

1. Softwareowy stos technologiczny

Rozdział ten skupia się na opisie środowisk uruchomieniowych oraz technologiach użytych w procesie tworzenia aplikacji.

1.1. Wykorzystane środowiska uruchomieniowe

W dzisiejszych czasach tworzenie aplikacji wymaga nie tylko znajomości języków programowania, ale również wykorzystania odpowiednich środowisk uruchomieniowych. W kontekście aplikacji w Spring Boot, React i PostgreSQL istotnymi elementami są Java, Node.js oraz Docker.

Java jest językiem programowania wykorzystywanym w Spring Boot. Dzięki niemu programiści mogą tworzyć rozbudowane aplikacje, które łatwo się skalują i są odporne na błędy. Java działa na wielu platformach, co pozwala na uruchomienie aplikacji na różnych systemach operacyjnych. Dodatkowo, język ten oferuje szeroki wybór narzędzi, które ułatwiają rozwijanie aplikacji.

Node.js z kolei jest środowiskiem uruchomieniowym, które pozwala na uruchamianie aplikacji w języku JavaScript poza przeglądarką internetową. W przypadku aplikacji w React, Node.js jest niezbędnym elementem, ponieważ pozwala na tworzenie aplikacji internetowych w oparciu o framework React. Node.js pozwala na łatwe zarządzanie zależnościami, automatyzację zadań i dostęp do wielu bibliotek i narzędzi.

Docker natomiast jest platformą, która umożliwia uruchomienie aplikacji w kontenerach. Dzięki temu programiści mogą uniknąć problemów z konfiguracją środowiska, które może być różne w zależności od systemu operacyjnego. Uruchomienie aplikacji w kontenerze Docker umożliwia przenoszenie aplikacji między różnymi środowiskami, co ułatwia zarządzanie aplikacją na różnych etapach cyklu życia aplikacji.

W kontekście aplikacji w Spring Boot, React i PostgreSQL, wykorzystanie tych trzech środowisk uruchomieniowych pozwala na łatwe tworzenie, rozwijanie i zarządzanie aplikacją. Java i Node.js umożliwiają tworzenie aplikacji, a Docker umożliwia uruchomienie aplikacji w kontenerze, co ułatwia przenoszenie aplikacji

między różnymi środowiskami. Dodatkowo, wykorzystanie tych środowisk uruchomieniowych umożliwia programistom dostęp do wielu narzędzi i bibliotek, które ułatwiają rozwijanie aplikacji oraz pozwala na łatwe skalowanie i zarządzanie aplikacją w różnych środowiskach.

1.2. Wykorzystane technologie

Aplikacja została stworzona w oparciu o następujące technologie:

➤ Warstwa serwerowa (backend):

- *Spring Boot* - to framework dla języka Java, który ułatwia i przyspiesza tworzenie aplikacji webowych. Spring Boot dostarcza gotowe narzędzia i biblioteki, które pozwalają na szybkie uruchomienie aplikacji bez potrzeby konfigurowania wielu szczegółów. Dzięki temu programiści mogą skupić się na tworzeniu logiki biznesowej zamiast zajmować się skomplikowaną konfiguracją infrastruktury. Spring Boot pozwala również na łatwe rozszerzanie i integrację z innymi narzędziami i bibliotekami, co czyni go popularnym wyborem wśród programistów Javy [1][6][3].
- *Hibernate* - to popularna biblioteka dla języka Java, która umożliwia programistom łatwe mapowanie obiektowo-relacyjne (ORM) między bazami danych a obiektami w programie. Hibernate pozwala na uniknięcie bezpośredniego korzystania z języka SQL i zapewnia elastyczność oraz skalowalność w zarządzaniu danymi. Biblioteka ta jest wykorzystywana w wielu aplikacjach webowych i mobilnych, które korzystają z relacyjnych baz danych. Hibernate umożliwia mapowanie klas Java na tabele w bazie danych i pozwala na łatwe zapisywanie, odczytywanie i modyfikowanie danych. Hibernate dostarcza również mechanizmy do obsługi transakcji, które gwarantują spójność danych. Dzięki temu programiści mogą skupić się na tworzeniu logiki biznesowej i nie muszą się martwić o detale dotyczące zarządzania danymi.
- *Liquibase* - to narzędzie open-source dla języka Java, które pozwala na kontrolowanie wersji baz danych oraz zarządzanie migracjami

schematu. Narzędzie to umożliwia łatwe wdrażanie i utrzymywanie zmian w strukturze bazy danych, a także zapewnia bezpieczeństwo i spójność danych. Liquibase pozwala na definiowanie zmian w bazie danych przy użyciu XML, YAML, JSON lub SQL. Dzięki temu programiści mogą łatwo kontrolować wersje struktury bazy danych i wdrażać zmiany w kontrolowany i zautomatyzowany sposób. Liquibase pozwala również na automatyczne wykrywanie zmian w schemacie i generowanie plików zmian, co ułatwia pracę programistom [8].

- *Lombok* - to biblioteka dla języka Java, która umożliwia programistom wygodne i szybkie tworzenie klas, eliminując potrzebę pisania powtarzalnego kodu. Biblioteka ta oferuje wiele adnotacji, które generują kod automatycznie, w tym metody dostępowe, konstruktory, metody `toString()`, `equals()` i `hashCode()`, a także gettery i settery. Lombok pozwala na zwiększenie produktywności programisty poprzez eliminowanie powtarzalnego kodu. Dzięki temu programiści mogą skupić się na tworzeniu logiki biznesowej, a nie na pisaniu metod dostępowych i innych powtarzalnych fragmentów kodu. Biblioteka ta również pomaga w utrzymaniu spójności kodu, ponieważ generowany kod jest zawsze zgodny z adnotacjami [7].
- *Spring Data JPA* - to projekt w ramach platformy Spring, który zapewnia wsparcie dla łatwej i wydajnej integracji JPA (Java Persistence API) z aplikacjami opartymi na Spring. JPA to standardowa specyfikacja Java EE do mapowania obiektowo-relacyjnego (ORM) w Javie. Spring Data JPA dodaje szereg funkcjonalności do JPA, takich jak automatyczne generowanie zapytań, transakcje i zarządzanie sesjami, co znacznie ułatwia pracę z bazami danych i upraszcza kodowanie. Spring Data JPA dostarcza również szereg udogodnień dla programistów, takich jak mechanizmy paginacji, sortowania i filtrowania danych.
- *Spring Security* – to projekt w ramach platformy Spring, który zapewnia narzędzia i mechanizmy do zabezpieczania aplikacji webowych. Jest to jedna z najpopularniejszych bibliotek bezpieczeństwa dla aplikacji Java [12].

Spring Security zapewnia mechanizmy uwierzytelniania i autoryzacji, które pozwalają na ochronę zasobów aplikacji przed nieuprawnionym dostępem. Uwierzytelnianie polega na weryfikacji tożsamości użytkownika, natomiast autoryzacja polega na kontrolowaniu dostępu użytkownika do określonych zasobów aplikacji.

➤ Warstwa kliencka (frontend):

- *React* - to biblioteka JavaScript, która służy do tworzenia interfejsów użytkownika. Jest to jedna z najpopularniejszych bibliotek frontendowych i jest wykorzystywana przez wiele firm i programistów na całym świecie [2].
- *React Bootstrap* - to biblioteka komponentów interfejsu użytkownika dla React, która integruje popularną bibliotekę CSS Bootstrap z frameworkiem React. React Bootstrap oferuje gotowe, łatwe w użyciu komponenty, takie jak przyciski, formularze, nawigacje, modale i wiele innych, które można wykorzystać w aplikacjach internetowych opartych na React [11].
- *React Hooks* - umożliwiają używanie stanu, efektów ubocznych (side effects), kontekstu, referencji do elementów DOM i innych funkcjonalności React bez konieczności tworzenia klasowych komponentów. Hooks są funkcjami, które pozwalają na "haczenie" się w mechanizmy wewnętrzne React, umożliwiając bardziej deklaratywne i łatwiejsze do zrozumienia komponenty funkcyjne.
- *Axios* – to biblioteka JavaScript służąca do wykonywania żądań HTTP z poziomu przeglądarki lub środowiska Node.js. Biblioteka umożliwia tworzenie asynchronicznych żądań sieciowych do serwera, pobieranie i przesyłanie danych oraz obsługę odpowiedzi.

➤ Baza danych:

- *Postgresql* - to system zarządzania relacyjnymi bazami danych (RDBMS), który został opracowany jako projekt open source. Jest to potężny, zaawansowany i wysoce skalowalny system bazodanowy,

który jest popularnym wyborem dla aplikacji internetowych i biznesowych, ze względu na swoją wydajność, niezawodność i rozbudowaną funkcjonalność [9].

1.3. Wykorzystane narzędzia w procesie tworzenia aplikacji

Stworzenie aplikacji wymagało użycia różnego rodzaju narzędzi takich jak:

- IntelliJ IDEA - IntelliJ IDEA to rozbudowane środowisko programistyczne stworzone przez firmę JetBrains, które oferuje zaawansowane narzędzia dla wielu języków programowania, w tym dla Javy, Kotlin, Groovy, Scala, Pythona i innych. Dzięki licznej ilości funkcji, takich jak podpowiadanie kodu, formatowanie kodu, debugowanie, refaktoryzacja, testowanie jednostkowe, integracja z narzędziami kontroli wersji oraz wsparcie dla różnych systemów budowania projektów, IntelliJ IDEA jest bardzo popularne wśród programistów. IDE oferuje łatwy w użyciu interfejs użytkownika oraz jest dostępne w dwóch wersjach - płatnej, a także bezpłatnej Community Edition, co umożliwia korzystanie z narzędzi IDE bez ponoszenia kosztów licencyjnych. IntelliJ IDEA zapewnia nie tylko szybkie i efektywne środowisko programistyczne, ale także pozwala na łatwe zarządzanie projektami, dzięki czemu programiści mogą skupić się na tworzeniu wysokiej jakości kodu.
- IntelliJ Diagrams – jest to rozszerzenie(plugin) dla środowiska IntelliJ IDEA stworzony przez JetBrains, które umożliwia łatwe generowanie diagramów UML oraz innych rodzajów diagramów wewnątrz samego IDE.
- Visual Studio Code – to darmowe, zintegrowane środowisko programistyczne (IDE) opracowane przez firmę Microsoft. Jest to bardzo popularne narzędzie programistyczne, które oferuje zaawansowane funkcje dla wielu języków programowania, takich jak JavaScript, TypeScript, Python, PHP, C++, C#, Go i wiele innych,

- DBeaver – to narzędzie do zarządzania bazami danych, które oferuje wiele funkcji dla programistów i administratorów baz danych. DBeaver umożliwia łatwe zarządzanie, organizowanie, analizowanie i wizualizowanie danych w wielu popularnych bazach danych, takich jak MySQL, PostgreSQL, Oracle, SQLite, MongoDB i wiele innych. DBeaver oferuje wiele funkcji, takich jak edycja danych, eksploracja bazy danych, wykonanie zapytań SQL, import i eksport danych, synchronizacja struktury bazy danych, wizualne tworzenie zapytań,
- Docker - to platforma służąca do wirtualizacji kontenerowej aplikacji. Dzięki Dockerowi programiści i administratorzy systemów mogą łatwo tworzyć, uruchamiać i zarządzać aplikacjami w izolowanych środowiskach kontenerów. Docker pozwala na przeniesienie aplikacji między różnymi środowiskami i systemami operacyjnymi bez potrzeby zmiany kodu, dzięki czemu programiści mogą szybko i łatwo przetestować i wdrożyć swoje aplikacje na różnych platformach.

