Wydział Matematyki
i Nauk Informacyjnych
POLITECHNIKA WARSZAWSKA

2022

Inżynieria Oprogramowania

Dokumentacja aplikacji PetShare

Szymon Pawlonka · Joanna Sowa · Jakub Urbański
Daniel Wujkowski · Łucja Żukowska
IiSI MiNI PW

Spis treści

1. Koncept projektu

System ma umożliwiać obsługę procesu adopcji zwierzątek domowych. Zweryfikowani przez administratora użytkownicy z rolą schroniska wystawiają ogłoszenia adopcji, zapewniając przy tym wszystkie wymagane dane. W ten sposób powstaje baza ogłoszeń adopcyjnych, którą następnie użytkownicy o roli adoptującego przeglądają. Mogą oni w każdej chwili utworzyć ofertę adopcyjną w odpowiedzi na wybrane ogłoszenie, a następnie śledzić jej status. Schroniska przeglądają i zarządzają otrzymanymi ofertami adopcji swoich zwierzątek, decydując o ich akceptacji lub odrzuceniu. Przed podjęciem ostatecznej decyzji schronisko ma także możliwość weryfikacji potencjalnego adoptującego. O statusie tej weryfikacji adoptujący jest informowany również poprzez system. System jest ponadto przygotowany na obsługę zgłoszeń użytkowników. Administrator ma możliwość przeglądania złożonych zgłoszeń, a następnie podejmowania decyzji o blokadzie, której ulec może zarówno adoptujący, jak i schronisko.

Moduły systemu:

- **moduł ogłoszeń** – udostępnia współdzieloną platformę do przeglądania wystawionych ogłoszeń, umożliwia ich filtrowanie oraz obserwowanie
- **moduł adopcji** – umożliwia złożenie wniosku adopcyjnego
- **moduł schroniska** – umożliwia zamieszczanie ogłoszeń adopcyjnych, ich przeglądanie oraz śledzenie ich statusu. Następnie pozwala akceptować/odrzucać złożone do wystawionych ofert wnioski adopcyjne
- **moduł administratora** – umożliwia administrację zalogowanych użytkowników oraz ich zatwierdzanie po wcześniejszej weryfikacji, pozwala również na zarządzanie zamieszczonymi ogłoszeniami

2. User Stories

2.1 Front sides

Jako...	chcę...	żeby...	z priorytetem...
Schronisko	Wstawiać ogłoszenie o dostępnym zwierzęciu do adopcji	Osoby adoptujące mogły zobaczyć ogłoszenie i zgłosić chęć adopcji zwierzęcia	MUST HAVE
	Przeglądać listę wszystkich swoich ogłoszeń z opcją ich zamykania	Osoby adoptujące nie mogły zgłaszać się pod zamkniętym ogłoszeniem	SHOULD HAVE
	Przeglądać listę zgłoszeń do adopcji zwierzęcia	Zweryfikować osobę, która ubiega się o adopcję	SHOULD HAVE
	Akceptować lub odrzucać wnioski o adopcję zwierzęcia	Adopcja została przyjęta i osoba adoptująca mogła zgłosić się po odbiór zwierzęcia	SHOULD HAVE
	Przeprowadzić proces weryfikacji osoby adoptującej po okresie próbnym	Schronisko miało pewność, że osoba adoptująca zapewnia zwierzęciu dobre warunki	COULD HAVE
Osoba adoptująca	Przeglądać i filtrować ogłoszenia interesujących mnie zwierzątek	Zobaczyć, jakie zwierzęta są dostępne do adopcji	MUST HAVE
	Zobaczyć szczegóły na temat zwierzątka	Zdecydować, czy chce je zaadoptować	MUST HAVE
	Wyświetlić dane kontaktowe schroniska	Dopytać o rzeczy niewyspecyfikowane w ogłoszeniu	SHOULD HAVE
	Dodawać zwierzątka do obserwowanych	Przemyśleć decyzję o adopcji i łatwo wrócić do ogłoszenia	COULD HAVE
	Zadeklarować chęć adopcji zwierzątka	Adoptować zwierzątko	MUST HAVE

