowe | Aktor jest na stronie startowej |
| Opis | Aktor chce się zarejestrować się w systemie kliniki weterynaryjnej. |
| Ścieżka główna: | 1. Aktor wybiera opcję „Zarejestruj” z paska nawigacyjnego. 2. Aktor wypełnia formularz rejestracji użytkownika. Wymagane pola to „Email”, „Imię”, „Nazwisko” , „Kontakt”, „Hasło”, „Powtórz hasło”. 3. Aktor zatwierdza rejestrację poprzez kliknięcie przycisku „Zarejestruj”. 4. Aktor zostaje przekierowany na stronę startową na zakładce „Logowanie”. |
| Ścieżka alternatywna | 4a. Aktor zostaje powiadomiony o błędach w polach formularzu rejestracji. |
| Warunki końcowe | Aktor znajduje się na stronie startowej na zakładce „Logowanie” |

Tabela 2 PU-2

|  |  |
| --- | --- |
| Lp. | PU-3 |
| Nazwa: | Wyświetlenie godzin otwarcia kliniki |
| Aktorzy: | Użytkownik niezalogowany |
| Warunki początkowe | Aktor jest na stronie startowej |
| Opis | Aktor chce wyświetlić godziny otwarcia kliniki. |
| Ścieżka główna: | 1. Aktor klika na opcję „Godziny otwarcia” z paska nawigacyjnego. 2. Aktorowi zostają wyświetlone godziny otwarcia. |
| Ścieżka alternatywna |  |
| Warunki końcowe | Aktorowi zostają wyświetlone godziny otwarcia kliniki |

Tabela 3 PU-3

|  |  |
| --- | --- |
| Lp. | PU-4 |
| Nazwa: | Przeglądanie profilu użytkownika |
| Aktorzy: | Właściciel, Weterynarz, Zarządca |
| Warunki początkowe | Aktor jest zalogowany |
| Opis | Aktor chce przejrzeć swój profil. |
| Ścieżka główna: | 1. Aktor klika na opcje „Profil” na bocznym pasku nawigacyjnym 2. Aktor zostaje przekierowany na stronę z profilem użytkownika. |
| Ścieżka alternatywna |  |
| Warunki końcowe | Aktor znajduje się na stronie z profilem użytkownika. |

Tabela 4 PU-4

|  |  |
| --- | --- |
| Lp. | PU-5 |
| Nazwa: | Zarejestrowanie zwierzęcia |
| Aktorzy: | Właściciel, Weterynarz, Zarządca |
| Warunki początkowe | Aktor jest zalogowany |
| Opis | Aktor rejestruje zwierzę w systemie. |
| Ścieżka główna: | 1. Aktor klika na opcje „Zwierzęta” na bocznym pasku nawigacyjnym. 2. Aktor klika na opcje „Rejestracja” z rozwiniętego menu. 3. Aktor wypełnia formularz rejestracji zwierzęcia. Wymagane pola to „Imię”, „Właściciel”, „Rodzaj”, „Data urodzenia”, „Płeć”. Opcjonalne pola to „Zdjęcie profilowe” 4. Aktor zatwierdza rejestrację poprzez kliknięcie przycisku „Zarejestruj”. 5. Aktor zostaje przekierowany na z listą zwierząt z komunikatem o udanej rejestracji. |
| Ścieżka alternatywna | 5a. Aktor zostaje powiadomiony o błędach przy wypełnianiu formularzy rejestracji. |
| Warunki końcowe | Aktor znajduje się na stronie z listą zwierząt z komunikatem o udanej rejestracji. |

Tabela 5 PU-5

|  |  |
| --- | --- |
| Lp. | PU-6 |
| Nazwa: | Przeglądanie listy zwierząt |
| Aktorzy: | Właściciel, Weterynarz, Zarządca |
| Warunki początkowe | Aktor jest zalogowany |
| Opis | Aktor przegląda listę zwierząt. |
| Ścieżka główna: | 1. Aktor klika na opcje „Zwierzęta” na bocznym pasku nawigacyjnym. 2. Aktor wybiera opcje „Lista” z rozwiniętego menu. 3. Aktor zostaje przekierowany na stronę z listą zwierząt. |
| Ścieżka alternatywna |  |
| Warunki końcowe | Aktor znajduje się na stronie z listą zwierząt |

Tabela 6 PU-6

|  |  |
| --- | --- |
| Lp. | PU-7 |
| Nazwa: | Filtrowanie wyników |
| Aktorzy: | Weterynarz, Zarządca |
| Warunki początkowe | Aktor jest na stronie z listą zwierząt |
| Opis | Aktor filtruje listę zwierząt. |
| Ścieżka główna: | 1. Aktor wpisuje wartości w pola flirtów. 2. Aktor wciska przycisk „Szukaj”. 3. Aktorowi ukazuje się przefiltrowana lista zwierząt. |
| Ścieżka alternatywna |  |
| Warunki końcowe | Aktorowi pokazana jest przefiltrowana lista zwierząt |

Tabela 7 PU-7

|  |  |
| --- | --- |
| Lp. | PU-8 |
| Nazwa: | Przeglądanie profilu zwierzęcia |
| Aktorzy: | Właściciel, Weterynarz, Zarządca |
| Warunki początkowe | Aktor znajduje się na stronie z listą zwierząt |
| Opis | Aktor przegląda profil zwierzęcia. |
| Ścieżka główna: | 1. Aktor klika na wiersz z listy zwierząt. 2. Aktor zostaje przekierowany na stronę profilu zwierzęcia. |
| Ścieżka alternatywna |  |
| Warunki końcowe | Aktor znajduje się na stronie z profilem zwierzęcia |

Tabela 8 PU-8

|  |  |
| --- | --- |
| Lp. | PU-9 |
| Nazwa: | Edytowanie profilu zwierzęcia |
| Aktorzy: | Właściciel, Weterynarz, Zarządca |
| Warunki początkowe | Aktor znajduje się na stronie z profilem zwierzęcia |
| Opis | Aktor edytuje profil zwierzęcia. |
| Ścieżka główna: | 1. Aktor klika na przycisk „Edytuj”. 2. Aktor zostaje przekierowany na stronę edycji profilu zwierzęcia. 3. Aktor edytuje wybrane pola. Edytowalne pola to „Imię”, „Właściciel”, „Rodzaj”, „Data urodzenia”, „Płeć”, „Zdjęcie profilowe”. 4. Aktor zatwierdza zmiany przez kliknięcie na przycisk „Zapisz”. 5. Aktor zostaje przekierowany na stronę z profilem zwierzęcia. |
| Ścieżka alternatywna | 5a. Aktor zostaje powiadomiony o błędach w polach formularzu edycji zwierzęcia. |
| Warunki końcowe | Zmiany zostały zapisane w systemie. |

Tabela 9 PU-9

|  |  |
| --- | --- |
| Lp. | PU-10 |
| Nazwa: | Edytowanie karty zdrowia zwierzęcia |
| Aktorzy: | Weterynarz, Zarządca |
| Warunki początkowe | Aktor znajduje się na stronie z profilem zwierzęcia |
| Opis | Aktor edytuje kartę medyczną zwierzęcia. |
| Ścieżka główna: | 1. Aktor klika na przycisk „Informacje medyczne” na górnym pasku nawigacyjnym profilu. 2. Aktor klika przycisk „Edytuj”. 3. Aktor edytuje wybrane informacje medyczne. Dostępne zakładki to „Ogólne”, „Układ mięśniowy”, „Układ nerwowy”, „Układ szkieletowy”, „Układ hormonalny”, Układ krążeniowy, „Układ limfatyczny”, „Układ rozrodczy”, „Układ oddechowy”, „Układ trawienny”, „Układ wydalniczy”, „Układ optyczny, „Dermatologia” i „Inne”. W zakładce „Ogólne” dostępne pola to „Waga”,” Chip”, „Sterylizacja” 4. Aktor zatwierdza zmiany poprzez kliknięcie na przycisk „Zapisz”. 5. Aktor zostaje przekierowany na stronę z profilem zwierzęcia. |
| Ścieżka alternatywna |  |
| Warunki końcowe | Zmiany zostały zapisane w systemie. |

Tabela 10 PU-10

|  |  |
| --- | --- |
| Lp. | PU-11 |
| Nazwa: | Potwierdzenie wyleczenia dolegliwości |
| Aktorzy: | Weterynarz, Zarządca |
| Warunki początkowe | Aktor znajduje się na stronie z profilem zwierzęcia |
| Opis | Aktor potwierdza wyleczenie dolegliwości. |
| Ścieżka główna: | 1. Aktor klika na przycisk „Choroby” na górnym pasku nawigacyjnym profilu. 2. Aktor klika przycisk „Niewyleczone” na liście. 3. Na wierszu pojawia się informacja „Wyleczone” z datą wyleczenia. |
| Ścieżka alternatywna |  |
| Warunki końcowe | Zmiany zostały zapisane w systemie. |

Tabela 11 PU-11

|  |  |
| --- | --- |
| Lp. | PU-12 |
| Nazwa: | Przeglądanie listy rezerwacji |
| Aktorzy: | Właściciel, Weterynarz, Zarządca |
| Warunki początkowe | Aktor jest zalogowany |
| Opis | Aktor przegląda listę rezerwacji. |
| Ścieżka główna: | 1. Aktor klika na opcje „Rezerwacje” na bocznym pasku nawigacyjnym. 2. Aktor wybiera opcje „Lista” z rozwiniętego menu. 3. Aktor zostaje przekierowany na stronę z listą rezerwacji. |
| Ścieżka alternatywna |  |
| Warunki końcowe | Aktor znajduje się na stronie z listą rezerwacji |

Tabela 12 PU-12

|  |  |
| --- | --- |
| Lp. | PU-13 |
| Nazwa: | Stworzenie rezerwacji. |
| Aktorzy: | Właściciel, Weterynarz, Zarządca |
| Warunki początkowe | Aktor jest zalogowany |
| Opis | Aktor tworzy rezerwacje w systemie. |
| Ścieżka główna: | 1. Aktor klika na opcje „Rezerwacje” na bocznym pasku nawigacyjnym. 2. Aktor klika na opcje „Rejestracja” z rozwiniętego menu. 3. Aktor wypełnia formularz rezerwacji. Wymagane pola to „Data”, „Weterynarz”, „Godzina”. Aktor Weterynarz ma dodatkowe pole „Właściciel”. 4. Aktor zatwierdza rejestrację poprzez kliknięcie przycisku „Zarejestruj”. 5. Aktor zostaje przekierowany na z listą rezerwacji z komunikatem o udanej rejestracji. |
| Ścieżka alternatywna | 5a. Aktor zostaje powiadomiony o błędach w polach formularzu rejestracji rezerwacji. |
| Warunki końcowe | Aktor znajduje się na stronie z listą rezerwacji z komunikatem o udanej rejestracji. |

Tabela 13 PU-13

|  |  |
| --- | --- |
| Lp. | PU-14 |
| Nazwa: | Anuluj rezerwacji. |
| Aktorzy: | Właściciel, Weterynarz, Zarządca |
| Warunki początkowe | Aktor jest na stronie z listą rezerwacji |
| Opis | Aktor anuluje rezerwacje. |
| Ścieżka główna: | 1. Aktor klika na opcje „Anuluj” na liście rezerwacji. 2. Aktor klika na opcje „Potwierdź” na medalu potwierdzenia. |
| Ścieżka alternatywna |  |
| Warunki końcowe | Rezerwacja zostaje usunięta z systemu |

Tabela 14 PU-14

|  |  |
| --- | --- |
| Lp. | PU-15 |
| Nazwa: | Przeglądanie listy odbytych wizyt |
| Aktorzy: | Właściciel, Weterynarz, Zarządca |
| Warunki początkowe | Aktor jest zalogowany |
| Opis | Aktor przegląda listę wizyt |
| Ścieżka główna: | 1. Aktor klika na opcje „Wizyty” na bocznym pasku nawigacyjnym. 2. Aktor wybiera opcje „Lista” z rozwiniętego menu. 3. Aktor zostaje przekierowany na stronę z listą wizyt. |
| Ścieżka alternatywna |  |
| Warunki końcowe | Aktor znajduje się na stronie z listą wizyt |