2. Koncepcja portalu

W celu stworzenia efektywnego i przyjaznego dla użytkowników portalu internetowego wspomagającego opiekę nad zwierzętami, konieczne jest precyzyjne określenie jego założeń funkcjonalnych i нефunkcjonalnych. Założenia funkcjonalne będą obejmować wszystkie funkcjonalności, jakie powinny być dostępne dla użytkowników portalu, natomiast założenia нефunkcjonalne będą obejmować aspekty techniczne i ogólne, takie jak szybkość działania portalu, jego dostępność i bezpieczeństwo.

2.1. Założenia funkcjonalne i нефunkcjonalne

W ramach projektowania i implementacji aplikacji, niezwykle ważnym etapem jest określenie jej założeń funkcjonalnych i нефunkcjonalnych. Założenia funkcjonalne określają konkretne funkcje, które aplikacja powinna wykonywać, natomiast założenia нефunkcjonalne opisują cechy, jakie powinna posiadać aplikacja, aby działała efektywnie, niezawodnie i zgodnie z oczekiwaniami użytkowników.

➤ Założenia funkcjonalne

- Aplikacja umożliwia dodawanie, usuwanie, aktualizowanie i pobieranie danych z bazy danych PostgreSQL.
- Użytkownicy aplikacji mogą przeglądać listę danych, wyszukiwać po kluczowych słowach,
- Aplikacja umożliwia przetwarzanie danych w czasie rzeczywistym i natychmiastowe odzwierciedlanie zmian w bazie danych w interfejsie użytkownika.
- Użytkownicy aplikacji mają możliwość logowania i rejestracji, co pozwala na przechowywanie danych użytkownika w bazie danych i udzielanie dostępu tylko zalogowanym użytkownikom.
- Każdy użytkownik może udostępniać informacje o ciekawych momentach w życiu ich zwierząt.

- Użytkownicy aplikacji mogą nawiązywać znajomości z innymi użytkownikami i dowiadywać się więcej informacji na temat ich zwierząt.
- Aplikacja daje możliwość agregowania informacji na temat swoich zwierząt oraz planować wydarzenia związane z nimi.
- Użytkownicy mogą dzielić się swoimi doświadczeniami na forum.
- Każda osoba w aplikacji może założyć zbiórkę na określony przez nią cel.
- Użytkownicy mogą brać udział w konkursach organizowanych dla społeczności oraz rozwiązywać przygotowane quizy na określony temat.

➤ Założenia niefunkcjonalne

- Aplikacja działa płynnie i responsywnie, nawet przy dużych ilościach danych w bazie danych.
- Baza danych PostgreSQL jest skalowalna i wydajna, co pozwala na przechowywanie i przetwarzanie dużych ilości danych.
- Dane przechowywane w bazie danych są bezpieczne i chronione przed nieautoryzowanym dostępem.
- Aplikacja jest łatwa w utrzymaniu i rozwijaniu, co pozwala na szybkie wprowadzanie zmian i poprawek.
- Wymagania środowiskowe aplikacji :

Wymagania środowiskowe są kluczowym elementem, który należy wziąć pod uwagę przed uruchomieniem aplikacji, zwłaszcza w przypadku, gdy korzysta się z różnych technologii, takich jak Spring Boot i React.

Minimalne wymagania sprzętowe:

- procesor: 2.5 GHz lub szybszy
- pamięć RAM: 4 GB lub więcej
- wolna przestrzeń dyskowa: 2 GB

Minimalne wymagania środowiskowe dla Spring Boot:

- Java 11 lub wyższa

- Gradle
- Spring Boot 2.4.0 lub wyższa

Minimalne wymagania środowiskowe dla React:

- Node.js 10.16 lub nowszy
- NPM 5.6 lub nowszy

System operacyjny:

Aplikacja w Spring Boot z wykorzystaniem React może działać na różnych systemach operacyjnych, w tym na Windows, macOS i Linux.

- Założenia technologiczne:

W pracy zdecydowano się na wykorzystanie szeregu nowoczesnych technologii do stworzenia działającej aplikacji. W tym rozdziale zostaną opisane narzędzia i rozwiązania, które przyczyniły się do powstania finalnego produktu.

Aplikacja składa się z dwóch głównych części: backendu, czyli części działającej po stronie serwera oraz frontendu, czyli interfejsu użytkownika. Backend został zaimplementowany z wykorzystaniem frameworka Spring Boot, który zapewnił solidne i wydajne rozwiązanie do budowy serwerowej części aplikacji. Frontend natomiast został stworzony w oparciu o React - popularny i zaawansowany framework do tworzenia interfejsów użytkownika. Dzięki temu użytkownicy otrzymali intuicyjny i przyjazny interfejs, który działał szybko i sprawnie.

Aby przechowywać dane, zdecydowano się na wykorzystanie jednego z najpopularniejszych systemów do zarządzania relacyjnymi bazami danych - Postgresql. Dzięki temu aplikacja mogła przechowywać duże ilości danych w formie tabelarycznej i zapewnić bezpieczny oraz szybki dostęp do informacji.

Wybór powyższych technologii był podyktowany ich niezawodnością, popularnością oraz wydajnością. Dzięki temu udało się stworzyć solidną i stabilną aplikację, która spełniła oczekiwania użytkowników i zapewniła niezbędną funkcjonalność.

3. Projekt i architektura warstwowa aplikacji

Projektowanie aplikacji to kluczowy proces, który decyduje o jej jakości, skalowalności i efektywności. Jednym z popularnych podejść projektowych jest architektura warstwowa, która pozwala na rozdzielenie funkcjonalności aplikacji na niezależne od siebie warstwy. Dzięki temu aplikacja staje się bardziej modułowa i łatwiejsza w utrzymaniu. W tym rozdziale omówimy projekt i architekturę warstwową aplikacji, jej zalety oraz zastosowanie w praktyce. Przedstawimy również przykładową implementację architektury warstwowej w naszej aplikacji dla miłośników zwierząt.



Rysunek 3 - Diagram modeli

3.1. Projekt bazy danych i wymagania środowiskowe

Celem niniejszego rozdziału jest przedstawienie struktury danych, które są wykorzystywane przez aplikację w trakcie jej działania. Do przechowywania informacji wykorzystywana jest relacyjna baza danych, która umożliwia organizację danych w tabelach połączonych ze sobą relacjami, reprezentowanymi przez klucze główne i obce. Wybór tego typu bazy danych został podyktowany specyficznymi wymaganiami aplikacji. W kolejnej części rozdziału zostanie przedstawiony sposób, w jaki dane są przechowywane w bazie danych.

3.1.1. Opis encji i relacji

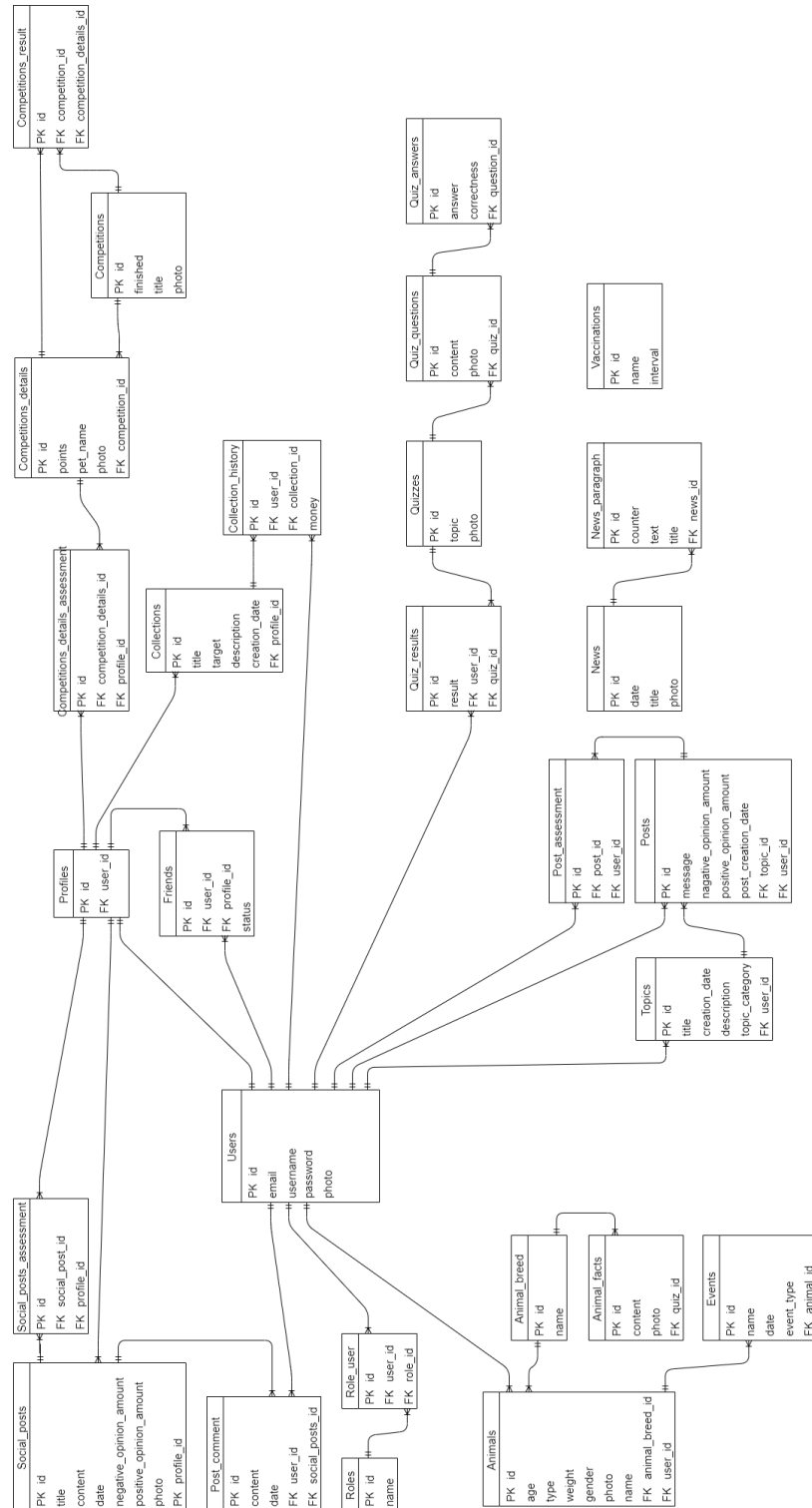
Baza danych z której korzysta przedstawiona aplikacja składa się z wielu tabel. Ze względu na złożoność bazy zostaną opisane najważniejsze z nich.

