



Dokumentacja Backend

Programowanie Zespołowe 2024/25

Skład Zespołu:

- Kacper Dąbrowski (Opiekun zespołu)
- Dominik Dylewski (Kierownik zespołu / Backend developer)
- Michał Zieliński (Skryba / Backend developer)
- Urszula Wróblewska (Frontend developer)
- Natalia Rutkowska (Frontend developer)
- Piotr Siwek (Backend developer)

Dokumentacja API: <https://petfinity.onrender.com/api>

Wydział Matematyki i Informatyki UMK

Opis Projektu

Celem powstania tej aplikacji jest uproszczenie procesu adopcji zwierząt, aby był bardziej świadomy i lepiej dopasowany do indywidualnych potrzeb użytkowników. Dzięki zastosowaniu systemu dopasowań użytkownicy mogą łatwo przeglądać profile zwierząt i odnajdywać te, które najlepiej pasują do ich stylu życia. Aplikacja wspiera odpowiedzialne decyzje adopcyjne, zwiększając szanse na trwałe i udane relacje między zwierzętami a opiekunami.

Użyte technologie

Do stworzenia backendu aplikacji na serwerze Node.js wykorzystano:

- NestJS jako główny framework aplikacyjny,
- MongoDB jako bazę danych NoSQL,
- TypeScript dla silnego typowania i strukturyzacji kodu
- różne biblioteki takie jak:
 - Mongoose do modelowania danych,
 - Bcrypt do hasowania haseł,
 - Passport do uwierzytelniania,
 - itp.

Serwer Node.js

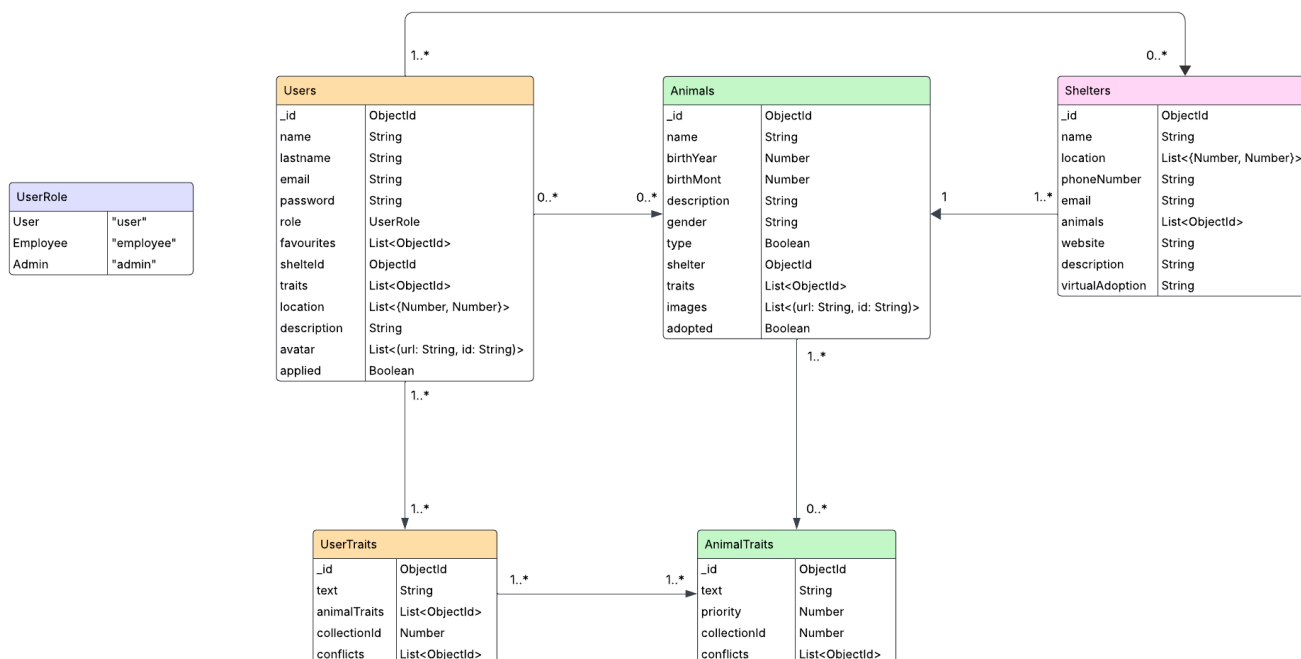
Node.js to środowisko uruchomieniowe JavaScript, które działa po stronie serwera. Umożliwia tworzenie szybkich, skalowalnych aplikacji dzięki nieblokującemu modelowi wejścia-wyjścia opartego na zdarzeniach. Dzięki bogatemu ekosystemowi bibliotek (npm), znacznie przyspiesza i ułatwia rozwój oprogramowania.

Serwer MongoDB

MongoDB to dokumentowa baza danych NoSQL, która przechowuje dane w formacie BSON, będącym binarnym rozszerzeniem JSON-a. Jest elastyczna, nie wymaga sztywnego schematu danych, co ułatwia szybkie iterowanie i zmiany w aplikacjach. MongoDB jest zoptymalizowana do pracy z dużymi zbiorami danych i oferuje wysoką wydajność oraz łatwe skalowanie w poziomie. Zapewnia bogate możliwości zapytań, indeksowania i agregacji danych.