Tabela 15 PU-15

|  |  |
| --- | --- |
| Lp. | PU-16 |
| Nazwa: | Zarejestrowanie wizyty. |
| Aktorzy: | Weterynarz, Zarządca |
| Warunki początkowe | Aktor jest zalogowany |
| Opis | Aktor tworzy nową wizytę w systemie. |
| Ścieżka główna: | 1. Aktor klika na opcje „Wizyty” na bocznym pasku nawigacyjnym. 2. Aktor klika na opcje „Rejestracja” z rozwiniętego menu. 3. Aktor wypełnia formularz wizyty. Wymagane pola to „Właściciel”, „Zwierzę”, „Data”, „Godzina”. Opcjonalne pola to „Notatka”, „Czynności”, „Szczepienia”, „Diagnozy”. 4. Aktor zatwierdza rejestrację poprzez kliknięcie przycisku „Zapisz”. 5. Aktor zostaje przekierowany na z listą wizyt z komunikatem o udanej rejestracji. |
| Ścieżka alternatywna | 5a. Aktor zostaje powiadomiony o błędach w polach formularzu wizyty. |
| Warunki końcowe | Aktor znajduje się na stronie z listą wizyt z komunikatem o udanej rejestracji. |

Tabela 16 PU-16

|  |  |
| --- | --- |
| Lp. | PU-17 |
| Nazwa: | Filtrowanie wyników |
| Aktorzy: | Właściciel, Weterynarz, Zarządca |
| Warunki początkowe | Aktor jest na stronie z listą wizyt |
| Opis | Aktor filtruje listę wizyt. |
| Ścieżka główna: | 1. Aktor wpisuje wartości w pola flirtów. 2. Aktor wciska przycisk „Szukaj”. 3. Aktorowi ukazuje się przefiltrowana lista wizyt. |
| Ścieżka alternatywna |  |
| Warunki końcowe | Aktorowi pokazana jest przefiltrowana lista wizyt |

Tabela 17 PU-17

|  |  |
| --- | --- |
| Lp. | PU-18 |
| Nazwa: | Przeglądanie profilu wizyty |
| Aktorzy: | Właściciel, Weterynarz, Zarządca |
| Warunki początkowe | Aktor znajduje się na stronie z listą wizyt |
| Opis | Aktor przegląda profil wizyty. |
| Ścieżka główna: | 1. Aktor klika na wiersz z listy wizyt. 2. Aktor zostaje przekierowany na stronę profilu wizy. |
| Ścieżka alternatywna |  |
| Warunki końcowe | Aktor znajduje się na stronie z profilem wizyty |

Tabela 18 PU-18

|  |  |
| --- | --- |
| Lp. | PU-19 |
| Nazwa: | Przeglądanie listy zabiegów |
| Aktorzy: | Właściciel, Weterynarz, Zarządca |
| Warunki początkowe | Aktor jest zalogowany |
| Opis | Aktor przegląda listę zabiegów. |
| Ścieżka główna: | 1. Aktor klika na opcje „Zabiegi” na bocznym pasku nawigacyjnym. 2. Aktor wybiera opcje „Lista” z rozwiniętego menu. 3. Aktor zostaje przekierowany na stronę z listą zabiegów. |
| Ścieżka alternatywna |  |
| Warunki końcowe | Aktor znajduje się na stronie z listą zabiegów. |

Tabela 19 PU-19

|  |  |
| --- | --- |
| Lp. | PU-20 |
| Nazwa: | Filtrowanie wyników |
| Aktorzy: | Właściciel, Weterynarz, Zarządca |
| Warunki początkowe | Aktor jest na stronie z listą zabiegów |
| Opis | Aktor filtruje listę wizyt. |
| Ścieżka główna: | 1. Aktor wpisuje wartości w pola flirtów. 2. Aktor wciska przycisk „Szukaj”. 3. Aktorowi ukazuje się przefiltrowana lista zabiegów. |
| Ścieżka alternatywna |  |
| Warunki końcowe | Aktorowi pokazana jest przefiltrowana lista zabiegów |

Tabela 20 PU-20

|  |  |
| --- | --- |
| Lp. | PU-21 |
| Nazwa: | Przeglądanie profilu zabiegu |
| Aktorzy: | Właściciel, Weterynarz, Zarządca |
| Warunki początkowe | Aktor znajduje się na stronie z listą zabiegów |
| Opis | Aktor przegląda profil zabiegu. |
| Ścieżka główna: | 1. Aktor klika na wiersz z listy zabiegów. 2. Aktor zostaje przekierowany na stronę profilu zabiegu. |
| Ścieżka alternatywna |  |
| Warunki końcowe | Aktor znajduje się na stronie z profilem zabiegu |

Tabela 21 PU-21

|  |  |
| --- | --- |
| Lp. | PU-22 |
| Nazwa: | Stworzenie zabiegu. |
| Aktorzy: | Weterynarz, Zarządca |
| Warunki początkowe | Aktor jest zalogowany |
| Opis | Aktor tworzy nową zabieg w systemie. |
| Ścieżka główna: | 1. Aktor klika na opcje „Zabiegi” na bocznym pasku nawigacyjnym. 2. Aktor klika na opcje „Rejestracja” z rozwiniętego menu. 3. Aktor wypełnia formularz zabiegu. Wymagane pola to „Właściciel”, „Zwierzę”, „Kategoria”, „Weterynarz”, „Termin”, „Godzina”, „Opis”. 4. Aktor zatwierdza rejestrację poprzez kliknięcie przycisku „Zapisz”. 5. Aktor zostaje przekierowany na z listą zabiegu z komunikatem o udanej rejestracji. |
| Ścieżka alternatywna | 5a. Aktor zostaje powiadomiony o błędach w polach formularzu rejestracji zabiegu. |
| Warunki końcowe | Aktor znajduje się na stronie z listą zabiegów z komunikatem o udanej rejestracji. |

Tabela 22 PU-22

|  |  |
| --- | --- |
| Lp. | PU-23 |
| Nazwa: | Anulowanie zabiegu |
| Aktorzy: | Weterynarz, Zarządca |
| Warunki początkowe | Aktor jest na stronie profilu zabiegu |
| Opis | Aktor anuluje zabieg. |
| Ścieżka główna: | 1. Aktor kilka przycisk „Anuluj zabieg”. 2. Aktor klika na opcje „Potwierdź” na medalu potwierdzenia. 3. Aktor jest przekierowywany na stronę z listą zabiegów. |
| Ścieżka alternatywna |  |
| Warunki końcowe | Zabieg zostaje usunięty z systemu |

Tabela 23 PU-23

|  |  |
| --- | --- |
| Lp. | PU-24 |
| Nazwa: | Złożenie raportu |
| Aktorzy: | Weterynarz, Zarządca |
| Warunki początkowe | Aktor jest na stronie profilu zabiegu |
| Opis | Aktor składa raport z zabiegu |
| Ścieżka główna: | 1. Aktor klika przycisk „Edytuj”. 2. Aktor wypełnia pole „Raport”. 3. Aktor klika przycisk „Zapisz” w celu zapisania zmian. |
| Ścieżka alternatywna |  |
| Warunki końcowe | Raport o wizycie zostaje zapisany w systemie. |

Tabela 24 PU-24

|  |  |
| --- | --- |
| Lp. | PU-25 |
| Nazwa: | Przeglądanie listy weterynarzy |
| Aktorzy: | Zarządca |
| Warunki początkowe | Aktor jest zalogowany |
| Opis | Aktor przegląda listę weterynarzy. |
| Ścieżka główna: | 1. Aktor klika na opcje „Weterynarze” na bocznym pasku nawigacyjnym. 2. Aktor wybiera opcje „Lista” z rozwiniętego menu. 3. Aktor zostaje przekierowany na stronę z listą weterynarzy. |
| Ścieżka alternatywna |  |
| Warunki końcowe | Aktor znajduje się na stronie z listą weterynarzy. |

Tabela 25 PU-25

|  |  |
| --- | --- |
| Lp. | PU-26 |
| Nazwa: | Przeglądanie profilu weterynarza |
| Aktorzy: | Zarządca |
| Warunki początkowe | Aktor znajduje się na stronie z listą weterynarzy |
| Opis | Aktor przegląda profil weterynarza. |
| Ścieżka główna: | 1. Aktor klika na wiersz z listy weterynarza. 2. Aktor zostaje przekierowany na stronę profilu weterynarza. |
| Ścieżka alternatywna |  |
| Warunki końcowe | Aktor znajduje się na stronie z profilem weterynarza. |

Tabela 26 PU-26

|  |  |
| --- | --- |
| Lp. | PU-27 |
| Nazwa: | Edytowanie profilu weterynarza |
| Aktorzy: | Zarządca |
| Warunki początkowe | Aktor znajduje się na stronie z profilem weterynarza |
| Opis | Aktor edytuje profil weterynarza. |
| Ścieżka główna: | 1. Aktor klika na przycisk „Edytuj”. 2. Aktor zostaje przekierowany na stronę edycji profilu weterynarza. 3. Aktor edytuje wybrane pola. Edytowalne pola to „Imię”, „Nazwisko”, „Kontakt”, „Email”,,” Specjalizacje”, „Data zatrudnienia”, „Specjalizacje”. 4. Aktor zatwierdza zmiany przez kliknięcie na przycisk „Zapisz”. 5. Aktor zostaje przekierowany na stronę z profilem weterynarza. |
| Ścieżka alternatywna | 5a. Aktor zostaje powiadomiony o błędach w polach formularzu edycji weterynarza. |
| Warunki końcowe | Zmiany zostały zapisane w systemie. |

Tabela 27 PU-27

|  |  |
| --- | --- |
| Lp. | PU-28 |
| Nazwa: | Rejestracja weterynarza. |
| Aktorzy: | Zarządca |
| Warunki początkowe | Aktor jest zalogowany |
| Opis | Aktor tworzy profil weterynarza w systemie. |
| Ścieżka główna: | 1. Aktor klika na opcje „Weterynarze” na bocznym pasku nawigacyjnym. 2. Aktor klika na opcje „Rejestracja” z rozwiniętego menu. 3. Aktor wypełnia formularz rejestracji weterynarza. Wymagane pola to „Imię”, „Nazwisko”, „Kontakt”, „Email”, „Data zatrudnienia”. Pola opcjonalne to „Zdjęcie profilowe”, „Specjalizacje”. 4. Aktor zatwierdza rejestrację poprzez kliknięcie przycisku „Zarejestruj”. 5. Aktor zostaje przekierowany na z listą weterynarzy z komunikatem o udanej rejestracji. |
| Ścieżka alternatywna | 5a. Aktor zostaje powiadomiony o błędach w polach formularzu rejestracji weterynarza. |
| Warunki końcowe | Aktor znajduje się na stronie z listą weterynarzy z komunikatem o udanej rejestracji. |

Tabela 28 PU-28

|  |  |
| --- | --- |
| Lp. | PU-29 |
| Nazwa: | Dodanie czynności medycznej do systemu. |
| Aktorzy: | Zarządca |
| Warunki początkowe | Aktor jest zalogowany |
| Opis | Aktor dodaje czynność medyczną w systemie. |
| Ścieżka główna: | 1. Aktor klika na opcje „Różne” na bocznym pasku nawigacyjnym. 2. Aktor klika na opcje „Czynności” z rozwiniętego menu. 3. Aktor wypełnia formularz czynności. Wymagane pola to „Czynność” i „Cena”. 4. Aktor zatwierdza rejestrację poprzez kliknięcie przycisku „Dodaj”. |
| Ścieżka alternatywna | 4a. Aktor zostaje powiadomiony o błędach przy wypełnianiu formularzy rejestracji. |
| Warunki końcowe | Na liście aktywności ukazała się nowo dodana aktywność |

Tabela 29 PU-29

|  |  |
| --- | --- |
| Lp. | PU-30 |
| Nazwa: | Usunięcie czynności medycznej do systemu. |
| Aktorzy: | Zarządca |
| Warunki początkowe | Aktor jest zalogowany |
| Opis | Aktor tworzy rezerwacje w systemie. |
| Ścieżka główna: | 1. Aktor klika na opcje „Różne” na bocznym pasku nawigacyjnym. 2. Aktor klika na opcje „Czynności” z rozwiniętego menu. 3. Aktor klika przycisk „Usuń” przy wierszu z usuwaną czynnością. 4. Aktor klika na opcje „Potwierdź” na medalu potwierdzenia. |
| Ścieżka alternatywna |  |
| Warunki końcowe | Z listy znika usunięta aktywność. |

Tabela 30 PU-30

|  |  |
| --- | --- |
| Lp. | PU-31 |
| Nazwa: | Dodanie nowej szczepionki do systemu. |
| Aktorzy: | Zarządca |
| Warunki początkowe | Aktor jest zalogowany |
| Opis | Aktor dodaje szczepionkę do systemu. |
| Ścieżka główna: | 1. Aktor klika na opcje „Różne” na bocznym pasku nawigacyjnym. 2. Aktor klika na opcje „Szczepionki” z rozwiniętego menu. 3. Aktor wypełnia formularz szczepionki. Wymagane pola to „Nazwa”, „Rodzaj”. Opcjonalne pola to „Wymagana”. 4. Aktor zatwierdza rejestrację poprzez kliknięcie przycisku „Dodaj”. |
| Ścieżka alternatywna | 4a. Aktor zostaje powiadomiony o błędach przy wypełnianiu formularzy rejestracji. |
| Warunki końcowe | Na liście aktywności ukazała się nowo dodana szczepionka. |