- „*users*” – tabela ta zawiera informacje o wszystkich użytkownikach portalu. Tak jak każda tabela w bazie posiada swój indywidualny ID generowany automatycznie. Każdy użytkownik w bazie ma swoją nazwę którą posługuje się podczas korzystania z portalu, adres email, hasło do logowania oraz zdjęcie.
- „*profiles*” – przechowuje ona informacje o profilu, który jest przypisany do każdego użytkownika.
- „*animals*” - znajdują się tutaj zwierzęta, które przynależą do użytkowników aplikacji wraz z informacjami o nich samych np. imię, wiek, waga, rasa.
- „*animal_breed*” - przechowuje dostępne rasy. Każda rasa ma charakterystyczne ciekawostki na swój temat, które są przechowywane w osobnej tabeli „*animal_facts*”,
- „*events*” - tabela ta zawiera informacje o wydarzeniach, które odbyły się lub mają się odbyć. Są one podzielone na typy, a każdy typ reprezentuje rodzaj wydarzenia. Ponadto przechowywana jest data w której dane wydarzenie ma się odbyć oraz jego nazwa,
- „*vaccinations*” - tabela ta ma za zadanie przechowywać szczepienia, które są proponowane przez aplikacje dla danego zwierzęcia. Szczepienia te mają indywidualny interwał czasowy w którym mogą się odbywać. W celu uniknięcia problemów z błędami w pisowni nazwy interwałów są typu wyliczeniowego,
- „*topics*” - tabela ta przechowuje informacje o pytaniach zadanych na forum innym użytkownikom portalu. Przechowywane tu informacje to między innymi temat danego wątku, który umożliwia potencjalnemu zainteresowanemu dotarcie do ciekawych dla niego treści oraz opis poruszanego problemu. Każdy temat ma przypisane posty stanowiące konwersacje pomiędzy użytkownikami.
- „*social_posts*”- dane w tej tabeli opisują posty dodawane przez użytkowników. Posty takie są informacją dla innych osób o istotnych momentach z życia pupila takie jak spacer, szczepienia itp.

- „*post_comment*” - post dodany przez osobę na swoim profilu może zostać skomentowany przez inne osoby, które mogą wyrazić swoją opinie o danym poście,
- „*collections*” - tabela ta przechowuje informacje o założonych przez użytkowników zbiorach,
- „*competitions*” - znajdują się tutaj informacje o konkursach organizowanych w ramach aplikacji,
- „*animal_facts*” - w tej tabeli przechowywane są fakty i ciekawostki o zwierzętach. Każda informacja w tabeli jest powiązana z rasą której bezpośrednio dotyczy,
- „*quizzes*” - tabela ta przechowuje tematy quizów, które są dostępne do rozwiązywania przez użytkowników. Tabela „*quizzes*” jest powiązana również z tabelą przechowującą pytania, która to połączona jest również z możliwymi odpowiedziami dotyczącymi pytania z oznaczeniem, która jest poprawna,
- „*news*” - znajdują się tutaj artykuły związanych z nowościami w tematyce zwierząt domowych.

3.1.2. Diagram ERD

Na poniższym diagramie została przedstawiana graficzna prezentacja obiektów i relacji zachodzących w bazie danych.



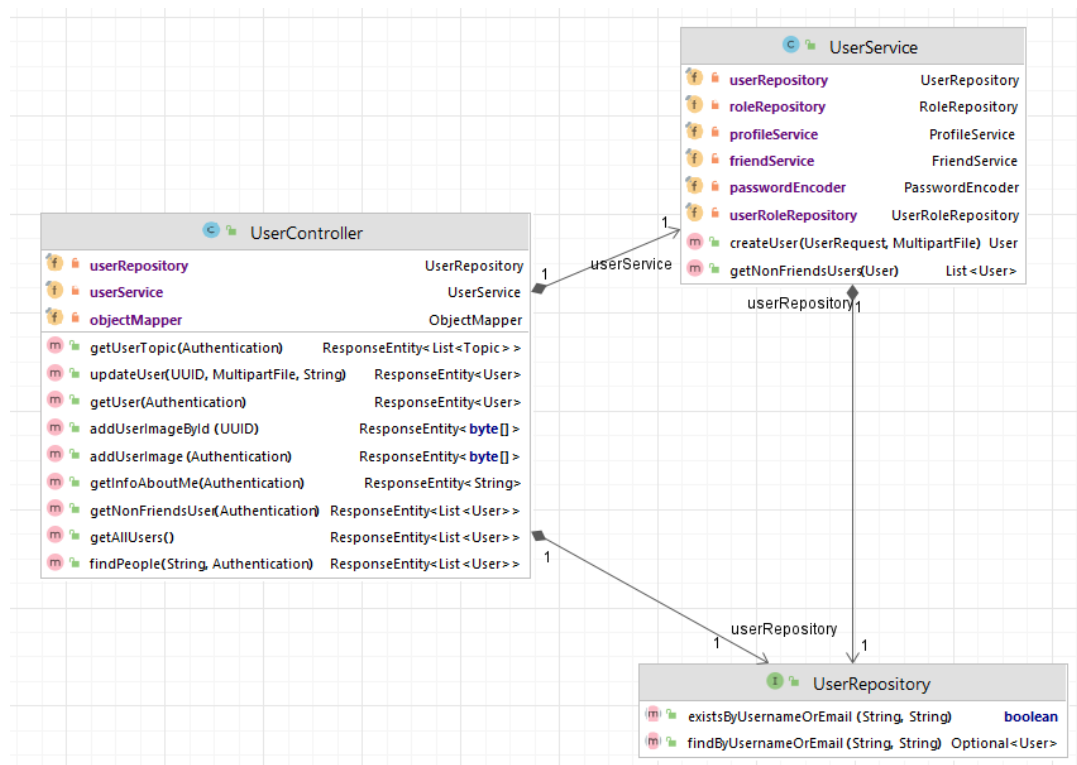
Rysunek 4 - Diagram ERD bazy danych

3.2. Schemat organizacyjny projektu

W niniejszym rozdziale zaprezentujemy szczegółowy schemat organizacyjny naszego projektu, który obejmuje diagramy klas warstwy biznesowej oraz modelów, diagram organizacyjny portalu oraz diagram przypadków użycia.

3.2.1. Diagram klas warstwy biznesowej

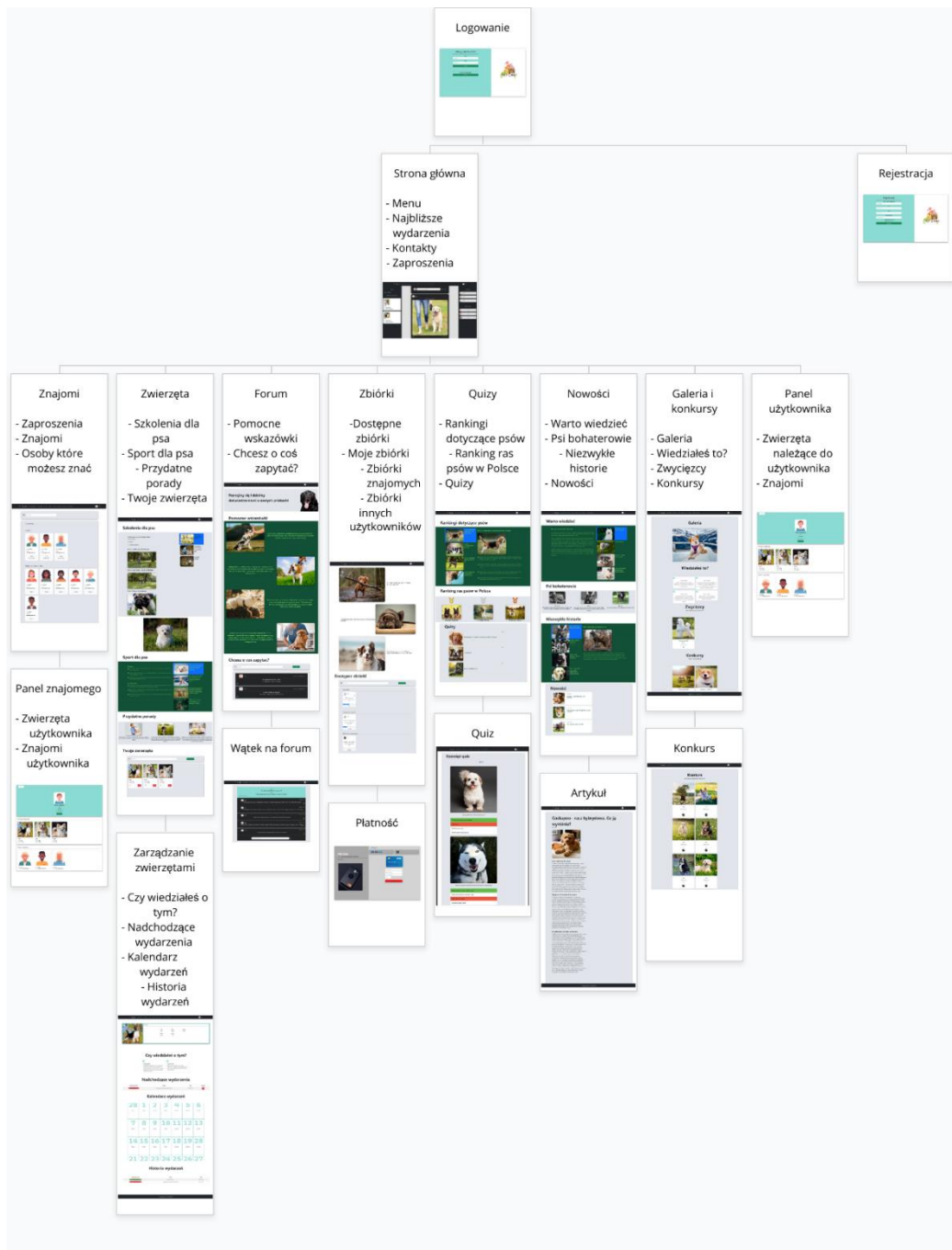
Ze względu na liczebność klas i relacji pomiędzy nimi przedstawiono tylko ich fragment.