Mongoose to biblioteka ODM (Object Data Modeling) dla Node.js, służąca do pracy z bazą MongoDB. Pozwala na definiowanie schematów, które nadają strukturę dokumentom w bazie, co ułatwia ich walidację i kontrolę spójności.

Diagram encji



Encje w bazie danych

Users – służy do przechowywania informacji o użytkowniku. Dostępne pola:

- `_id` (typ: `ObjectId`) – unikalne id użytkownika,
- `email` (typ: `String`) – email użytkownika,
- `name` (typ: `String`) – imię użytkownika,
- `lastname` (typ: `String`) – nazwisko użytkownika,
- `password` (typ: `String`) – zahasowane hasło użytkownika,
- `role` (typ: `UserRole`) – rola użytkownika [admin, pracownik lub użytkownik],
- `favourites` (typ: `List<ObjectId>`) – tablica przechowująca id zwierząt polubionych przez danego użytkownika,
- `shelterId` (typ: `ObjectId`) – id schroniska, dla którego dany użytkownik pracuje,
- `traits` (typ: `List<ObjectId>`) – tablica przechowująca id tagów opisujących użytkownika,
- `location` (typ: `List<{Number, Number}>`) – tablica przechowująca ostatnio wybraną lokalizację wyszukiwania zwierząt,
- `description` (typ: `String`) – opis użytkownika,
- `avatar` (typ: `List<{url:string, id:string}>`) – tablica przechowująca link do zdjęcia użytkownika,
- `applied` (typ: `Bool`) – określa czy użytkownik złożył podanie o stworzenie schroniska.

Animals – służy do przechowywania informacji o zwierzakach ze schronisk.

Dostępne pola:

- `_id` (typ: `ObjectId`) – unikalne id zwierzaka,
- `name` (typ: `String`) – imię zwierzaka,
- `birthYear` (typ: `Number`) – rok urodzenia zwierzaka,
- `birthMonth` (typ: `Number`) – miesiąc urodzenia zwierzaka,
- `description` (typ: `String`) – opis zwierzaka,
- `gender` (typ: `String`) – płeć zwierzaka (pies/suczka/kot/kotka),
- `type` (typ: `Bool`) – określenie jakie to zwierze (false – pies, true – kot),
- `shelter` (typ: `ObjectId`) – id schroniska, w którym znajduje się zwierzak,
- `traits` (typ: `List<ObjectId>`) - tablica przechowująca id tagów opisujących zwierzaka,
- `images` (typ: `List<{url:string, id:string}>`) – tablica przechowująca linki do zdjęć zwierzaka,
- `adopted` (typ: `Bool`) – oznaczenie czy zwierzak został zaadoptowany.

Shelters – służy do przechowywania informacji o schroniskach. Dostępne pola:

- `_id` (typ: `ObjectId`) – unikalne id schroniska,
- `name` (typ: `String`) – nazwa schroniska,
- `location` (typ: `List<{Number, Number}>`) – tablica przechowująca lokalizację schroniska na mapie,
- `phoneNumber` (typ: `String`) – numer telefonu do danego schroniska,
- `email` (typ: `String`) – email schroniska,
- `animals` (typ: `List<ObjectId>`) – tablica przechowująca id zwierzaków przebywających w schronisku,
- `website` (typ: `String`) – odnośnik do strony schroniska,
- `description` (typ: `String`) – opis schroniska,
- `virtualAdoption` (typ: `String`) – informacje o tym jak odbywa się adopcja wirtualna w danym schronisku.

AnimalTraits – służy do przechowywania tagów opisujących zwierzęta.

Dostępne pola:

- `_id` (typ: `ObjectId`) – unikalne id tagu,
- `text` (typ: `String`) – nazwa tagu,
- `priority` (typ: `Number`) – określenie priorytetu tagu,
- `collectionId` (typ: `Number`) – numer pytania dotyczącego tagu,
- `conflicts` (typ: `List<ObjectIds>`) – tablica przechowująca id tagów, które się wzajemnie wykluczają.

UserTraits – służy do przechowywania tagów opisujących użytkownika.

Dostępne pola:

- `_id` (typ: `ObjectId`) – unikalne id tagu,
- `text` (typ: `String`) – nazwa tagu,
- `animalTraits` (typ: `List<ObjectId>`) – tablica przechowująca id tagów zwierza, które są ze sobą powiązane ,
- `collectionId` (typ: `Number`) – numer pytania dotyczącego tagu,
- `conflicts` (typ: `List<ObjectId>`) – tablica przechowująca id tagów, które się wzajemnie wykluczają.

RefreshTokens – służy do przechowywania tokenów uwierzytelniających.

Dostępne pola:

- `_id` (typ: `ObjectId`) – unikalne id tokenu,
- `userId` (typ: `ObjectId`) – id użytkownika, do którego jest przypisany token,
- `token` (typ: `String`) – unikalny token,
- `expiresAt` (typ: `Date`) – data wygaśnięcia tokenu,
- `createdAt` (typ: `Date`) – data utworzenia tokenu.