Tabela 31 PU-31

|  |  |
| --- | --- |
| Lp. | PU-32 |
| Nazwa: | Usunięcie szczepionki z systemu. |
| Aktorzy: | Zarządca |
| Warunki początkowe | Aktor jest zalogowany |
| Opis | Aktor usuwa szczepionkę z systemu. |
| Ścieżka główna: | 1. Aktor klika na opcje „Różne” na bocznym pasku nawigacyjnym. 2. Aktor klika na opcje „Szczepionki” z rozwiniętego menu. 3. Aktor klika przycisk „Usuń” przy wierszu z usuwaną szczepionką. 4. Aktor klika na opcje „Potwierdź” na medalu potwierdzenia. |
| Ścieżka alternatywna |  |
| Warunki końcowe | Z listy znika usunięta szczepionka. |

Tabela 32 PU-32

## Wymagania niefunkcjonalne

|  |  |
| --- | --- |
| Cecha | Dostępność |
| Priorytet | Poszczególne funkcje systemu są dostępne dla ściśle określonych użytkowników |
| Miara |  |

|  |  |
| --- | --- |
| Cecha | Pojemność |
| Priorytet | System powinien zajmować do 100 GB pamięci |
| Miara | Gigabajty |

|  |  |
| --- | --- |
| Cecha | Wydajność |
| Priorytet | System może jednocześnie obsługiwać do 100 transakcji jednocześnie bez obciążenia |
| Miara | Liczba transakcji na sekundę |

|  |  |
| --- | --- |
| Cecha | Regulacje prawne |
| Priorytet | Dane przechowywane o weterynarzach są poddanie ochronie |
| Miara |  |

## Opis prototypów

### Strona startowa

Pierwszy widok aplikacji, przedstawia ekran niezalogowanego użytkownika (Rysunek 1). Z tego poziomu posiada on trzy opcje pokazana na zakładkach z lewej strony ekranu. Użytkownik może się zalogować, przejrzy godziny otwarcia kliniki oraz zarejestrować jako właściciel. W celu zalogowania użytkownik wypełnia pola „Email” „Hasło”. Po kliknięciu na przycisk „Zaloguj” i pomyślnej walidacji zostaje on przekierowany na swój profil.

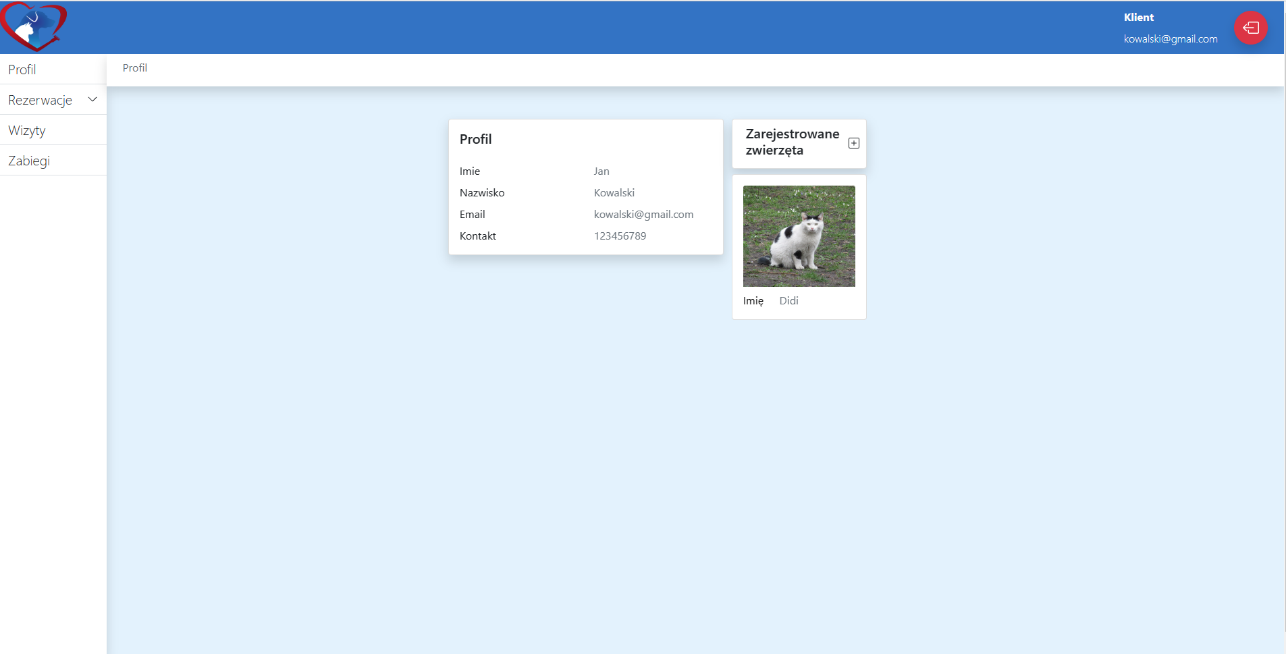
Obraz zawierający pies, zrzut ekranu

Opis wygenerowany automatycznie

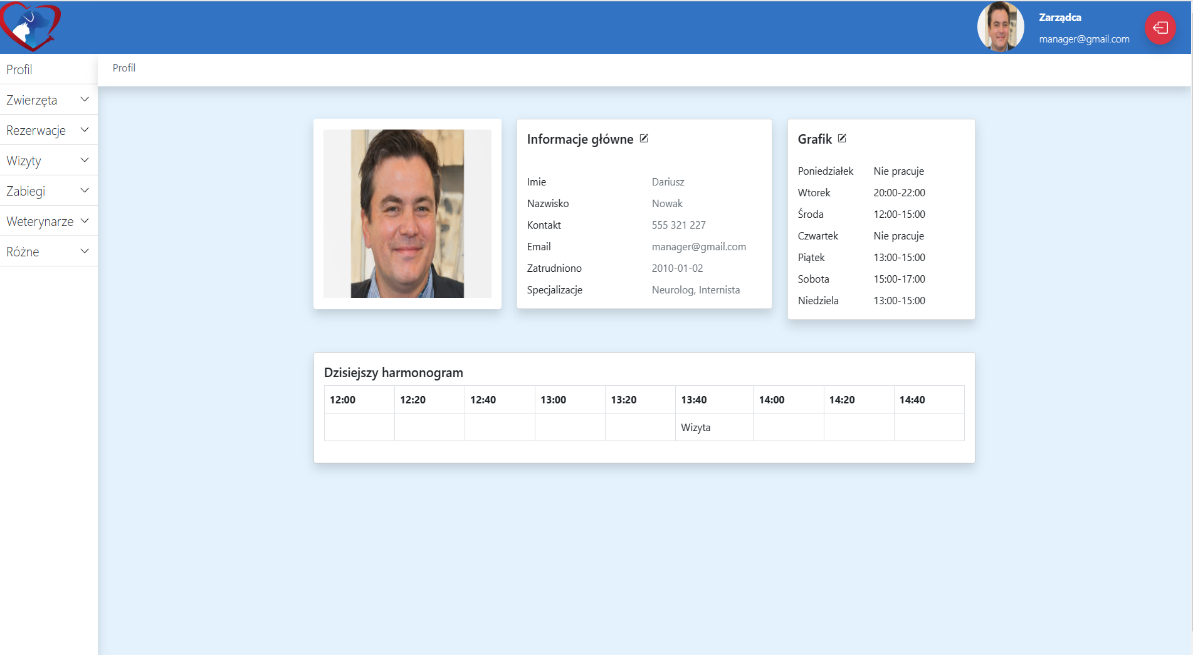
Rysunek 1 Ekran niezalogowanego użytkownika

### Strona profilowa

Strona profilowa różnią się w zależności od rodzaju użytkownika. Dla klientów wyświetlane są dane personalne oraz lista zarejestrowanych przez niego zwierząt (Rysunek 2). W przypadku gdy użytkownik jest weterynarzem wyświetlane są dane personalne, godziny pracy oraz dzisiejszy grafik (Rysunek 3).



Rysunek 2 Ekran profilu właściciela



Rysunek 3 Ekran profilu weterynarza

### Nawigacja

Użytkownika nawiguje po głównych ekranach aplikacji przy pomocy bocznego paska. Zawartość ta różnie się w zależności od typu użytkownika (Rysunek 4,5,6).

Obraz zawierający tekst

Opis wygenerowany automatycznieObraz zawierający tekst, Czcionka, zrzut ekranu, numer

Opis wygenerowany automatycznieObraz zawierający tekst

Opis wygenerowany automatycznie

Rysunek 4 Menu właściciela Rysunek 5 Menu weterynarza Rysunek 6 Menu zarządcy

W przypadku kliknięcia na element menu ze strzałką do dołu, zostanie rozwinięta lista z opcjami przejścia do listy elementów lub ich rejestracji.

### Lista elementów

Na tych ekranach są wylistowanie elementy zrejestrowanie w systemie (Rysunek7) . W zależności od elementów użytkownik ma możliwość przefiltrowania wyników po wybranych atrybutów. W przypadku kliknięcia na wiersz z listy, użytkownik zostanie przekierowany na profil elementu.

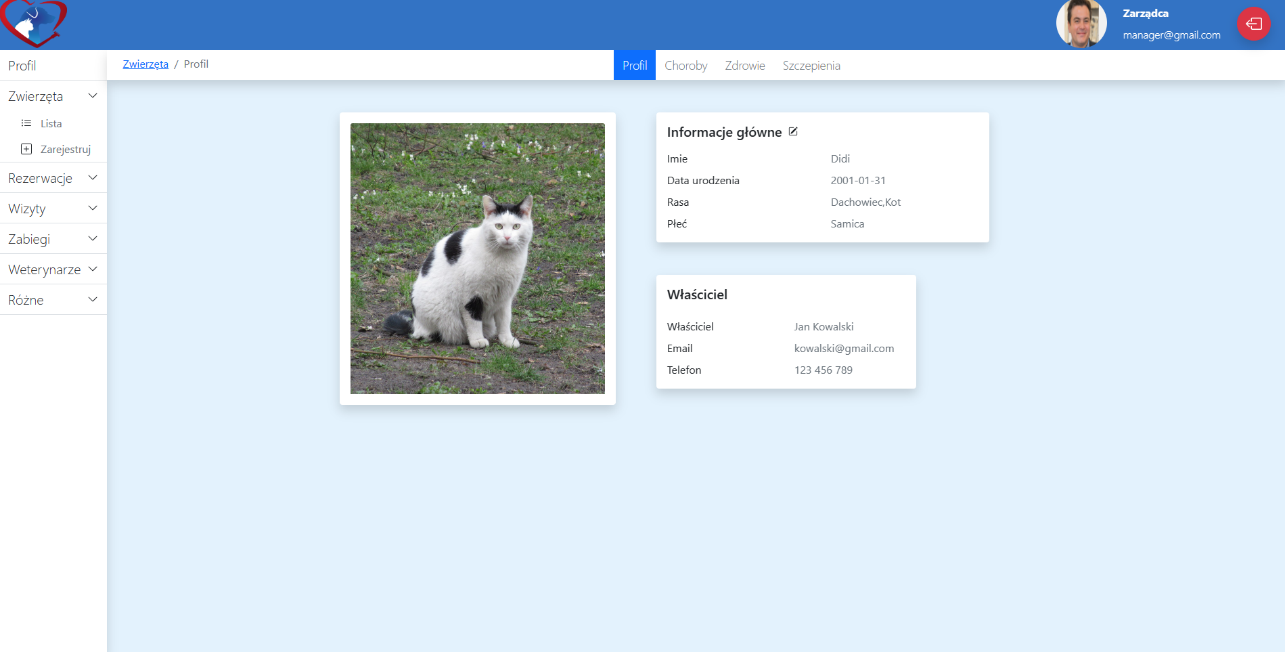
Obraz zawierający stół

Opis wygenerowany automatycznie

Rysunek 7 Lista

### Profil zwierzęcia

Profil zwierzęcia posiada 4 zakładki. Zakładka Profil przedstawia informacje profilowe zwierzęcia jak i jego właściciela (Rysunek 8). Zakładka choroby przedstawia historie chorób zwierzęcia (Rysunek 9). W przypadku, kiedy choroba jest niewyleczona na kolumnie pokazuje się odpowiedni guzik „Niewyleczone”. Kiedy użytkownik ją naciśnie zostanie wstawiona data wyleczenie odpowiadająca dacie kliknięcia. Zakładka zdrowie przedstawia informacje medyczne zwierzęcia, podzielone na odpowiednie kategorie. Weterynarz może je edytować po kliknięciu na ikonę przy tytule zakładki (Rysunek 10). Zakładka szczepienia przedstawia dwie listy szczepień, wykonanych oraz niewykonanych wymaganych. (Rysunek 11).