Rysunek 5 - Przykładowy diagram klas stworzony dla warstwy biznesowej aplikacji

3.2.2. Diagram organizacyjny portalu

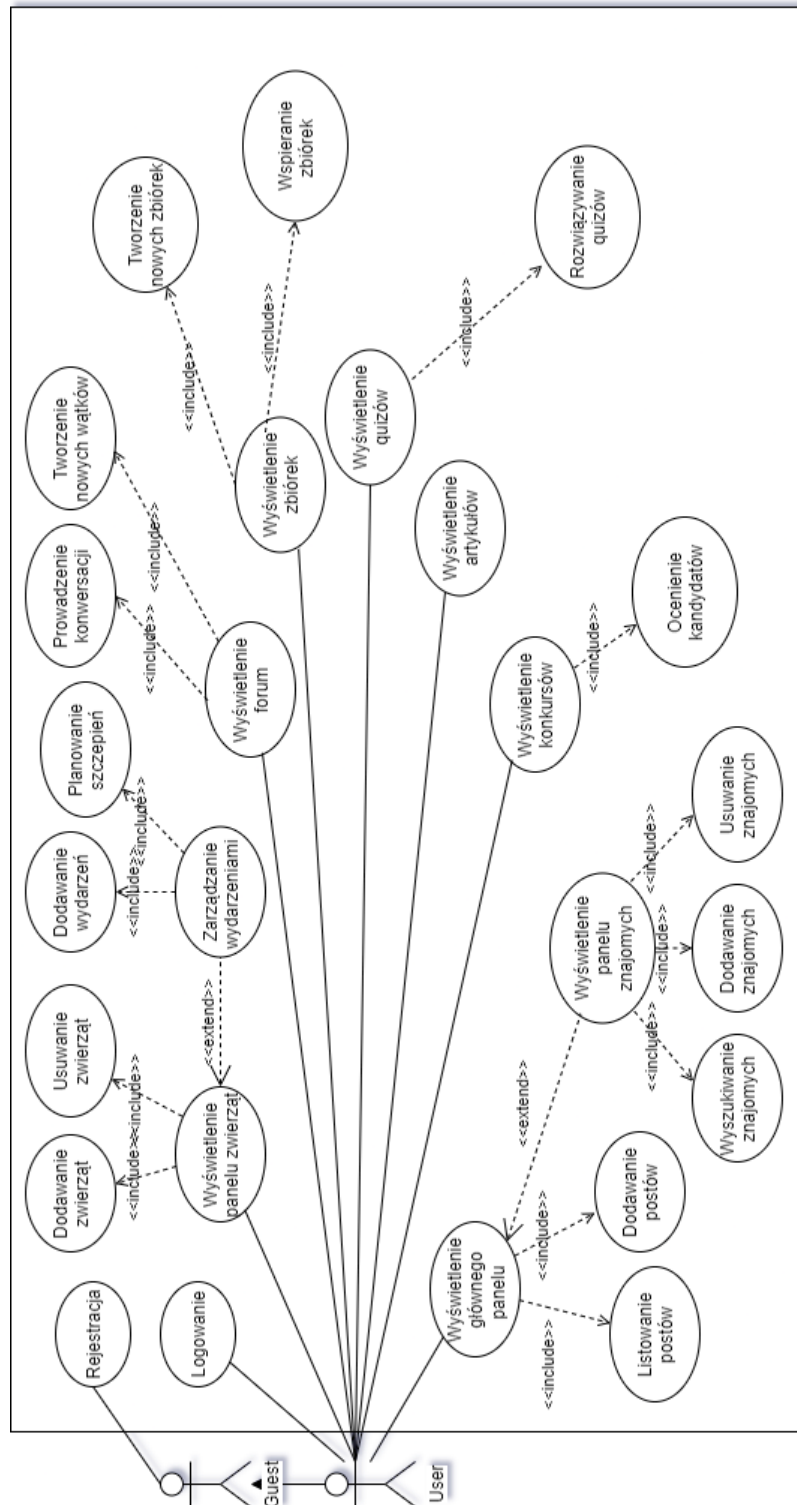
Diagram przedstawiam wizualną prezentację portalu internetowego z uwzględnieniem dostępnych widoków.



Rysunek 3 - Diagram organizacyjny portalu

3.2.3. Diagram przypadków użycia

Na poniższym diagramie UML przedstawiono dostępne funkcjonalności dla roli użytkownika i gościa w oparciu o obowiązujące standardy [10].



Rysunek 4 - Diagram przypadków użycia dla roli "Guest" i "User"

Aplikacja posiada dwa poziomy dostępu:

- gość (guest)
- użytkownik (user)

Każda w wyżej wymienionych roli posiada indywidualny dostęp do różnych danych.

- gość,
 - rola gościa w aplikacji ogranicza się do rejestracji,
- użytkownik
 - logowanie,
 - wyświetlanie głównego panelu z postami społecznościowymi,
 - wyświetlanie i zarządzanie znajomymi,
 - zarządzanie zwierzętami przy użyciu odpowiedniego panelu,
 - korzystanie z forum społeczności,
 - wyświetlanie i wspieranie zbiórek charytatywnych,
 - rozwiązywanie quizów,
 - wyświetlanie tematycznych artykułów,
 - głosowanie w zorganizowanych konkursach.

3.3. Warstwa persystencji

Tematem tego rozdziału będzie warstwa persystencji czyli nieodłączna część aplikacji wykorzystującej bazę danych. Warstwa ta odpowiada za operacje na bazie danych takie jak zapisywanie danych do bazy, ich odczyt lub aktualizowanie.

Spring Data JPA (*Java Persistence API*) jest narzędziem rozszerzającym możliwości frameworka Spring Boot. Został stworzony w celu ułatwiania pracy programisty poprzez zmniejszenie ilości kodu, który należy napisać w celu obsługi bazy danych. Wiele czynności dzieje się automatycznie po stronie frameworka Spring. Konfiguracje Spring Data JPA należy rozpocząć poprzez dodanie odpowiedniej zależności do pliku konfiguracyjnego czyli „*spring-boot-starter-data-jpa*”, następnie do pliku konfiguracyjnego trzeba dodać adnotację „*@EnableJpaRepository*”, która dokona skanowania klas repozytoriów.

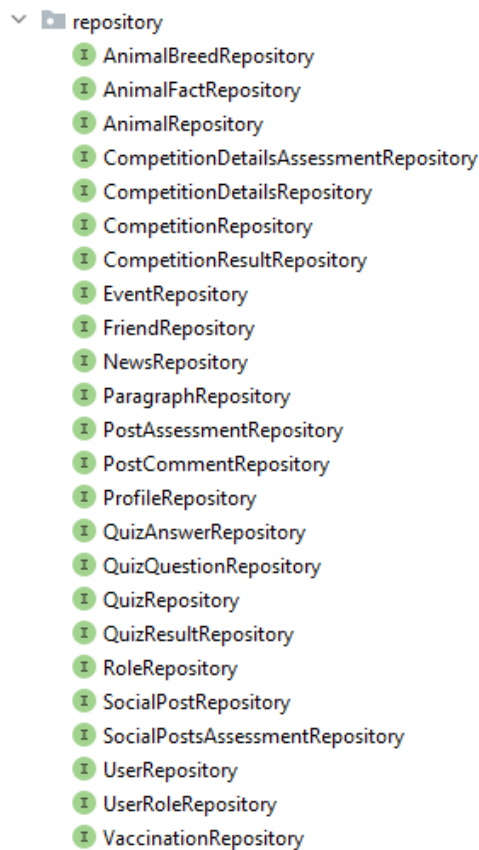
Ostatnim krokiem jest podanie informacji o bazie danych, takich jak np. nazwa bazy danych, adres serwera bazy danych, nazwę użytkownika i hasło. Mapowanie obiektowo-relacyjne (ORM) jest kluczowym procesem w projektowaniu aplikacji, które korzystają z relacyjnych baz danych. ORM pozwala na bezpośrednie odwzorowanie encji (czyli klas reprezentujących obiekty w aplikacji) na tabelki w bazie danych. Proces ten może być wykonywany ręcznie, lecz zdecydowanie łatwiej i szybciej jest skorzystać z narzędzi ORM, takich jak Hibernate, JPA, czy Spring Data JPA. Spring Data JPA jest jednym z popularnych frameworków ORM w języku Java, który zapewnia wiele gotowych rozwiązań do mapowania encji na tabele w bazie danych. W celu przeprowadzenia procesu mapowania, należy użyć adnotacji dostarczanych przez Spring Data JPA. Oto kilka z najczęściej używanych adnotacji:

- „@Entity” – adnotacja ta określa klasę jak encję, która zostanie zapisana w bazie danych,
- „@Table” – adnotacja ta wraz z parametrem „name” określa nazwę tabeli w bazie danych,
- „@Column” – dodanie tej adnotacji pozwoli na określenie nazwy kolumny w tabeli, która będzie odpowiadać polu klasy encji,
- „@Id” – reprezentuje ona pole w klasie jako klucz główny (primary key).

Wykorzystanie adnotacji umożliwia łatwe i automatyczne mapowanie pól encji na kolumny w bazie danych, a także ułatwia zarządzanie relacjami między encjami.

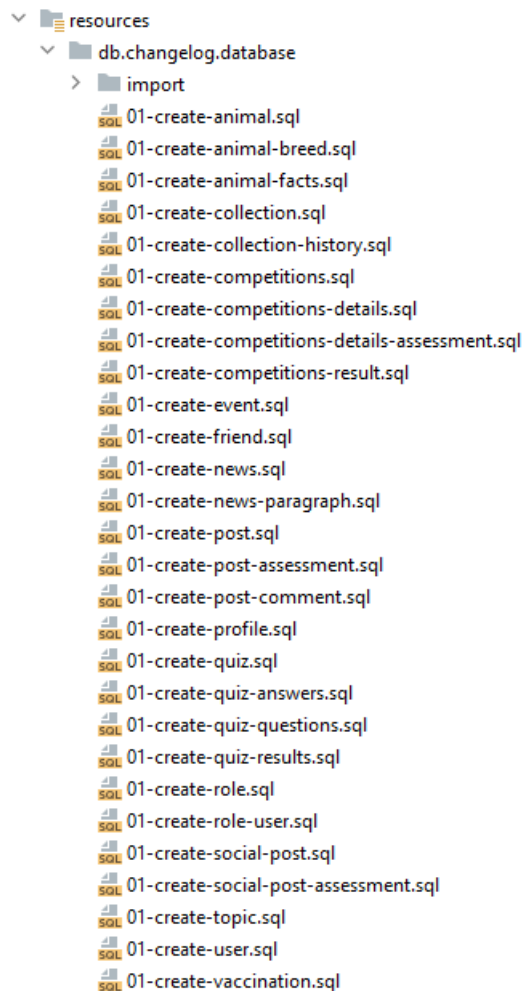
Operacje CRUD (Create, Read, Update, Delete) są podstawowymi czynnościami, które możemy wykonywać na bazie danych. W przypadku Spring Data JPA operacje te są wykonywane przez repozytoria, które rozszerzają interfejs `JpaRepository`. Daje to dostęp do metod umożliwiających zapisywanie obiektu do bazy czyli metoda „`save()`”, ale i usuwanie poprzez metodę „`delete()`”. Otrzymujemy również możliwość przeszukiwania dostępnego repozytorium poprzez metody takie jak „`findById()`” lub „`findAll()`” itp.

Aplikacja przedstawiona w niniejszej pracy wykorzystuje szereg tego typu interfejsów.



Rysunek 5 - Przykładowa struktura repozytoriów

Stworzenie schematu bazy danych można dokonać na parę sposobów. Jednym z nich jest wygenerowanie go przy użyciu Hibernate, który to jest w stanie na podstawie zdefiniowanych encji stworzyć tabele i kolumny natomiast jest to rozwiązanie mało bezpieczne. Lepszym rozwiązaniem jest użycie do tego celu Liquibase, który na podstawie kody SQL stworzy nam odpowiedni schemat bazy danych.



Rysunek 6 - Struktura plików Liquibase

3.4. Warstwa kontrolera

Warstwa kontrolera jest jednym z kluczowych elementów architektury aplikacji internetowych. Jej głównym zadaniem jest obsługa żądań HTTP przesyłanych przez klienta i przekierowanie ich do odpowiednich metod kontrolera.

Żądanie HTTP jest protokołem komunikacyjnym wykorzystywanym w Internecie do wymiany danych między serwerem a klientem. Warstwa kontrolera jest odpowiedzialna za obsługę żądań HTTP, co oznacza, że po odebraniu żądania, warstwa ta analizuje jego parametry i zwraca odpowiednią odpowiedź.

Kontroler to klasa, która zawiera metody, które są wywoływane w odpowiedzi na żądania HTTP. Metody te zawierają logikę biznesową aplikacji, która pozwala na przetworzenie żądania i zwrócenie odpowiedzi do klienta.

Aby wykonać żądanie HTTP, konieczne jest przesłanie odpowiedniego typu metody (np. GET, POST, DELETE) oraz prawidłowego adresu. Metoda definiuje, co dokładnie chcemy zrobić z zasobem, a adres określa, który zasób ma być przetwarzany.

W celu przetworzenia żądania, warstwa kontrolera powinna być wyposażona w odpowiednie metody, które zostaną wywołane w zależności od typu żądania HTTP. Metody te są oznaczane odpowiednimi adnotacjami, takimi jak np. "@GetMapping" czy "@PostMapping". Adnotacje te określają, jakie żądanie HTTP zostanie obsłużone przez daną metodę kontrolera.

W przypadku, gdy metoda kontrolera wymaga przekazania dodatkowych parametrów, takich jak ciało (w postaci obiektu JSON), parametr lub zmienna w adresie URL, kontroler musi być wyposażony w dodatkowe elementy obsługi danych. Przykładowo, jeśli chcemy przekazać drobną informację bez przesyłania całego ciała, możemy to zrobić przez przekazanie odpowiedniej zmiennej w adresie URL.

3.5. Warstwa serwisowa

Warstwa serwisowa stanowi kluczowy element prezentowanej aplikacji, który odpowiada za przetwarzanie żądań otrzymanych przez kontrolery. Głównym celem tej warstwy jest przetwarzanie logiki biznesowej i wyodrębnienie jej z kontrolera w celu uzyskania bardziej przejrzystego i modułowego kodu.

W kontekście prezentowanej aplikacji, serwisy są wykorzystywane do przetwarzania logiki, która jest odpowiedzialna za wiele mechanizmów, w tym sprawdzanie, czy w najbliższym czasie powinno się przeprowadzić szczepienie zwierzęcia. Choć taką logikę można byłoby zaimplementować bezpośrednio w kontrolerze, stosowanie osobnej warstwy serwisów jest uważane za dobry

zwyczaj. Dzięki temu uzyskuje się bardziej czytelny i przejrzysty kod w warstwie kontrolera, a także minimalizuje się redundancję kodu.

Warstwa serwisowa umożliwia również wielokrotne wykorzystanie tych samych reguł biznesowych w różnych miejscach aplikacji, co zapewnia spójność i jednoznaczność działania. Dzięki temu łatwiej jest utrzymać aplikację i wprowadzać do niej zmiany. Warto zauważyć, że stosowanie osobnej warstwy serwisów nie tylko ułatwia pracę z kodem, ale także ułatwia testowanie aplikacji, co z kolei przekłada się na jakość końcowego produktu.

3.6. Warstwa bezpieczeństwa

Bezpieczeństwo w aplikacji, która przechowuje wrażliwa dane jest szczególnie ważne aby nikt niepowołany nie mógł mieć dostępu do danych jej użytkowników. Tworząc aplikacje z wykorzystaniem frameworka Spring Boot mamy do dyspozycji kilka rozwiązań, które są dostarczone w ramach modułu Spring Security. Moduł ten dostarcza wiele mechanizmów chroniących aplikację przed atakami i nieuprawnionym dostępem.

W celu dobrego zabezpieczenia przedstawianej aplikacji użyto popularnego wśród aplikacji internetowych i mobilnych standardu JWT (JSON Web Token). Standard ten umożliwia generowanie zaszyfrowanego tokenu, których przechowuje informacje o użytkowniku, jego roli oraz czasie przez jaki będzie jeszcze aktywny. Token JWT jest generowany podczas logowania do serwisu i jest wysyłany z każdym żądaniem. Po otrzymaniu żądania aplikacja sprawdza czy token jest prawidłowy, czy dany użytkownik ma uprawnienia do żądanego zasobu, czy token już nie wygasł oraz czy token nie znajduje się na czarnej liście tokenów. Lista taka jest stworzona po to aby po wylogowaniu token znajdujący się po stronie klienta nie mógł zostać użyty do niepowołanego dostępu do serwisu.

3.7. Warstwa wizualna (React)

Warstwa wizualna jest jednym z najważniejszych elementów każdej aplikacji internetowej. Odpowiada za wygląd i interakcję użytkownika z aplikacją.

React jest popularnym frameworkiem JavaScript, który umożliwia tworzenie dynamicznych i interaktywnych interfejsów użytkownika.

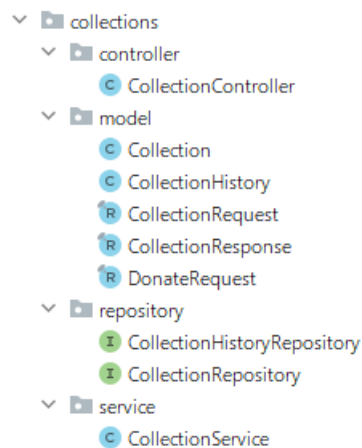
React opiera się na idei komponentów, czyli odizolowanych części interfejsu użytkownika, które można łączyć w bardziej złożone struktury. Komponenty React są łatwe w użyciu i pozwalają na tworzenie bardziej czytelnego i zorganizowanego kodu. Implementacja interfejsu użytkownika to proces tworzenia widoków, czyli elementów interfejsu użytkownika, z którymi użytkownik może interagować. W React implementacja interfejsu użytkownika odbywa się poprzez tworzenie komponentów. Komponenty React mogą mieć wiele różnych funkcji, takich jak wyświetlanie treści, obsługa zdarzeń użytkownika, pobieranie danych z serwera czy wyświetlanie komunikatów o błędach. Każdy komponent ma swoje własne właściwości (props) oraz stan (state), który może być zmieniany w trakcie działania aplikacji. Aby interfejs użytkownika mógł pobierać i wysyłać dane do serwera, konieczna jest komunikacja z warstwą serwisową. W architekturze REST (Representational State Transfer) do komunikacji między aplikacją a serwerem wykorzystuje się tzw. API (Application Programming Interface). Aby ułatwić tworzenie żądań sieciowych w aplikacji React, często wykorzystuje się bibliotekę "axios". Jest to popularna biblioteka do wykonywania żądań HTTP, która oferuje wiele dodatkowych funkcjonalności, które ułatwiają pracę z API REST. Jedną z głównych zalet biblioteki "axios" jest to, że umożliwia ona automatyczne przetwarzanie danych wejściowych i wyjściowych. Dzięki temu można łatwo wysyłać i odbierać dane w formacie JSON, a biblioteka zajmuje się ich automatycznym przetwarzaniem na poziomie kodu. Inną ważną funkcjonalnością biblioteki "axios" jest obsługa błędów. W przypadku, gdy żądanie sieciowe kończy się niepowodzeniem, biblioteka umożliwia łatwe przechwytywanie i obsługę błędów, co ułatwia debugowanie aplikacji. Dodatkowo, biblioteka "axios" oferuje wiele funkcjonalności związanych z uwierzytelnianiem, takich jak automatyczne dołączanie nagłówków uwierzytelniających, czy obsługa tokenów JWT.

4. Implementacja projektu

W tym rozdziale przedstawiona zostanie implementacja projektu, który składa się z trzech głównych elementów: backendu w Spring Boot, frontendu w React oraz bazy danych w Postgresql. Opisane zostaną szczegóły implementacyjne każdego z tych elementów, a także omówione zostaną decyzje projektowe, jakie podjęto w trakcie implementacji.

4.1. Implementacja backendu w Spring Boot

Warstwa serwerowa została zrealizowana przy użyciu frameworka Spring Boot oraz języka Java w środowisku IntelliJ IDEA. Aplikacja po stronie serwerowej składa się z kilku istotnym elementów współpracujących ze sobą, takie jak kontrolery REST API, serwisy, repozytoria, modele danych.