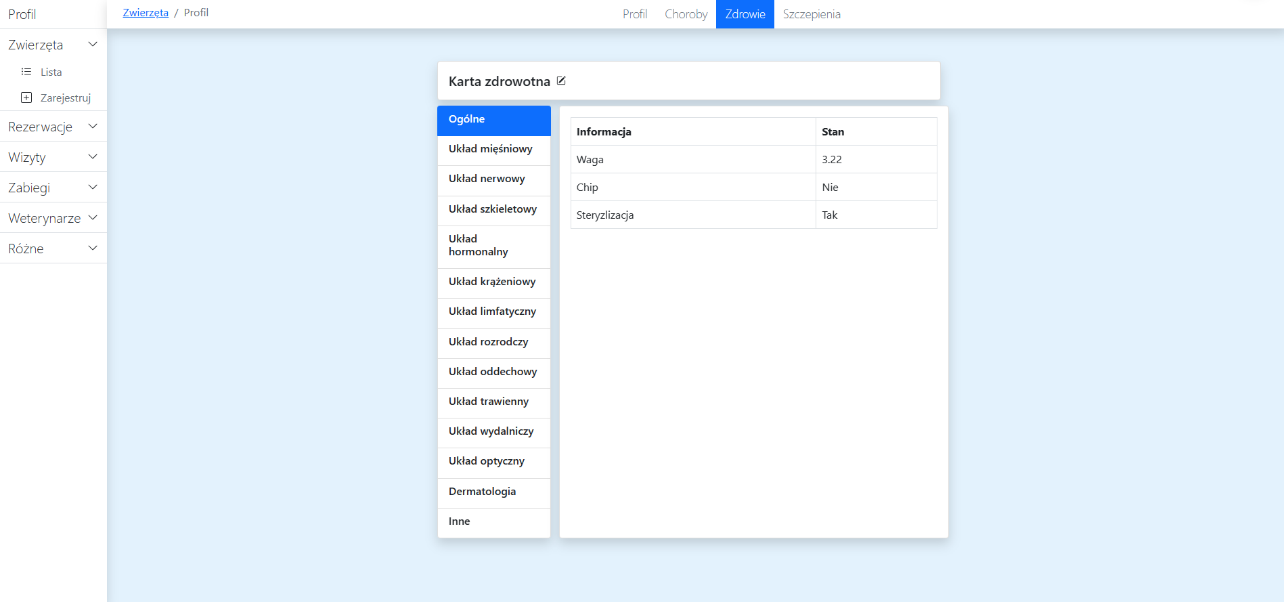


Rysunek 8 Profil zwierzęcia

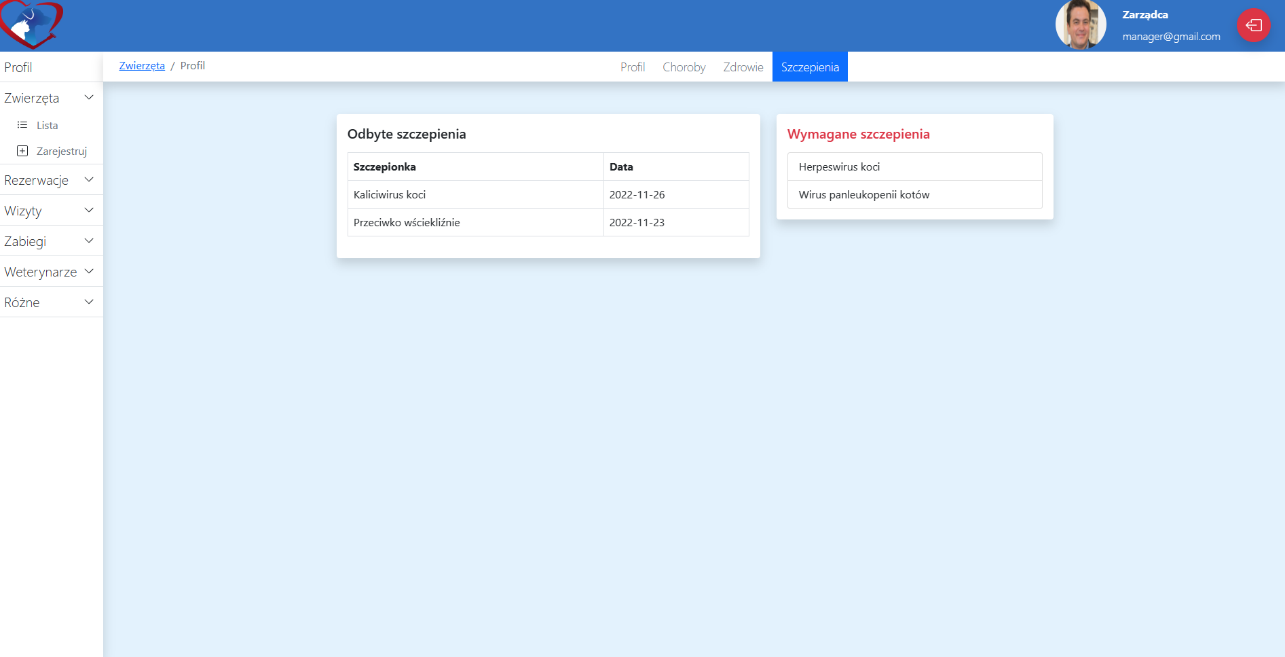
Obraz zawierający stół

Opis wygenerowany automatycznie

Rysunek 9 Choroby zwierzęcia



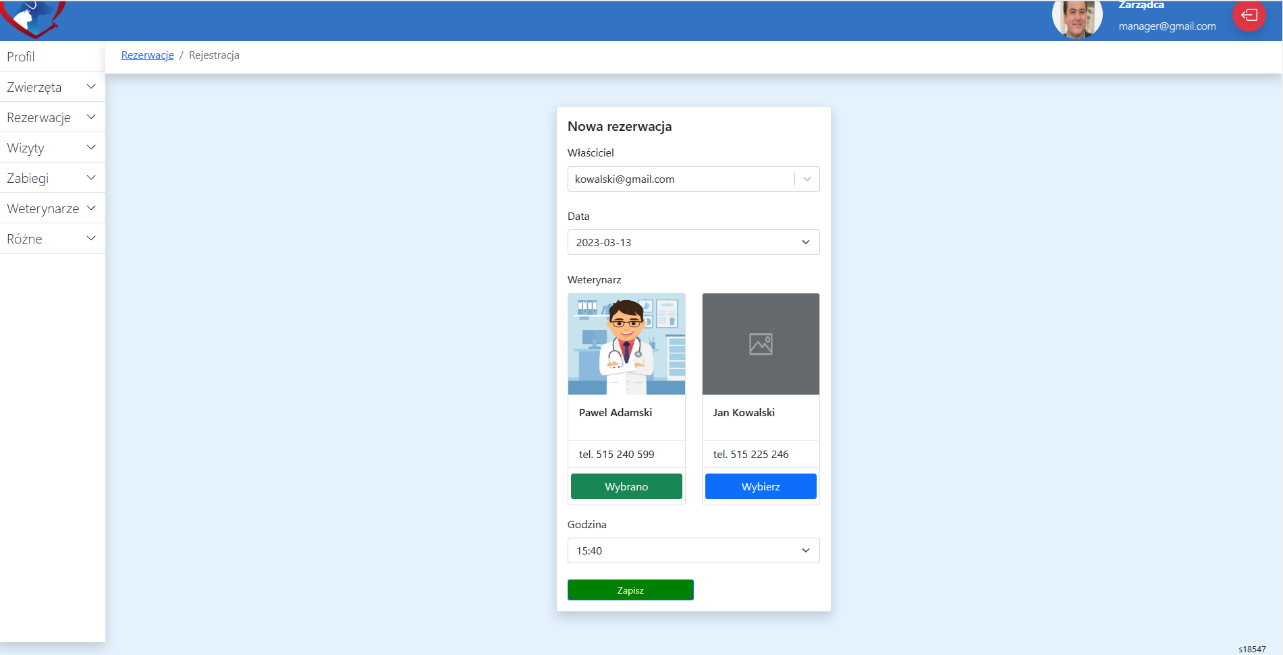
Rysunek 10 Zdrowie zwierzęcia



Rysunek 11 Szczepienia zwierzęcia

### Tworzenie rezerwacji

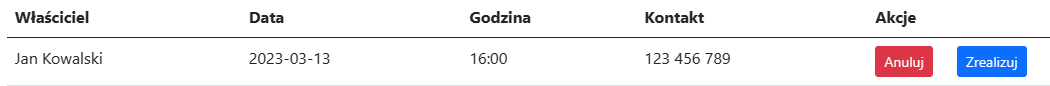
Użytkownik tworzy rezerwacje poprzez rozwinięcie opcji „Rezerwacje” na bocznym pasku i wybraniu opcji zarezerwuj. Na ekranie rezerwacji początkowo pojawiają się pola wyboru terminu rezerwacji oraz w przypadku, kiedy rezerwujący jest weterynarzem, wybór właściciela. Po wybraniu daty pojawia się lista weterynarzy dostępnych tego dnia. Użytkownik następnie wybiera weterynarza klikając na przycisk „Wybierz”. Kolejno pojawia się lista dostępnych terminów. Po wybraniu użytkownik zatwierdza rejestrację rezerwacji klikając na przycisk „Zapisz” (Rysunek 12).



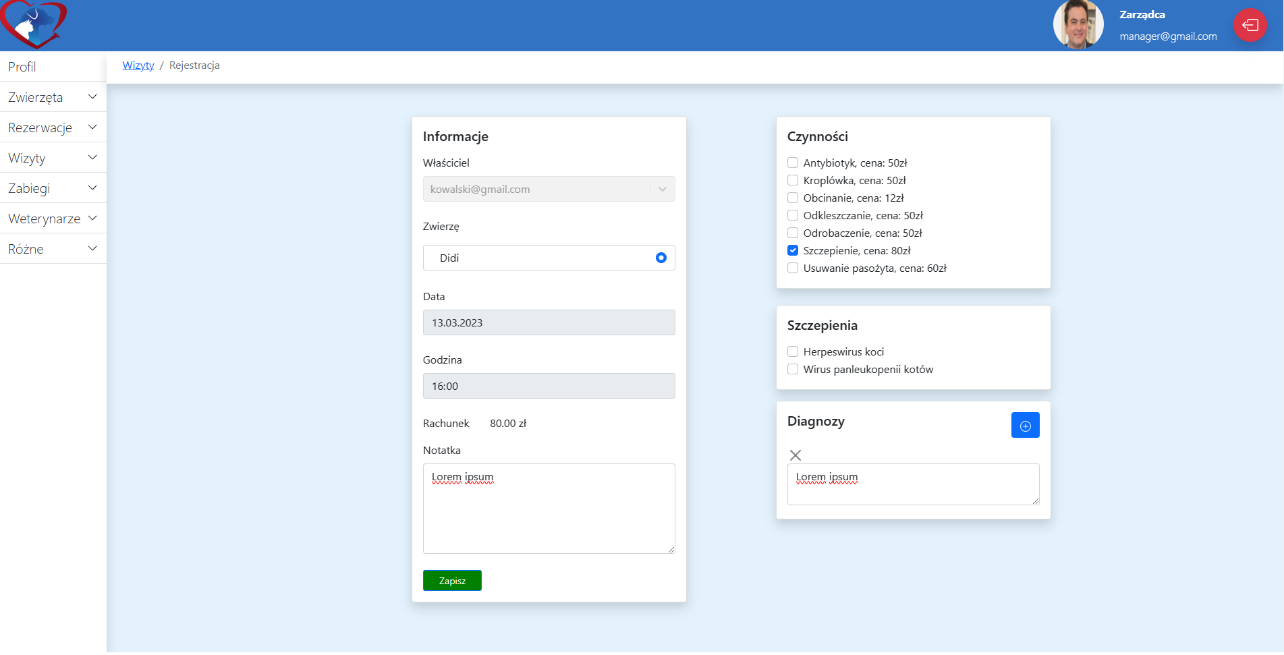
Rysunek 12 Rezerwacja

### Rejestracja wizyty

Użytkownik weterynarz ma możliwość rejestracji wizyty na dwa sposoby. Może to zrobić poprzez rozwinięcie zakładki „Wizyty” z pasku bocznym i kliknięciu na opcję „Zarejestruj” lub z ekranu listy rezerwacji klikając ba przycisk „Zrealizuj” w kolumnie „Akcje” (Rysunek 13). W takim przypadku pola „Data”, „Godzina” oraz „Właściciel” zostaną automatycznie wypełnione. Lista zwierząt pojawia się na podstawie wybranego właściciela. Użytkownik może wybrać wykonane czynności oznaczając checkbox przy nazwie. Na ich podstawie kalkulowane jest pole „Rachunek”. Na podstawie wybranego zwierzęcia pojawia się lista możliwych szczepień. Użytkownik ma możliwość wystawienia kilku diagnoz po kliknięciu na niebieską ikonę lub ich uśnięcie klikając na krzyżyk nad diagnozą. Użytkownik rejestruje wizytę klikając na przycisk „Zapisz” (Rysunek 14). Użytkownik zostanie następnie przekierowany do listy wizyt z właśnie zrejestrowaną wizytą oznaczoną na zielono (Rysunek 15).