Rysunek 7 - Przykładowa struktura plików po stronie serwera

Rolą kontrolera jest obsługiwanie żądań klienta oraz zwracanie odpowiedzi HTTP. Do tego celu potrzebne jest zdefiniowanie metod HTTP (Rysunek 11).

```
@GetMapping("/{id}/animals/upcoming")
public ResponseEntity<List<EventResponse>> getUpcomingEvents(@PathVariable("id") UUID id){
    var list :List<EventResponse> = eventService.getUpcomingEvents(id);
    return ResponseEntity.ok(list);
}
```

Rysunek 8 - Przykład metody HTTP

Wewnątrz metody skorzystano w innej metody, która znajduje się w warstwie biznesowej czyli w serwisie. Metoda ta jest odpowiedzialna za wykonanie zadania i zwrócenie oczekiwanych danych do warstwy kontrolera (Rysunek 12).

```
public List<EventResponse> getUpcomingEvents(UUID id) {  
    var eventList : List<Event> = eventRepository.findByAnimalId(id);  
    List<Event> matchingEvents = new ArrayList<>();  
    for (Event event : eventList) {  
        Calendar calendar = Calendar.getInstance();  
        calendar.setTime(event.getDate());  
  
        if (getCalendarWithoutTime(Calendar.getInstance().getTime()).before(getCalendarWithoutTime(calendar.getTime()))) {  
            matchingEvents.add(event);  
        }  
    }  
    matchingEvents.sort(Comparator.comparing(Event::getDate));  
    return matchingEvents.stream().map(e -> new EventResponse(e.getDate().toString(), e, e.getAnimal())).toList();  
}
```

Rysunek 9 - Przykładowa metoda w warstwie biznesowej

4.2. Implementacja frontendu w React

Warstwa kliencka została zaimplementowana przy użyciu frameworka React. Aplikacja korzysta z komponentów, która składają się na widoki w aplikacji (Rysunek 13).



Rysunek 10 - Przykładowe komponenty po stronie klienta

Komponentem można nazwać fragment interfejsu użytkownika odpowiedzialnego za wyświetlanie określonej części aplikacji. Do pobierania danych z API zostały zdefiniowane serwisy (Rysunek 14).



Rysunek 11 - Przykładowe serwisy po stronie klienta

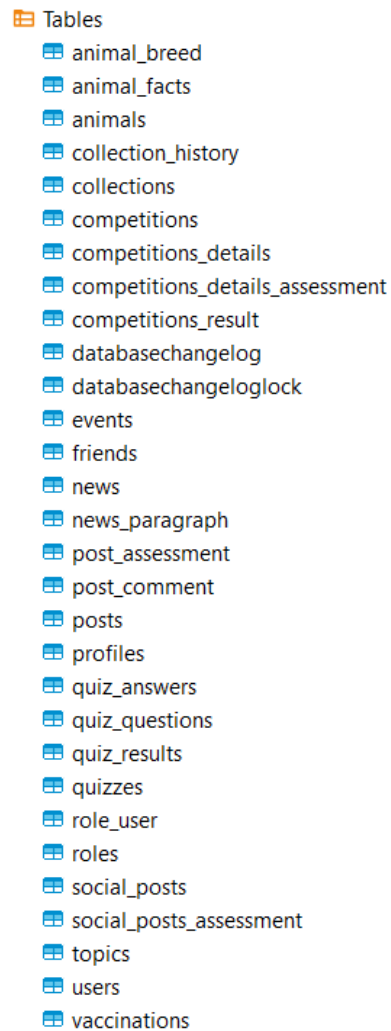
Każdy serwis posiada metody, które umożliwiają komunikację z API. Przykładem takiej metody jest ta przedstawiona na rysunku 15. Ma ona za zadanie pobrać nadchodzące wydarzenia dla zwierząt użytkownika.

```
const getUpcomingEvents = () => {  
  return axios.get(API_URL + "/events/animals/upcoming", {  
    headers: { Authorization: "Bearer " + localStorage.getItem("userToken") },  
  });  
};
```

Rysunek 12 - Przykład metody z serwisu

4.3. Implementacja bazy danych w Postgresql

Baza danych Postgresql jest uruchomiona z wykorzystaniem narzędzia Docker, co pozwala na łatwe zarządzanie bazą oraz uniezależnienie jej od systemu operacyjnego, na którym działa aplikacja. Można nią zarządzać z poziomu programu DBeaver, natomiast za stworzenie struktury bazy odpowiedzialny jest Liquibase, który na podstawie zdefiniowanych schematów (Rysunek 9) generuje zadaną bazę. Pozwala to na zapewnienie kontroli nad strukturą tworzonej bazy, w przeciwieństwie do automatycznego generowania jej z wykorzystaniem narzędzia Hibernate.



Rysunek 13- Tabele w bazie danych

Konfiguracja narzędzia Docker została zamieszczona w pliku konfiguracyjnym po stronie serwera aplikacji. Wywołanie komendy „docker-compose up” spowoduje utworzenie kontenera z bazą danych.

5. Konfiguracja i pierwsze uruchomienie aplikacji

W tym rozdziale omówimy kroki potrzebne do skonfigurowania aplikacji w Spring Boot z wykorzystaniem React oraz przeprowadzimy pierwsze uruchomienie aplikacji.

5.1. Konfiguracja kontenera Postgresql

Aby skorzystać z Postgresql możemy zainstalować go na swoim komputerze lub skorzystać z narzędzia jakim jest docker-compose. Docker-compose umożliwia uruchamianie aplikacji w izolowanych środowiskach tzw. kontenerach. Dzięki temu, nie jest konieczne instalowanie Postgresql na komputerze lokalnym, a samo uruchomienie bazy danych sprowadza się do wykonania kilku poleceń w konsoli. Aby uruchomić Postgresql z wykorzystaniem docker-compose należy przygotować plik konfiguracyjny „docker-compose.yml” dzięki któremu możemy określić wszystkie wymagane parametry. Po przygotowaniu pliku konfiguracyjnego i zamieszczeniu go w głównym folderze projektu możemy z wykorzystaniem terminala stworzyć kontener używając do tego celu komendy „docker-compose up”.

5.2. Zbudowanie projektu Spring Boot

Jeśli spełniamy wymagania środowiskowe związane z aplikacją możemy przystąpić do uruchomienia części serwerowej. Pierwszą czynnością będzie przejście do katalogu w którym znajduje się plik „build.gradle”, a następnie wykonanie w terminalu komendy „gradle build” spowoduje to zbudowanie aplikacji oraz zainstalowanie jej zależności. Wynikiem zbudowania aplikacji będzie plik JAR. W katalogu w którym znajduje się plik JAR należy uruchomić komendę „java -jar nazwa_pliku.jar”. Po prawidłowym wykonaniu wszystkich kroków aplikacja powinna się uruchomić i być dostępna pod adresem „http://localhost:8080”.

5.3. Zbudowanie projektu React

Jeśli wymagania środowiskowe zostały spełnione możemy przeprowadzić proces zbudowania aplikacji w React. Aby tego dokonać należy przejść do głównego katalogu aplikacji i uruchomić komendę „npm start”. Jeśli wszystko wykonało się prawidłowo aplikacja powinna być dostępna pod adresem „http://localhost:3000”.

5.4. Pierwsze uruchomienie aplikacji

Aby rozpocząć korzystanie z aplikacji należy przejść pod adres „http://localhost:3000/login”. Przekieruje to użytkownika na stronie logowania do portalu (Rysunek 17).



Rysunek 14 - Widok strony logowania do portalu

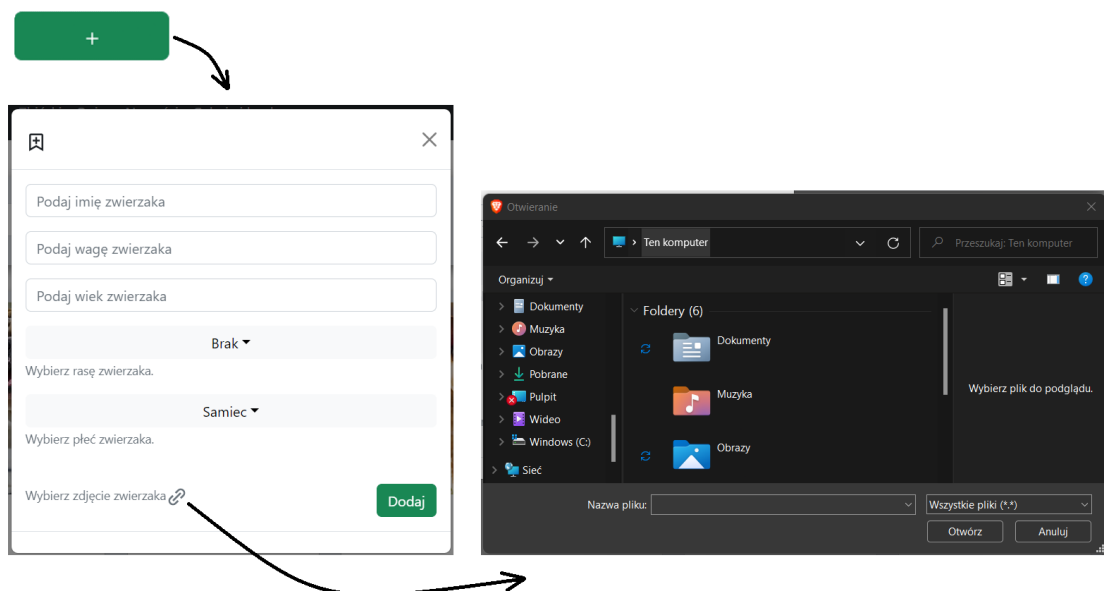
6. Testy akceptacyjne

Celem tego rozdziału będzie sprawdzenie czy aplikacja spełnia stawiane przed nią wymagania. Testy akceptacyjne opierają się na scenariuszach testowych, które obejmują wszystkie przypadki użycia, które mogą wystąpić w systemie. Scenariusze te są zwykle opracowane w oparciu o wymagania użytkowników lub wymagania biznesowe.

6.1. Zarządzanie zwierzętami

Głównym celem aplikacji jest przekazanie użytkownikowi możliwości kontrolowania życia i zdrowia posiadanych zwierząt. Każda osoba korzystająca z portalu może dodać swoje zwierzęta do pewnego rodzaju kolekcji.

Aby dodać zwierzę do kolekcji należy nacisnąć zielony przycisk z plusem co spowoduje wyświetlenie formularza.



Rysunek 15 - Widok formularza dodawania zwierząt

W celach testowych wypełniono formularz następującymi danymi:

- imię – „Max”,
- waga - „12”,
- wiek – „7”,
- rasa – „Beagle”,
- płeć – „Male”,
- zdjęcie – zdjęcie załadowane z urządzenia.

Efektem dodania zwierzęcia będzie pojawianie się go na liści w postaci karty (Rysunek 19). Czerwony przycisk z napisem „Usuń” spowoduje usunięcie zwierzęcia wraz z wszystkim danymi, które go dotyczą.



Rysunek 16 - Karta reprezentująca dodane zwierzę

Jeśli użytkownik poda dane niezgodne z przyjętymi wymaganiami otrzyma on informacje od walidacji formularza (Rysunek 20).