Rysunek 13 Realizacja rezerwacji



Rysunek 14 Wizyta

Obraz zawierający stół

Opis wygenerowany automatycznie

Rysunek 15 Zarejestrowana wizyta

# Projekt

## Ogólny opis architektury

Opisywana aplikacja całości składa się z trzech głównych modułów. Aplikacji internetowej opartej o architekturę Single-Page Application napisanej przy użyciu biblioteki React.js. Nawiązuje ona komunikację z serwerem REST API, zaimplementowanego z użyciem frameworka Express.js, poprzez wysyłanie żądań HTTP z danymi w formacie JSON, do odpowiednich końcówek. Serwer łączy się z systemem zarządzania bazą danych MS SQL Server i na podstawie przesłanych danych i wykonuje zapytanie lub ich szereg w języku Transact-SQL. Bazując na wyniku zapytań SQL zwraca odpowiedź HTTP o stosownym kodzie i ciele. Po otrzymaniu odpowiedzi na jej podstawie aplikacja kliencka wyświetla odpowiednią zawartość na stronie internetowej. Aplikacja kliencka i serwerowa zostały napisane w rozszerzeniu języka JavaScript, TypeScript. Uzupełnia on go o m.in. statyczne typowanie.

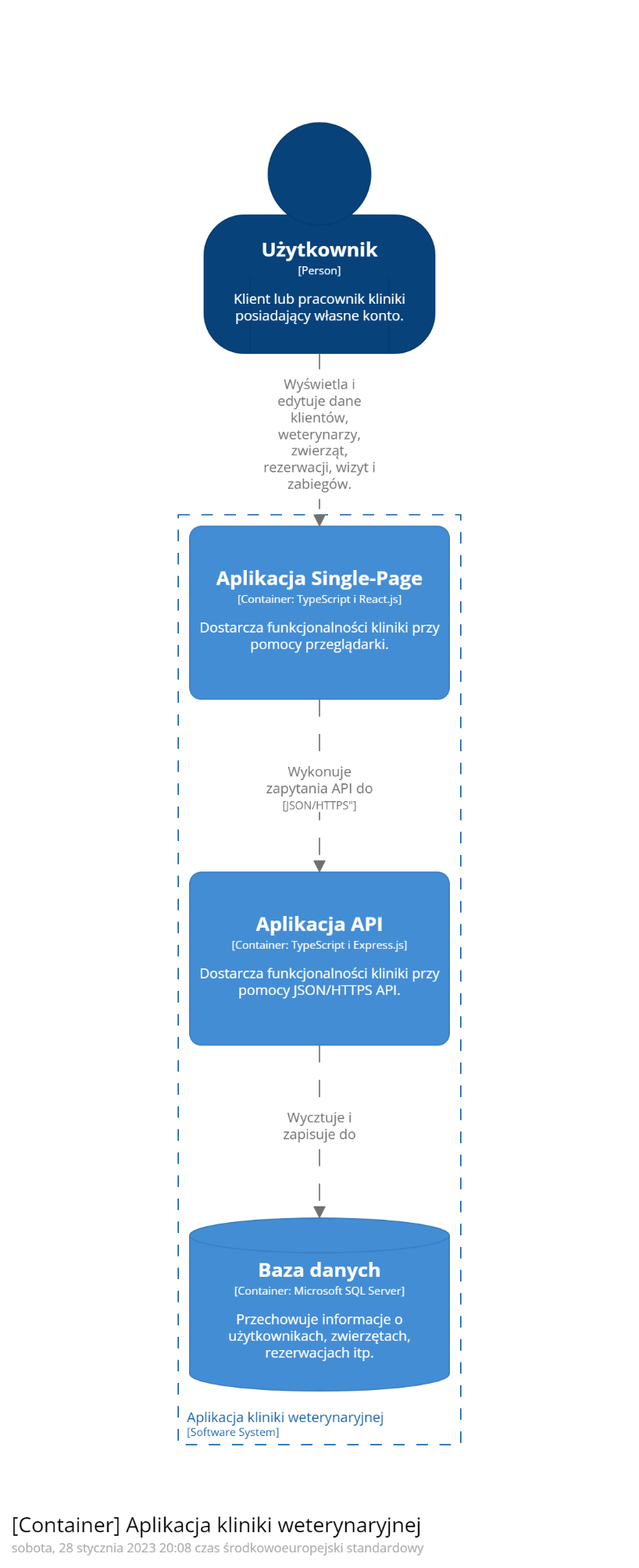
## Model C4

### Poziom 1



Rysunek 16 Model C4 Poziom 1

### Poziom 2



Rysunek 17 Model C4 Poziom 2

## Aplikacja kliencka

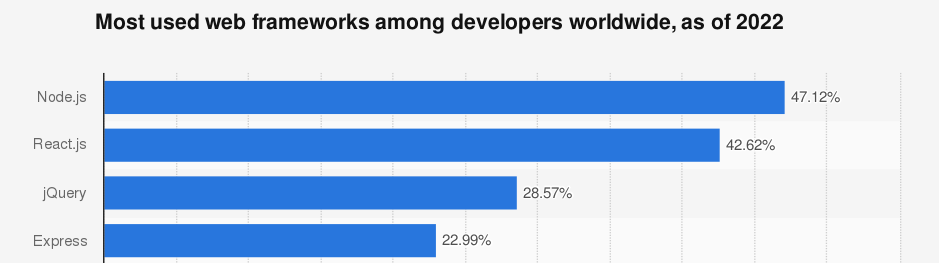
### Wykorzystane biblioteki

#### React.js

Otwarto-źródłowa i darmowa biblioteka JavaScript, służąca do tworzenia UI aplikacji internetowych. Biblioteka ta posiada szereg funkcjonalności pozwalających na budowanie aplikacji typu SPA:

* **Wirtualny DOM** - reprezentacja JavaScript Document Object Model. React przechowuje DOM w pamięci, gdzie przy zmianie stanu wyszukuje różnice między wirtualnymi i prawdziwym i wprowadza zmiany. Proces ten o wiele szybszy niż tradycyjna aktualizacja DOM przy pomocy JavaScript. [6]
* **JSX** - rozszerzenie składni JavaScript. Pozwala na wstawianie znaczników HTML bezpośrednio przy pomocy JavaScript. [7]
* **Hooki** – odpowiadają za bieżący stan komponentów aplikacji. Do najużyteczniejszych hooków bibliotek React.js wlicza się m.in. useState, który określa i pozwala na zarządzanie stanem komponentów oraz useEffect, odpowiedzialny za zarządzanie cyklem życia komponentu. [8]

Został stworzony przez firmę Facebook (obecnie Meta) w 2013 roku i obecnie jest jednym z najpopularniejszych szkieletów webowych [9] . W związku z popularnością biblioteka ta cieszy się również wysokim wsparciem ze strony społeczności.



Rysunek 18 Wykres popularności React.js

#### React Router

Standardowa biblioteka odpowiedzialna za routring aplikacji React.js. Pozwala na poruszanie się między widokami poszczególnych komponentów.

#### Bootstrap 5

Biblioteka CSS, rozwijana przez programistów Twittera. Zawiera ona zestawy zdefiniowanych stylów napisanych w HTML, CSS oraz wykorzystujące JavaScript, poszerzający je o dodatkowa funkcjonalność. Elementy biblioteki wykorzystane w aplikacji to m.in.:

* Formularze
* Przyciski
* Tabele
* Nawigacje

### Opis architektury

#### Uwierzytelnianie i autoryzacja

Wyróżnieni użytkownicy aplikacji są identyfikowani przy pomocy właściwości obiektu Window przeglądarki sessionStorage. Na podstawie żądania HTTP do aplikacji serwerowej zwracany jest obiekt z wartościami określającymi użytkowania, jego typ oraz wygenerowany JSON Web Token, który zostaje umieszczony w sessionStorage przeglądarki.

 const user: userInfo = {

                            userId: user.UserId,

                            userType: userType,

                            userTypeId: userTypeId,

                            isManager: user.Manager,

                            Email: user.Email,

                            token: token,

                            ProfileImage: user.ProfileImage,

                        };

const handleLogin = (user) => {

        sessionStorage.setItem('user', user);

        naviagate('/profile');

    };

Zostały również zaimplementowane funkcje pomocnicze w celu pozyskania odpowiednich zawartości z sessionStorage.

#### Aplikacja

Aplikacja zbudowana jest z szeregu komponentów, gdzie na podstawie działań użytkownika, przebudowywany jest obiektowy model dokument (DOM). Zawartość aplikacja jest renderowana w kontekście dokumencie HTML wewnątrz znacznika od id root, gdzie komponent App.tsx reprezentuję całą zawartość widoku aplikacji.

const root = ReactDOM.createRoot(document.getElementById('root')!);

root.render(

    <BrowserRouter>

        <App />

    </BrowserRouter>

);

#### Routing

Wewnątrz komponentu App.tsx zadeklarowane są odpowiednie adresy URL przy pomocy biblioteki React Router. Każdy z nich posiada przypisany element będący komponentem zajmującym się budowę odpowiedniej strony internetowej aplikacji. W czasie nawigacji na odpowiedni adres, w to miejsce wstawiana jest zawartość komponentu.

<Routes>

      <Route path="/profile" element={isNotLogged(<UserProfile />)} />

      <Route path="/animals" element={isNotLogged(<AnimalList />)} />

     <Route path="/animals/register element={isNotLogged(<AnimalForm />)}

                    />

Zadeklarowane zostały również dodatkowe funkcje odpowiedzialne za sprawdzenia czy użytkownik ma odpowiednie uprawnienia do danego adresu URL. W przypadku braku dostępu jest on przekierowany na stronę startową.

const isNotLogged = (element) => {

        if (isAuthenticated()) {

            return element;

        } else return <Navigate to="/" />;

    };

const isNotPersonel = (element) => {

        if (isAuthenticated()) {

            if (isVet()) {

                return element;

            } else return <Navigate to="/profile" />;

        } else return <Navigate to="/profile" />;

    };

#### Strony i komponenty

Aplikacja jest zbudowana z grup komponentów funkcyjnych biblioteki React.js. W aplikacji można wyróżnić dwie główne kategorie:

* **Obraz zawierający tekst

  Opis wygenerowany automatycznieStrona (Page)** – są odpowiedzialne za główną budowę strony aplikacji. Są one przypisane do odpowiednich adresów URL w pliku App.tsx. W celu zwiększenia czytelności pliku, wydzielono część ich zawartości do oddzielnych komponentów.

Rysunek 19 Struktura plików strony

* **Komponenty wielokrotnego użytku** – jak nazwa wskazuje. Został stworzony szereg komponentów, które można wykorzystywać wielokrotnie w budowie różnych stron. Większość z nich tworzy zawartość na podstawie podanych parametrów.

Stany w aplikacji są zarządzanie przy użyciu hooku useState() , zadeklarowanych wewnątrz komponentów.

const [pagedList, setPagedList] = useState<Animal[][]>([]);

Na podstawie stanów następnie budowane są fragmeny DOM przy użyciem składni JSX.

<tbody>

{pagedList[selectedPage]?.map((animal) => {

      return (

      <tr

          key={animal.AnimalId}

               onClick={() => {

                navigate('/animals/' + animal.AnimalId)}}

            className={newId === animal.AnimalId ? 'table-success': ''>

<td>{animal.Name}</td>

               <td>{`${animal.AnimalType?.Family},

${animal.AnimalType?.Race}`}

               </td>

               <td>{animal.BirthDate}</td>

              <td>{animal?.Owner.Email}</td>

</tr>);})}

</tbody>

Strony jak i poszczególne komponenty pobierają dane z aplikacji serwerowej podczas ładowania strony z użyciem hooka useEffect(). W zależności od rodzaju użytkownika jest wykowana żądanie do aplikacji serwerowej, na podstawie których modyfikowane są odpowiednie stany komponentu.

 let response;

                if (isOwner()) {

                    response = await animalApiCalls.getAnimalsbyOwner(

                        getCurrentUser().userId

                    );

                } else if (isVet() || isManager()) {

                    response = await animalApiCalls.getAnimals();

                }

                if (response) {

                    if (response.status == 200) {

                        const list = await response.json();

                        divideListIntoPages(list);

                        setAnimalList(list);

                    }

                    if (response.status == 500) {

                        setServerError(true);

                    }

                    if (response.status == 404) {

                        setEmpty(true);

                    }

                }

#### Wymiana danych

Za wymianę danych odpowiedzialne są funkcję apiCalls, gdzie na podstawie dostarczonych argumentów, tworzone jest odpowiednie żądanie HTTP do aplikacji serwerowej. Wykonywane są przy pomocy funkcji API fetch().

getAnimals = async () => {

        const url = this.baseUrl;

        const options = createHttpGetOptions(isVet());

        const promise = await fetch(url, options);

        responseAuthenticationHandler(promise);

        return promise;

    };

Zostały również zadeklarowane dodatkowe funkcji opowiadające za deklarację parametrów żądania HTTP.

export function createHttpGetOptions(haveAuth: boolean) {

    let token: string | null = null;

    haveAuth ? (token = getCurrentUser().token) : null;

    const options = {

        method: 'GET',

        headers: {

            'Content-Type': 'application/json',

            Authorization: 'Bearer ' + token,

        },

    };

    return options;

}

## Aplikacja serwerowa

### Wykorzystanie biblioteki

#### Node.js

Aplikacja została napisana przy użyciu Node.js. Jest to jednowątkowe środowisko uruchomienia JavaScript wykorzystujące silnik V8. Pozwala na tworzenie skalowalnych aplikacji serwerowych [10]. Na wybór tego środowiska wynikł z kilku powodów. Jest on wysoce efektywny w procesie tworzenia kodu, szybkości i wydajność. Ma on również wysoką popularny, co zapewnia mu szerokie wsparcie ze strony społeczności [11]. Jeszcze innym istotnym czynnikiem jest możliwość napisania zarówno aplikacji serwerowej jak i aplikacji klienckiej w tym samych języku.

#### Express.js

Szybki i minimalistyczny szkielet Node.js. Służy on do budowy aplikacji typu REST API. Jest on wysoce elastyczny, minimalny i skalowalny. Pozwala również na takie funkcjonalności jak zarządzanie ciasteczkami lub sesją [12] .

#### Node-mssql

Klient systemu zarządzania bazą danych Microsoft SQL Server. Pozwala on na wymianę danych z bazą danych poprzez tworzenie surowych zapytań w języku T-SQL. W celu jej użycia jest odpowiedni plik konfiguracyjny. Zdecydowałem się na użycie zapytań SQL zamiast zastosowania mapowania obiektowo-relacyjnego (ORM) z kilku powodów:

* Szybkość wynikającej m.in. z braku dodatkowej warstwy abstrakcji.
* Możliwość stworzenia złożonych zapytań.

#### nbcrypt.js

Aplikacja do zapewnienia bezpieczeństwa haseł użytkowników, wykorzystuje funkcje haszującą udostępnioną poprzez bibliotekę Bcrypt. Wykorzystuje ona symetryczny blok szyfrowy Blowfisher [13].

#### Json Web Token

Pozwala na generowanie JSON Web Tokenów. Jest to otwarty standard, którego głównym zastosowaniem jest w proces uwierzytelniania i autoryzacji. [14]

#### UUID4

Biblioteka służąca do generowania identyfikatorów UUID. Jest on ciąg znaków w formacie 8-4-4-4-12, co zapewnia wysoką gwarancję unikalności identyfikatora. [15]

### Opis architektury

Aplikacja serwerowa jest interfejsem REST API. Zadeklarowana jest funkcja zwracająca aplikację z polem przyjmującym plik konfigurujący połączenie z bazą danych oraz wykorzystującą funkcjonalności dostarczonych przez Express.js, jakich jak express() i use.

const app = express();

export default function (db) {

…

return app;

}

Framework Express.js bezpośrednio nie narzuca konkretnych wzorców architektonicznych, pomimo tego zdecydowałem się na napisanie aplikacji w języku TypeScript przy użyciu paradygmatu programowania zorientowanej na obiektach (Object-oriented programming). Wybór ten został podyktowany chęcią modularyzacji aplikacji oraz możliwością wykorzystania wstrzykiwania zależności. Zaletą tego wyboru jest m.in. większa czytelności kodu oraz ułatwienie pisania testów jednostkowych. Jako że aplikacja składa się z szeregu końcówek REST API, podzieliłem je na 3 typy klas, których instancje są zadeklarowane wewnątrz funkcji odpowiedzialnej tworzącej aplikację.

const reservationRepository = new ReservationRepository(

        db,

        ownerRepository,

        vetRepository

    );

const reservationRouter = new ReservationRouter(reservationController);

const reservationController = new ReservationController(reservationRepository);

app.use('/reservations', reservationRouter.router);

#### Klasy Router

Klasy router jest klasą dodatkową rozdzielającą definicje końcówek API od klas Controller. Posiada ona właściwość o klasie Router dostarczonej z frameworka Express.js oraz przyjmuje jako parametr w konstruktorze obiekt klasy Controller, której metody są kolejno wywoływane dla odpowiedniej końcówki.

router: Router;

    constructor(reservationController: ReservationController) {

        const router = express.Router();

        router.get('/', isAuthorizated, reservationController.getReservations);

        router.post('/', isAuthorizated, reservationController.registerReservation);

        router.delete(

            '/:ReservationId',

            isAuthorizated,

            reservationController.deleteReservation

        );

        this.router = router;

    }

#### Klasy Controller

Klasy Controller odpowiedzialna jest za przetwarzaniem żądań HTTP. Jako parametr konstruktora przyjmują obiekty klasy Repository. Na podstawie metody i parametrów żądania, wywoływana jest odpowiednia metoda obiektu klasy Repository, gdzie w zależności od wyniku jej działania zwracana jest stosowna odpowiedź HTTP o odpowiednim kodzie statusu i zawartości w formacie JSON.

    reservationRepository: ReservationRepository;

    constructor(reservationRepository: ReservationRepository) {

        this.reservationRepository = reservationRepository;

    }

    getReservations = async (req, res) => {

        const parameters: GetReservationParameters = {

            vetId: req.query.VetId,

            date: req.query.Date,

            ownerId: req.query.OwnerId,

        };

        const results = await this.reservationRepository.getReservations(

            parameters

        );

        return getResponseHandler(res, results);

    };

    registerReservation = async (req, res) => {

        const reservation: Reservation = req.body;

        const results = await this.reservationRepository.createReservation(

            reservation

        );

        return postResponseHandler(res, results);

    };

Dodatkowo, w celu zmniejszenia powtarzalności kodu zadeklarowano funkcje odpowiedzialna za zwracanie poszczególnych odpowiedzi HTTP w zależności od rodzaju operacji.

export const getResponseHandler = async (response, results) => {

    if (results instanceof Error) {

        return response.status(500).json({});

    } else if (results == null) {

        return response.status(404).json({});

    } else return response.status(200).json(results);

};

#### Klasy Repository

Klasy Repository przyjmują plik konfiguracyjny databaseConfiguration jako parametr, dzięki któremu przy pomocy biblioteką mssql, tworzona jest pula połączeń z bazą danych MS SQL Server. Na podstawie dostarczonych parametrów oraz określonej metody, tworzone jest zapytanie w języku T-SQL realizujące odpowiednią operację CRUD do bazy danych (od ang. create, read, update and delete). W przypadku kiedy metoda klasy wymaga danych z innej tabeli, jako właściwość zadeklarowany jest odpowiedni obiekt klasy Repository.

animalTypeRepository: AnimalTypeRepository;

    ownerRepository : AnimalTypeRepository;

    constructor(

        databse,

        animalTypeRepository: AnimalTypeRepository,

        ownerRepository: OwnerRepository

    ) {

        super(databse);

        this.animalTypeRepository = animalTypeRepository;

        this.ownerRepository = ownerRepository;

    }

    getAnimal = async (animalId: string) => {

        try {

            const pool = await sql.connect(this.databaseConfiguration);

            const result = await pool

                .request()

                .input('AnimalId', sql.VarChar, animalId)

                .query(

                    'Select AnimalId,Name,BirthDate,AnimalTypeId,OwnerId,ProfileImage,Sex From Animal where AnimalId=@AnimalId'

                );

            const animalRecord = result.recordset[0];

            if (animalRecord == undefined) {

                return null;

            }

            const animalType = await this.animalTypeRepository.getAnimalTypes({

                animalTypeId: animalRecord.AnimalTypeId,

            });

            const ownerResult = await this.ownerRepository.getOwner(

                animalRecord.OwnerId

            );

            const newAnimal = new Animal(

                animalId,

                animalRecord.Name,

                animalRecord.BirthDate.toISOString().split('T')[0],

                animalRecord.OwnerId,

                animalRecord.ProfileImage,

                animalRecord.Sex,

                animalRecord.AnimalTypeId,

                animalType,

                { Email: ownerResult.Email }

            );

            return newAnimal;

        } catch (error) {

            console.log(error);

            return error;

        }

    };

W metodach wymagających modyfikacji zwartości bazy danych przy pomocy kilku wyrażeń SQL, zastosowano funkcję biblioteki mssql pozwalającą na zastosowanie transakcji, gdzie w przypadku wystąpienia błędu, cała transakcja zostanie cofnięta, w celu zachować aksjomatów ACID (niepodzielność, spójność, izolacja, trwałość).

  const pool = await sql.connect(this.databaseConfiguration);

            const transaction = new sql.Transaction(pool);

            try {

                await transaction.begin();

                let results = await new sql.Request(transaction)

                    .input('UserId', sql.VarChar, UserId)

                    .input('Email', sql.VarChar, Email)

                    .input('Password', sql.VarChar, hashedPassword)

                    .input('Name', sql.VarChar, Name)

                    .input('LastName', sql.VarChar, LastName)

                    .input('Contact', sql.VarChar, Contact)

                    .query(

                        'INSERT INTO Owner(OwnerId,Name,LastName,Contact) Values(@UserId,@Name,@LastName,@Contact)'

                    );

                if (results.rowsAffected[0] != 1) {

                    throw Error('Transaction error');

                }

                results = await new sql.Request(transaction)

                    .input('UserId', sql.VarChar, UserId)

                    .input('Email', sql.VarChar, Email)

                    .input('Password', sql.VarChar, hashedPassword)

                    .query(

                        'INSERT INTO [USER](UserId,Email,Password,OwnerId,VetId,Manager)values(@UserId,@Email,@Password,@UserId,null,null);'

                    );

                if (results.rowsAffected[0] != 1) {

                    throw Error('Transaction error');

                }

                await transaction.commit();

                return UserId;

            } catch (error) {

                transaction.rollback();

                console.log(error);

                return Error('Trasaction error');

            }

#### Funkcje pośredniczące

Aplikacja posiada funkcją isAuthorizated. Sprawdza ona czy przekazywane żądanie posiada nagłówek Token Bearer oraz odpowiedni JWT. Ma to za zadanie autoryzacji żądania, gdzie w przypadku jej braku, zostanie zwrócona stosowna odpowiedź.

export const isAuthorizated = async (req, res, next) => {

    const authHeader = req.headers['authorization'];

    const token = authHeader && authHeader.split(' ')[1];

    const key = configAuthKey.default.secret;

    if (token == null) {

        return res.status(401).json({});

    } else {

        await jwt.verify(token, key, (err, user) => {

            if (err) {

                return res.status(403).json({});

            } else {

                return next();

            }

        });

    }

};

Dodatkowo aplikacja posiada szereg dodatkowych funkcji, wspomagające realizację założeń aplikacji.