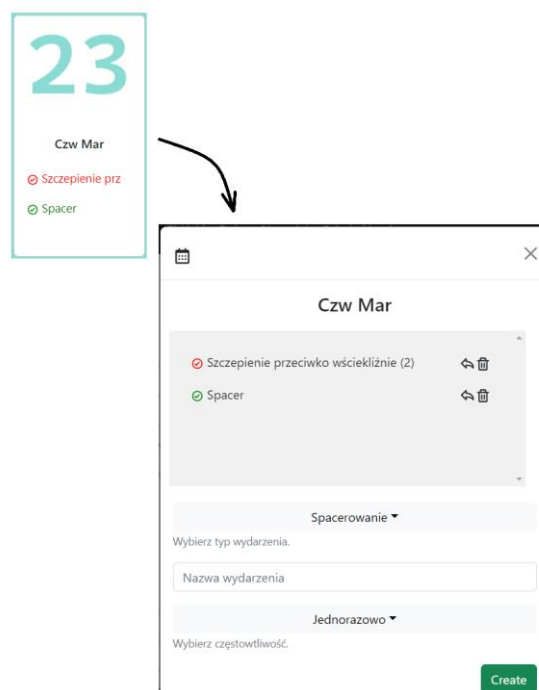
Rysunek 17 - Przykład działania walidacji formularza

6.2. Zarządzanie wydarzeniami

Każde zwierzę, które jest przypisane do użytkownika może posiadać zbiór wydarzeń przypisanych do danego zwierzęcia. Wydarzenia te reprezentują fakty z życia pupila zaplanowane bądź zrealizowane na przestrzeni czasu. Użytkownik ma możliwość dodania takiego wydarzenia oraz zaklasyfikowanie go do konkretnej kategorii. Kategorie dostępne do wyboru to:

- „szczepienie”
- „karmienie”
- „czesanie”
- „spacer”

Aby dodać wydarzenie do terminu, który nas interesuje należy wybrać odpowiedni dzień następnie wybrać typ wydarzenia, nadać mu nazwę oraz określić częstotliwość w jakim ma występować. Częstotliwość jest uzależniona od typu wydarzenia w związku z czym np. szczepienie będzie mogło być wykonane jednorazowo ponieważ wymaga indywidualnego zaplanowania w zależności od rodzaju szczepionki.



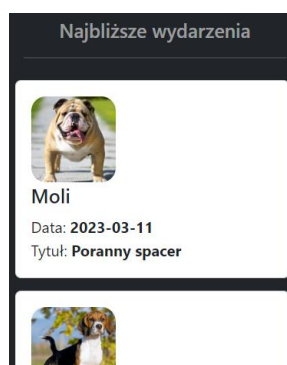
Rysunek 18 - Formularz dodawania nowego wydarzenia

Stworzone wydarzenie może zostać usunięte oraz udostępnione poprzez odpowiednie przyciski znajdujące się obok nazwy wydarzenia. Jeżeli zwierzę ma zlecone przez lekarza szczepienia użytkownik może zaplanować je w kalendarzu. Od teraz aplikacja sama będzie przypominała o tym że zbliża się szczepienie. Jeśli termin takiego szczepienia się zbliża aplikacja poinformuje o tym użytkownika poprzez przygotowany do tego celu alert (Rysunek 22). Jeśli osoba wybierze dogodny dla niej termin szczepienie zostanie dodane do kalendarza i pojawi się na liści wydarzeń oczekujących.



Rysunek 19 - Alert odnośnie zbliżającego się szczepienia

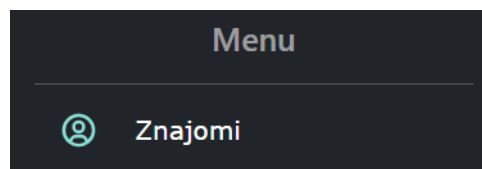
Biorąc pod uwagę ilość wydarzeń jakie mogą być skojarzone ze zwierzętami aplikacja przypomina o tych, których termin realizacji jest najbliższy (Rysunek 23).



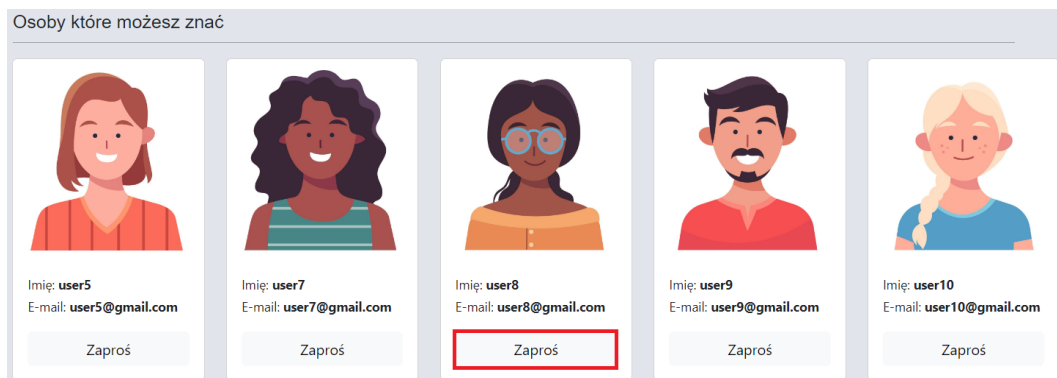
Rysunek 20 - Zbliżające się wydarzenia

6.3. Zarządzanie znajomymi

Portal skupia się w dużej mierze na kontaktach z innymi osobami. Jednym ze sposobów nawiązywania znajomości jest dodawanie innych użytkowników do grona znajomych. Można tego dokonać w zakładce „Znajomi” (Rysunek 24). Aby dodać osobę do znajomych w pierwszej kolejności należy znaleźć to sobą na liście proponowanych lub wyszukać ją. Następnie można wysłać do tej osoby zaproszenie do grona znajomych.

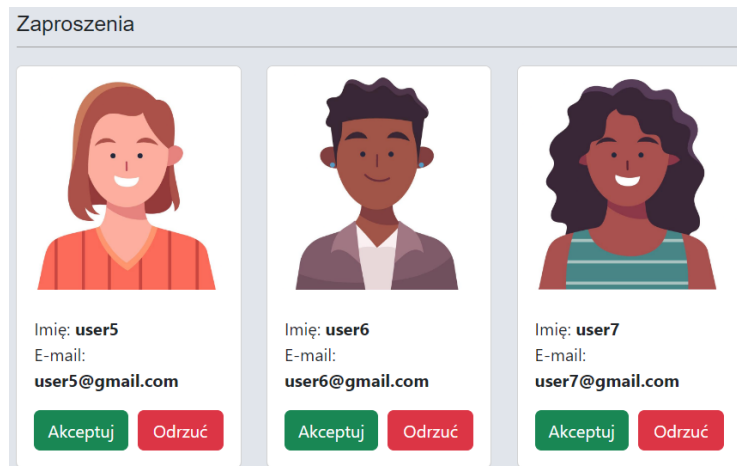


Rysunek 21 - Zakładka "Znajomi"



Rysunek 22 - Widok proponowanych osób

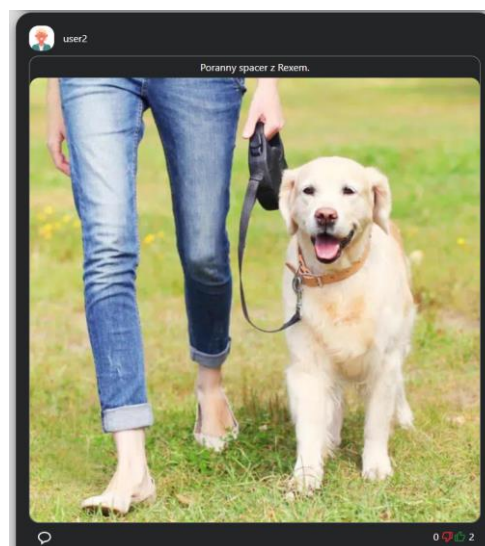
Po wysłaniu zaproszenia użytkownik może zaakceptować je lub odrzucić (Rysunek 26).



Rysunek 23 - Karty reprezentujące zaproszenia

6.4. Interakcje ze społecznością

Interakcje ze społecznością realizowane są poprzez udostępniane posty (Rysunek 27). Celem postów jest przekazywanie informacji o wydarzeniach z życia naszych pupili, takich jak wizyty u weterynarza, ważne daty, ciekawostki czy też zdjęcia. Dzięki temu nasi znajomi mogą być na bieżąco z życiem naszych pupili oraz w razie potrzeby wesprzeć nas w trudniejszych sytuacjach, jak np. choroba czy zagubienie zwierzaka. Interakcje ze społecznością stanowią ważny element aplikacji, umożliwiając użytkownikom dzielenie się swoimi doświadczeniami, poradami oraz nawiązywanie nowych znajomości w świecie miłośników zwierząt.



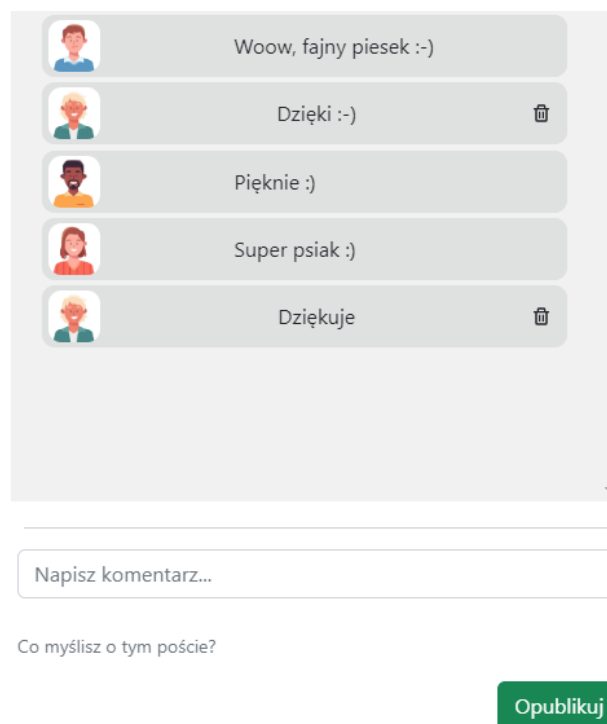
Rysunek 24 - Przykładowy post społeczności

Posty są przygotowywane w przeznaczonym do tego celu miejscu czyli w górnej części tablicy z postami (Rysunek 28). Podajemy tutaj tekst który będzie opisywał to co chcemy udostępnić oraz możemy załączyć zdjęcie.



Rysunek 25 - Pole do tworzenia postów

Każdy post posiada opcje oceniania oraz sekcje komentarzy, w której znajomi mogą sobie nawzajem komentować posty.

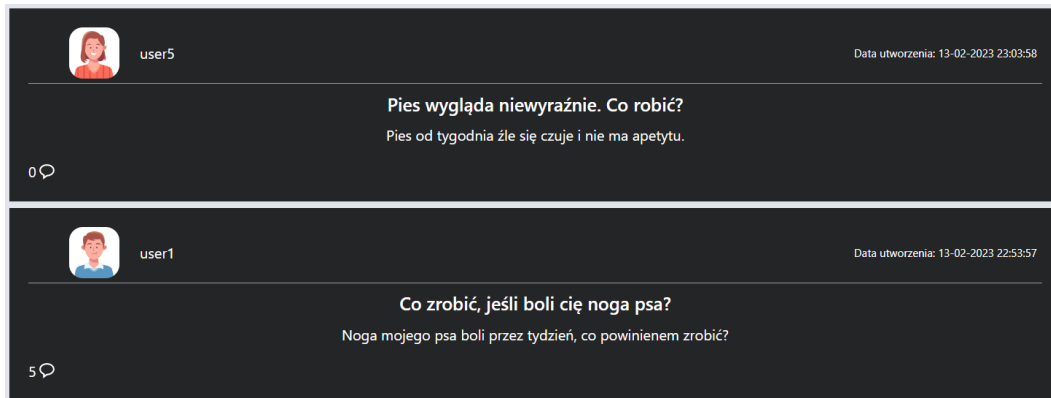


Rysunek 26 - Formularz do dodawania komentarzy

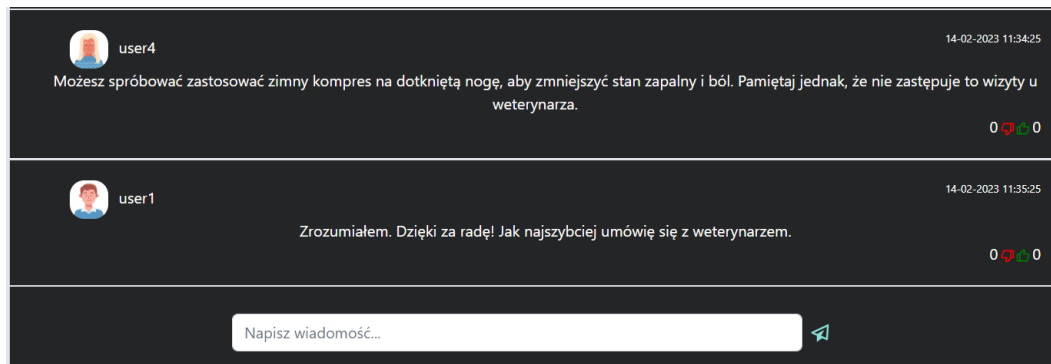
Komentarze są dodawane poprzez przygotowany do tego celu panel (Rysunek 29). Daje on możliwość tworzenia komentarzy ale również usuwania wcześniej dodanych.

6.5. Forum społeczności

Forum społeczności to miejsce w którym użytkownicy portalu komunikują się pomiędzy sobą na tematy, które ich interesują.



Rysunek 27 - Widok dostępnych wątków na forum



Rysunek 28 - Widok konwersacji

Wątki na dany temat są prowadzone na wzór dialogu pomiędzy użytkownikami, co pozwala łatwo przeanalizować konwersacje.

6.6. Zbiórki charytatywne

Zbiórki charytatywne mogą być zakładane przez użytkowników na wyznaczone przez nich cele. Osoba chcąc wpłacić pieniądze na zbiórkę może wyszukać ją spośród wszystkich dostępnych jak również może zasilić zbiórkę swoich znajomych.

Zbiórka na psy
Zbieram pieniądze na małe psy

50%

Wesprzyj

Zbiórka na psy
Cel: 1000\$

| Użytkownik | Kwota |
|------------|--------|
| user4 | 400 \$ |
| user7 | 100 \$ |

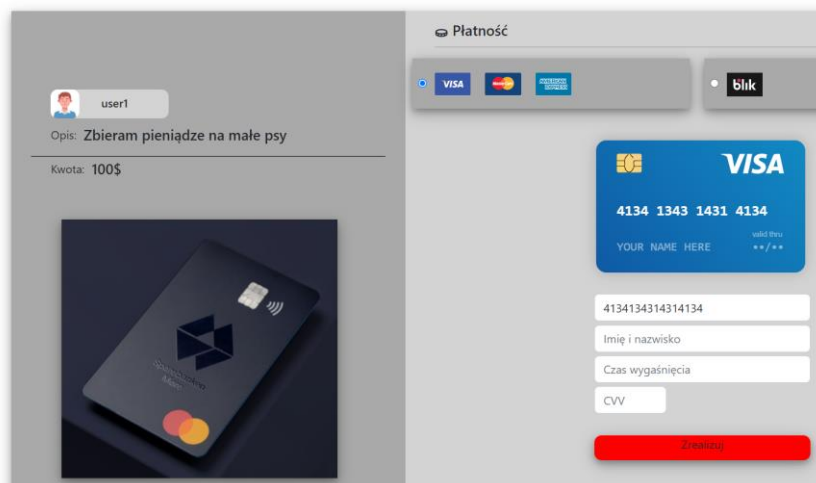
50%

Podaj kwotę

Wesprzyj

Rysunek 29 - Formularz wspierania zbiórki

Aby wesprzeć zbiórkę należy wybrać tą która jest dla nas odpowiednią, co spowoduje wyświetlenie się formularza. W wyznaczonym polu podajemy kwotę oraz naciskamy na przycisk z napisem „Wesprzyj” (Rysunek 32). Aby dokonać płatności użytkownik zostanie przekierowany na odpowiedni widok (Rysunek 33).



Rysunek 30 - Widok realizacji płatności

Po wybraniu odpowiedniej metody płatności, wpisaniu poprawnych danych oraz zaakceptowaniu transakcji użytkownik zostanie przekierowany z powrotem do aplikacji.

6.7. Prasa branżowa

Prasa branżowa jest zbiorem artykułów, których tematyka jest skierowana do posiadaczy zwierząt domowych.



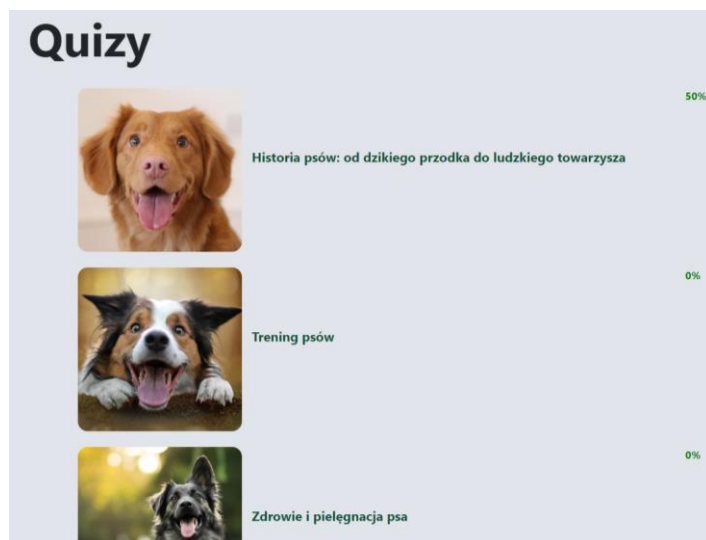
Rysunek 31 - Widok sekcji z artykułami



Rysunek 32 - Widok przykładowego artykułu

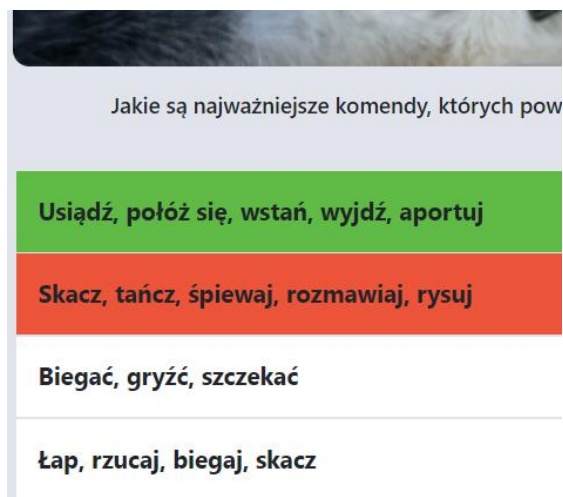
6.8. Tematyczne quizy

W aplikacji można rozwiązać szereg quizów i sprawdzić swoją wiedzę na różne tematy związane ze zwierzętami. Wynik każdego quizu można poprawić co pomoże lepiej zaznajomić się z ciekawymi faktami na temat zwierząt.



Rysunek 33 - Widok dostępnych quizów

Każdy quiz składa się z dziesięciu pytań, a każde pytanie ma cztery odpowiedzi z czego tylko jedna jest prawidłowa.



Rysunek 34 - Widok pytania do quizu

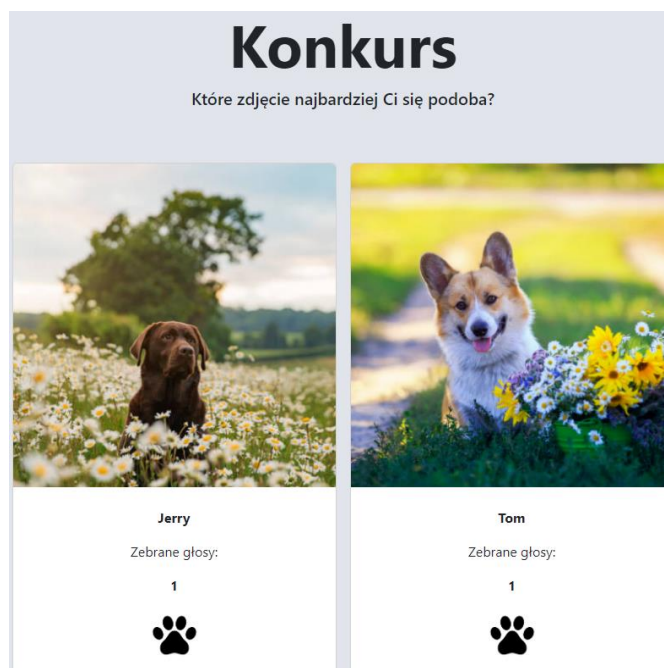
6.9. Tematyczne konkursy

Sekcja konkursów daje możliwość głosowanie przez użytkowników na najładniejsze zdjęcie. Do dyspozycji użytkowników zawsze są konkursy dopasowane do obecnie popularnej kategorii.



Rysunek 35 - Widok konkursów

Po wybraniu konkursu mamy dostęp do zdjęć, na które można oddać swój głos co spowoduje zwiększenie ilości punktów przy zdjęciu oraz zmienienie pozycji zdjęcia w rankingu.



Rysunek 36 - Widok przykładowego konkursu