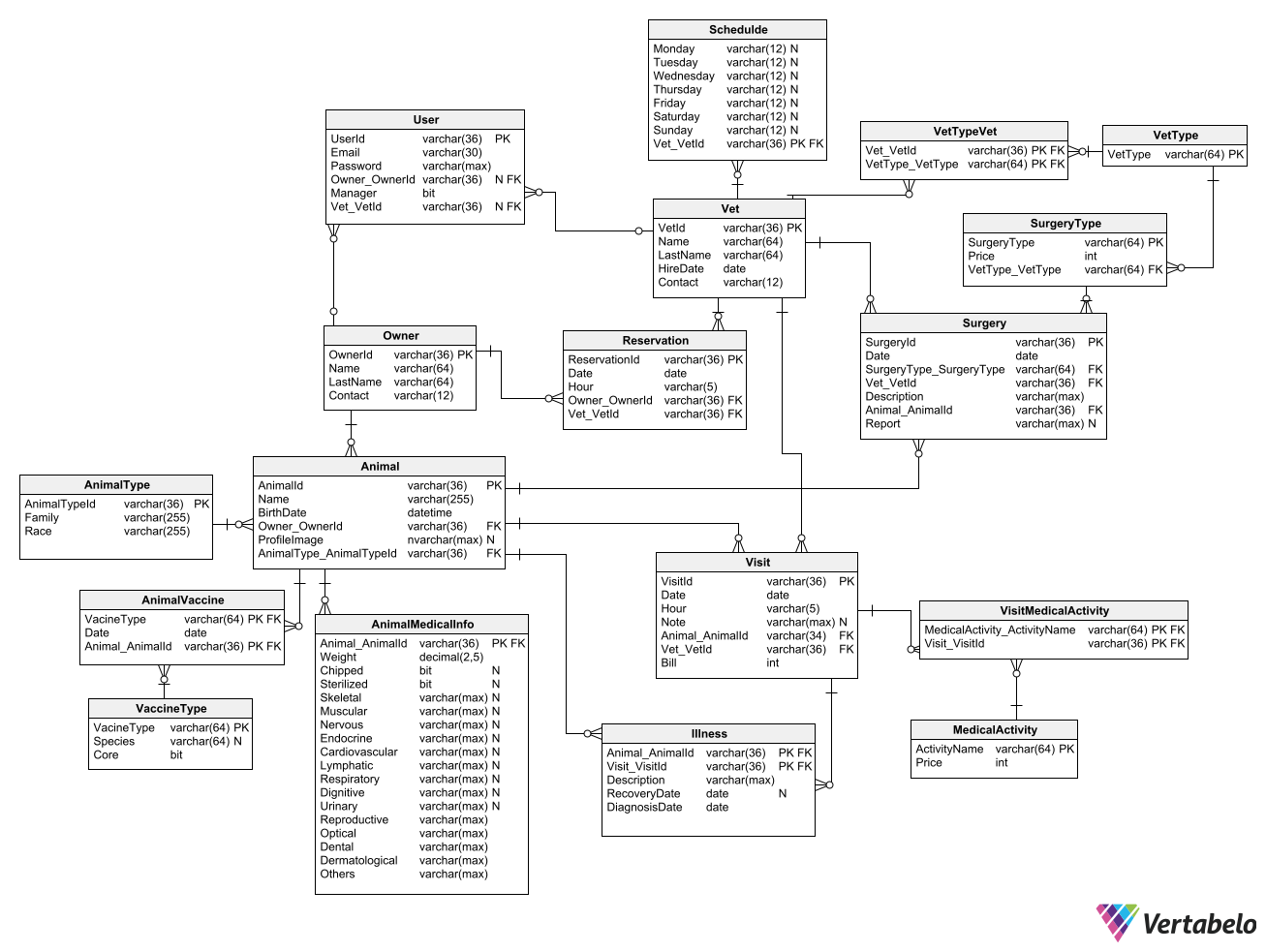
## Warstwa trwałości

### Baza danych

Warstwa trwałości została zaimplementowana w formie relacyjnej bazy danych. Wykorzystana został system zarządzania bazą danych MS SQL Server w darmowej wersji Express. Wykorzystuje ona rozszerzenie języka SQL, Transact-SQL. Zdecydowałem się na wybór na tego systemu zarządzania bazą danych z kilku wymienionych powodów:

* Możliwość nadaniu tabeli pogrupowanego indeksu. Indeksy pogrupowanie zwiększają wydajność zapytań sortujących. Większość zapytań do bazy danych opisywanej aplikacji wymagające pobrania grupy wierszy są sortowane po poszczególnych kolumnach, więc użycie tego typu indeksów jest wysoko zmniejszy ich koszt.
* Uprzednie doświadczenie, zdobyte w procesie studiów.

### Diagram encji



Rysunek 20 Diagram encji

#### Tabela User

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nazwa pola** | **Typ** | **Wymagane** | **Opis** | **Indeks** |
| UserId | varchar(36) | Tak | Klucz główny | Niepogrupowany |
| Email | varchar(36) | Tak | Email użytkownika | Niepogrupowany |
| Password | varchar(36) | Tak | Hasło użytkownika |  |
| OwnerId | varchar(36) | Nie | Klucz obcy | Niepogrupowany |
| VetId | varchar(36) | Nie | Klucz obcy | Niepogrupowany |
| Manager | bit | Tak | Określa czy użytkownik jest zarządcą |  |

Tabela 33 User

#### Tabela Owner

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nazwa pola** | **Typ** | **Wymagane** | **Opis** | **Indeks** |
| OwnerId | varchar(36) | Tak | Klucz główny | Niepogrupowany |
| Name | varchar(36) | Tak | Imię właściciela |  |
| LastName | varchar(36) | Tak | Nazwisko właściciela |  |
| Contact | varchar(36) | Tak | Numer telefonu właściciela |  |

Tabela 34 Owner

#### Tabela Vet

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nazwa pola** | **Typ** | **Wymagane** | **Opis** | **Indeks** |
| VetId | varchar(36) | Tak | Klucz główny | Niepogrupowany |
| Name | varchar(36) | Tak | Imię weterynarza |  |
| LastName | varchar(36) | Tak | Nazwisko weterynarza |  |
| Contact | varchar(36) | Tak | Numer telefonu weterynarza |  |
| HireDate | date | Tak | Data zatrudnienia weterynarza |  |
| ProfileImage | varchar(max) | Nie | Zdjęcie profilowe zwierzęcia |  |

Tabela 35 Vet

#### Tabela VetType

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nazwa pola** | **Typ** | **Wymagane** | **Opis** | **Indeks** |
| VetType | varchar(36) | Tak | Klucz główny | Niepogrupowany |

Tabela 36 VetType

#### Tabela VetTypeVet

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nazwa pola** | **Typ** | **Wymagane** | **Opis** | **Indeks** |
| VetType | varchar(36) | Tak | Klucz obcy, klucz główny | Niepogrupowany |
| VetId | Varchar(36) | Tak | Klucz obcy, klucz główny | Niepogrupowany |

Tabela 37 VetTypeVet

#### Tabela Schedulde

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nazwa pola** | **Typ** | **Wymagane** | **Opis** | **Indeks** |
| VetId | varchar(36) | Tak | Klucz obcy | Niepogrupowany |
| Monday | varchar(12) | Nie | Godziny pracy w poniedziałek |  |
| Tuesday | varchar(12) | Nie | Godziny pracy we wtorek |  |
| Wednesday | varchar(12) | Nie | Godziny pracy w środę |  |
| Thursday | varchar(12) | Nie | Godziny pracy w czwartek |  |
| Friday | varchar(12) | Nie | Godziny pracy w piątek |  |
| Saturday | varchar(12) | Nie | Godziny pracy w sobotę |  |
| Sunday | varchar(12) | Nie | Godziny pracy w niedzielę |  |

Tabela 38 Schedulde

#### Tabela Animal

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nazwa pola** | **Typ** | **Wymagane** | **Opis** | **Indeks** |
| Animald | varchar(36) | Tak | Klucz główny | Niepogrupowany |
| Name | varchar(36) | Tak | Imię zwierzęcia |  |
| BirthDate | date | Tak | Data urodzenia zwierzęcia | Pogrupowany |
| OwnerId | varchar(36) | Tak | Id właściciela zwierzęcia | Niepogrupowany |
| ProfileImage | varchar(max) | Nie | Zdjęcie profilowe zwierzęcia |  |
| AnimalTypeId | varchar(36) | Tak | Klucz obcy |  |

Tabela 39 Animal

#### Tabela AnimalType

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nazwa pola** | **Typ** | **Wymagane** | **Opis** | **Indeks** |
| AnimalTypeId | varchar(36) | Tak | Klucz główny | Niepogrupowany |
| Family | varchar(255) | Tak | Rodzaj zwierzęcia |  |
| Race | varchar(255) | Tak | Rasa zwierzęcia |  |

Tabela 40 AnimalType

#### Tabela AnimalMedicalInfo

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nazwa pola** | **Typ** | **Wymagane** | **Opis** | **Indeks** |
| AnimalId | varchar(36) | Tak | Klucz obcy, Klucz główny | Niepogrupowany |
| Weight | decimal(2,5) | Nie | Waga zwierzęcia |  |
| Chipped | bit | Nie | Czy zwierzę posiada chip |  |
| Sterilized | bit | Nie | Czy zwierzę zostało wysterylizowanie |  |
| Skeletal | varchar(max) | Nie | Informacje o układzie kostnym |  |
| Muscular | varchar(max) | Nie | Informacje o układzie mięśniowym |  |
| Nervous | varchar(max) | Nie | Informacje o układzie nerwowym |  |
| Endocrine | varchar(max) | Nie | Informacje o układzie hormonalnym |  |
| Cardiovascular | varchar(max) | Nie | Informacje o układzie serowo-naczyniowym |  |
| Lymphatic | varchar(max) | Nie | Informacje o układzie limfatycznym |  |
| Respiratory | varchar(max) | Nie | Informacje o układzie oddechowym |  |
| Dignitive | varchar(max) | Nie | Informacje o układzie trawiennym |  |
| Urinary | varchar(max) | Nie | Informacje o układzie wydalniczym |  |
| Reproductive | varchar(max) | Nie | Informacje o układzie rozrodczym |  |
| Optical | varchar(max) | Nie | Informacje o układzie optyczny, |  |
| Dental | varchar(max) | Nie | Informacje o stanie uzębienia |  |
| Dermatological | varchar(max) | Nie | Informacje o stanie skóry |  |
| Others | varchar(max) | Nie | Inne |  |

Tabela 41 AnimalMedicalInfo

#### Tabela VaccineType

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nazwa pola** | **Typ** | **Wymagane** | **Opis** | **Indeks** |
| VaccineType | varchar(64) | Tak | Klucz główny | Niepogrupowany |
| Species | varchar(255) | Nie | Dla jakiego rodzaju zwierzęcia jest przeznaczona |  |
| Core | bit | Tak | Czy szczepionka jest wymagana |  |

Tabela 42 VaccineType

#### Tabela AnimalVaccine

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nazwa pola** | **Typ** | **Wymagane** | **Opis** | **Indeks** |
| VaccineType | varchar(64) | Tak | Klucz obcy, Klucz główny | Niepogrupowany |
| AnimalId | varchar(36) | Tak | Klucz obcy, Klucz główny | Niepogrupowany |
| Date | date | Tak | Data podania | Pogrupowany |

Tabela 43 AnimalVaccine

#### Tabela Reservation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nazwa pola** | **Typ** | **Wymagane** | **Opis** | **Indeks** |
| ReservationId | varchar(36) | Tak | Klucz główny | Niepogrupowany |
| Date | date | Tak | Data rezerwacji |  |
| Hour | varchar(5) | Tak | Godzina przyjęcia |  |
| OwnerId | varchar(36) | Tak | Klucz obcy | Niepogrupowany |
| VetId | varchar(36) | Tak | Klucz obcy | Niepogrupowany |

Tabela 44 Reservation

#### Tabela Visit

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nazwa pola** | **Typ** | **Wymagane** | **Opis** | **Indeks** |
| VetId | varchar(36) | Tak | Klucz główny | Niepogrupowany |
| Date | date | Tak | Data wizyty | Pogrupowany |
| Hour | varchar(5) | Tak | Godzina wizyty |  |
| Note | varchar(max) | Nie | Notatka o wizycie |  |
| VetId | varchar(36) | Tak | Klucz obcy | Niepogrupowany |
| AnimalId | varchar(36) | Tak | Klucz obcy | Niepogrupowany |

Tabela 45 Visit

#### Tabela MedicalActivity

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nazwa pola** | **Typ** | **Wymagane** | **Opis** | **Indeks** |
| ActivityName | varchar(64) | Tak | Klucz główny | Niepogrupowany |
| Price | int | Tak | Opłata za czynność |  |

Tabela 46 MedicalActivity

#### Tabela VisitMedicalActivity

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nazwa pola** | **Typ** | **Wymagane** | **Opis** | **Indeks** |
| ActivityName | varchar(64) | Tak | Klucz obcy, klucz główny | Niepogrupowany |
| VisitId | varchar(36) | Tak | Klucz obcy, klucz główny | Niepogrupowany |

Tabela 47 VisitMedicalActivity

#### Tabela Illness

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nazwa pola** | **Typ** | **Wymagane** | **Opis** | **Indeks** |
| AnimalId | varchar(36) | Tak | Klucz obcy, klucz główny | Niepogrupowany |
| VisitId | varchar(36) | Tak | Klucz obcy, klucz główny | Niepogrupowany |
| Description | varchar(max) | Tak | Opis dolegliwości |  |
| DiagnosisDate | date | Tak | Data diagnozy |  |
| RecoveryDate | data | Nie | Data wyleczenia |  |

Tabela 48 Illness

#### Tabela SurgeryType

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nazwa pola** | **Typ** | **Wymagane** | **Opis** | **Indeks** |
| SurgeryType | varchar(64) | Tak | Klucz główny | Niepogrupowany |
| Price | int | Tak | Cena za zabieg |  |
| VetType | varchar(64) | Tak | Klucz obcy | Niepogrupowany |

Tabela 49 SurgeryType

#### Tabela Surgery

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nazwa pola** | **Typ** | **Wymagane** | **Opis** | **Indeks** |
| SurgeryId | varchar(36) | Tak | Klucz główny | Niepogrupowany |
| Date | date | Tak | Data zabiegu | Pogrupowany |
| SurgeryType | varchar(64) | Tak | Klucz obcy | Niepogrupowany |
| Description | varchar(max) | Tak | Opis zabiegu |  |
| AnimalId | varchar(36) | Tak | Klucz obcy | Niepogrupowany |
| Report | varchar(max) | Nie | Raport z wykonanego zabiegu |  |

Tabela 50 Surgery

### Indeksy

W celu poprawienia szybkości zapytań SQL zostały zastosowanie na kolumnach odpowiednie indeksy:

* Indeksy niepogrupowane na kolumnach kluczy głównych i obcych. Ten typ indeksu dobrze wspiera zapytania warunkowe.
* Indeksy pogrupowane na wszelkich kolumnach z datami. Większość zapytań zakresowych jest posortowane po tych kolumnach, w celu zachowania porządku. Indeks pogrupowany bardzo dobrze wspiera takie rodzaju zapytania, co wynika z faktu, że dane w pliku są uporządkowane względem pogrupowanego indeksu. [16]
* Z wyniku częstych modyfikowania rekordów fillfactor został ustawiony na niski, aby zmniejszyć potrzebę częstego przebudowywania indeksów.

# Testy i wdrożenia

Testy aplikacji serowej i klienckiej zostały wykonane z użyciem szkieletu do testowania JavaScript, Jest. Jest on nastawiony przede wszystkim na prostotę. [17]

## Testy aplikacji serwerowej

Testy aplikacji zostały wykonane poprzez wykonanie testów zintegrowanych. Dla każdej końcówki API, w zależności od metody http został zaimplementowany szereg testów, symulujące każdą możliwą sytuację. W celu realizacji testów aplikacja łączy się ze specjalnie zaimplementowaną, identyczną kopię bazy danych.

Wykorzystano również test jednostkowe w testach funkcji nienawiązujących połączenia z bazą danych.

# Podsumowanie

## Efekty końcowe pracy

Rezultatem prac nad projektem uskutkował powstaniem aplikacji spełniającej narzucone wymagania użytkowe. Program jest zdolny do obsługi kliniki weterynaryjnej i jej klientów, co w znacznym stopniu zmniejszy czas wymagany do jej zarządzania.

Implementacja aplikacji pozwoliła mi na do głębsze zaznajomienie z wykorzystanymi technologiami i technikami oraz poznanie nowych. Proces ten dała mi również doświadczenie w procesie tworzenia oprogramowania komercyjnego, które będę mógł dalej wykorzystać i poszerzać w dalszej edukacji i karierze. Również samo określenie wymagań aplikacji pozwoliło mi na poznanie stosownego języka i stylistyki. Zarówno rezultat projektu, jak i nabyte doświadczenie w czasie jego tworzenia, uznaje za wysoce zadawalające.

## Możliwości kierunku rozwoju aplikacji

Możliwości kierunku rozwoju aplikacji to:

* Podłączenie aplikacji do API z informacji zdrowotnym zwierząt określonych ras.
* Dodanie kalkulatora BMI dla zwierząt.
* Stworzenie dedykowanej aplikacji mobilnej.

# Bibliografia

|  |  |
| --- | --- |
| [1] | McHugo, „Unlocking the origins and biology,” w *Unlocking the origins and biology of domestic animals using ancient DNA and paleogenomics*, 2019. |
| [2] | A. F. Fernando Ramirez Rozzi, „Earliest Animal Cranial Surgery: from Cow to Man in the Neolithic,” 19 Kwiecień 2018. [Online]. Available: https://www.nature.com/articles/s41598-018-23914-1. |
| [3] | K. Heintzman, „A cabinet of the ordinary: domesticating veterinary education,” w *A cabinet of the ordinary: domesticating veterinary education*, 1766-1799,2018. |
| [4] | jw, „W roku 2021 dostęp do internetu posiadało blisko 92,4 proc. polskich gospodarstw domowych,” 25 Listopad 2021. [Online]. Available: https://www.wirtualnemedia.pl/artykul/gus-internet-mobilny-w-polsce-spoleczenstwo-informacyjne-raport. |
| [5] | M. Mańka, „Raport FEDIAF o zwierzętach domowych,” 8 Marzec 2020. [Online]. Available: https://vetkompleksowo.pl/kategorie-tematyczne/raport-fediaf-a-nasi-czworonozni-towarzysze/. |
| [6] | „Reactjs.org,” [Online]. Available: https://reactjs.org/docs/faq-internals.html. |
| [7] | „W3Schools,” [Online]. Available: https://www.w3schools.com/react/react\_jsx.asp. |
| [8] | „Reactjs.org,” [Online]. Available: https://reactjs.org/docs/hooks-state.html. |
| [9] | „Statista,” 2022. [Online]. Available: https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/. |
| [10] | Node.js. [Online]. Available: https://nodejs.org/en/about/. |
| [11] | „Is Node.js Still Relevant For Your Startup in 2023?,” 11 Październik 2022. [Online]. Available: https://fireart.studio/blog/why-node-js-is-still-a-good-choice-for-your-startup-in-2020/. |
| [12] | M. Kuba, „Boringowl.io,” 16 Wrześeń 2020. [Online]. Available: https://boringowl.io/tag/express-js. |
| [13] | H. Patel, „blog.logrocket.com,” 15 Listpoad 2022. [Online]. Available: https://blog.logrocket.com/password-hashing-node-js-bcrypt/. |
| [14] | „JSON-WEB-TOKENS,” [Online]. Available: https://auth0.com/learn/json-web-tokens. |
| [15] | P. Leach, Microsoft, M. Mealling, Refactored Netowrds, LLC R. Salz, DataPower Technology, „A Universally Unique IDentifier (UUID) URN Namespace,” Lipiec 2005. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc4122. |
| [16] | IBM, „IBM,” 4 Październik 2020. [Online]. Available: https://www.ibm.com/docs/en/ias?topic=indexes-clustered-non-clustered. |
| [17] | Facebook, „Github,” 14 Maj 2014. [Online]. Available: https://github.com/facebook/jest/blob/88a94d5d1bc1f387317a3068bf510ab992c5dc64/README.md. |
| [18] | P. Krill, „Infoworld.com,” 10 Marzec 2016. [Online]. Available: https://www.infoworld.com/article/3042705/javascript-founder-brendan-eich-webassembly-is-a-game-changer.html. |
| [19] | I. N. Service, „InfoWorld,” 1 Październik 2012. [Online]. Available: https://www.infoworld.com/article/2614863/microsoft-augments-javascript-for-large-scale-development.html. |
| [20] | „Most used web frameworks among developers worldwide, as of 2022,” 21 Maj 2022. [Online]. Available: https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/. |
| [21] | M. Kuba, „boringowl.io,” 16 Wrzesień 2020. [Online]. Available: https://boringowl.io/tag/express-js/. |

# Spis ilustracji

[Rysunek 1 Ekran niezalogowanego użytkownika 33](#_Toc138056080)

[Rysunek 2 Ekran profilu właściciela 34](#_Toc138056081)

[Rysunek 3 Ekran profilu weterynarza 34](#_Toc138056082)

[Rysunek 4 Menu właściciela Rysunek 5 Menu weterynarza Rysunek 6 Menu zarządcy 35](#_Toc138056083)

[Rysunek 7 Lista 35](#_Toc138056084)

[Rysunek 8 Profil zwierzęcia 36](#_Toc138056085)

[Rysunek 9 Choroby zwierzęcia 37](#_Toc138056086)

[Rysunek 10 Zdrowie zwierzęcia 37](#_Toc138056087)

[Rysunek 11 Szczepienia zwierzęcia 38](#_Toc138056088)

[Rysunek 12 Rezerwacja 39](#_Toc138056089)

[Rysunek 13 Realizacja rezerwacji 39](#_Toc138056090)

[Rysunek 14 Wizyta 40](#_Toc138056091)

[Rysunek 15 Zarejestrowana wizyta 40](#_Toc138056092)

[Rysunek 16 Model C4 Poziom 1 42](#_Toc138056093)

[Rysunek 17 Model C4 Poziom 2 43](#_Toc138056094)

[Rysunek 18 Wykres popularności React.js 44](#_Toc138056095)

[Rysunek 19 Struktura plików strony 47](file:///C:\Users\user\OneDrive%20-%20Polsko-Japońska%20Akademia%20Technik%20Komputerowych\Desktop\Aplikacja%20kliniki%20weterynaryjnej.docx#_Toc138056096)

[Rysunek 20 Diagram encji 57](file:///C:\Users\user\OneDrive%20-%20Polsko-Japońska%20Akademia%20Technik%20Komputerowych\Desktop\Aplikacja%20kliniki%20weterynaryjnej.docx#_Toc138056097)

# Spis tabel

[Tabela 1 PU-1 13](#_Toc138056105)

[Tabela 2 PU-2 14](#_Toc138056106)

[Tabela 3 PU-3 14](#_Toc138056107)

[Tabela 4 PU-4 15](#_Toc138056108)

[Tabela 5 PU-5 16](#_Toc138056109)

[Tabela 6 PU-6 16](#_Toc138056110)

[Tabela 7 PU-7 16](#_Toc138056111)

[Tabela 8 PU-8 17](#_Toc138056112)

[Tabela 9 PU-9 18](#_Toc138056113)

[Tabela 10 PU-10 19](#_Toc138056114)

[Tabela 11 PU-11 19](#_Toc138056115)

[Tabela 12 PU-12 20](#_Toc138056116)

[Tabela 13 PU-13 20](#_Toc138056117)

[Tabela 14 PU-14 21](#_Toc138056118)

[Tabela 15 PU-15 21](#_Toc138056119)

[Tabela 16 PU-16 22](#_Toc138056120)

[Tabela 17 PU-17 22](#_Toc138056121)

[Tabela 18 PU-18 23](#_Toc138056122)

[Tabela 19 PU-19 23](#_Toc138056123)

[Tabela 20 PU-20 24](#_Toc138056124)

[Tabela 21 PU-21 24](#_Toc138056125)

[Tabela 22 PU-22 25](#_Toc138056126)

[Tabela 23 PU-23 25](#_Toc138056127)

[Tabela 24 PU-24 26](#_Toc138056128)

[Tabela 25 PU-25 26](#_Toc138056129)

[Tabela 26 PU-26 27](#_Toc138056130)

[Tabela 27 PU-27 27](#_Toc138056131)

[Tabela 28 PU-28 28](#_Toc138056132)

[Tabela 29 PU-29 29](#_Toc138056133)

[Tabela 30 PU-30 30](#_Toc138056134)

[Tabela 31 PU-31 30](#_Toc138056135)

[Tabela 32 PU-32 31](#_Toc138056136)

[Tabela 33 User 58](#_Toc138056137)

[Tabela 34 Owner 58](#_Toc138056138)

[Tabela 35 Vet 58](#_Toc138056139)

[Tabela 36 VetType 59](#_Toc138056140)

[Tabela 37 VetTypeVet 59](#_Toc138056141)

[Tabela 38 Schedulde 59](#_Toc138056142)

[Tabela 39 Animal 60](#_Toc138056143)

[Tabela 40 AnimalType 60](#_Toc138056144)

[Tabela 41 AnimalMedicalInfo 61](#_Toc138056145)

[Tabela 42 VaccineType 61](#_Toc138056146)

[Tabela 43 AnimalVaccine 61](#_Toc138056147)

[Tabela 44 Reservation 62](#_Toc138056148)

[Tabela 45 Visit 62](#_Toc138056149)

[Tabela 46 MedicalActivity 62](#_Toc138056150)

[Tabela 47 VisitMedicalActivity 62](#_Toc138056151)

[Tabela 48 Illness 63](#_Toc138056152)

[Tabela 49 SurgeryType 63](#_Toc138056153)

[Tabela 50 Surgery 63](#_Toc138056154)