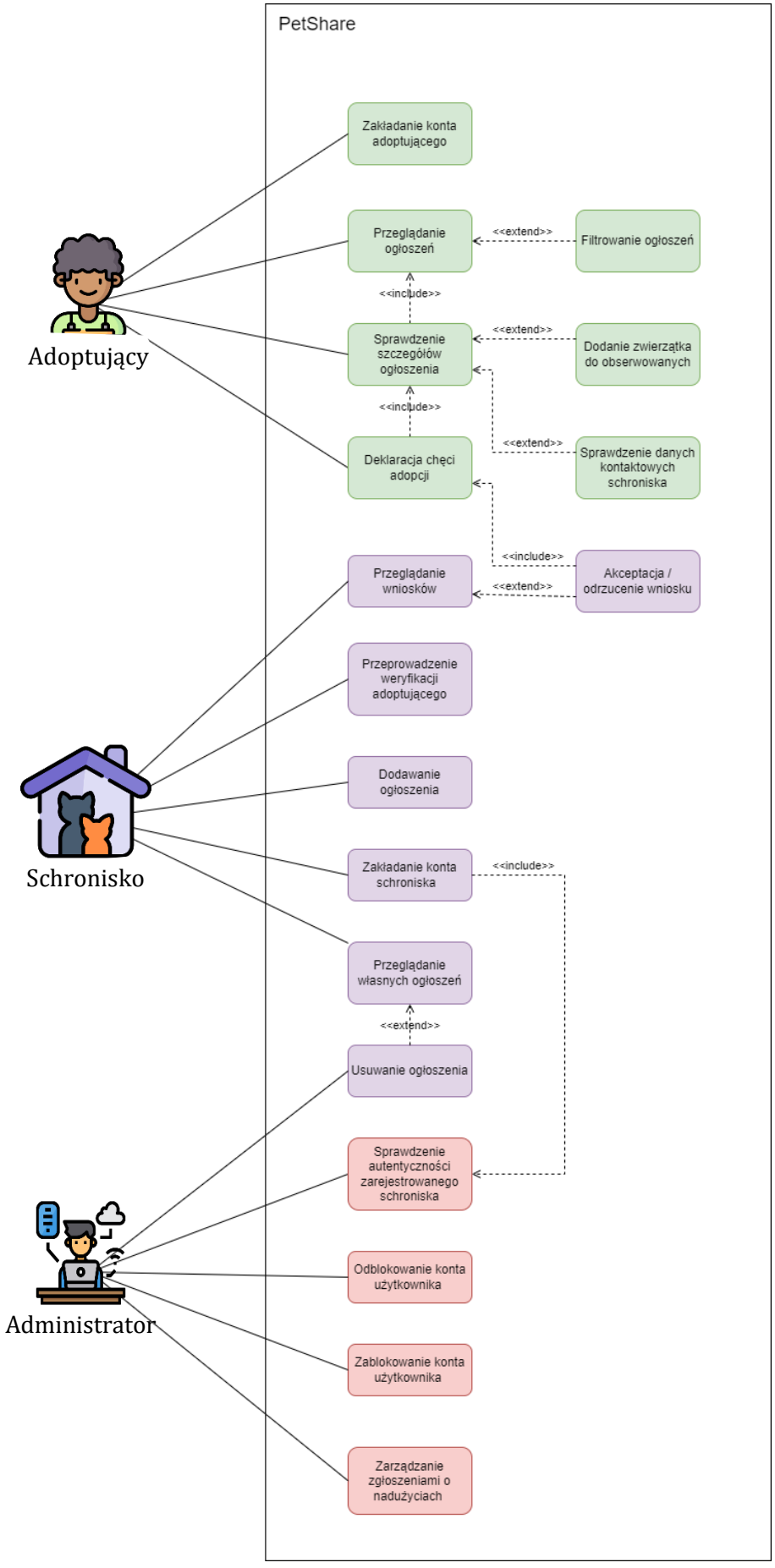
Administrator	Potwierdzić autentyczność zarejestrowanego schroniska	Uniknąć fraudów	SHOULD HAVE
	Usunąć dowolne ogłoszenie	Uniknąć możliwości wystąpienia spamu	MUST HAVE
	Blokować/Odblokować konto dowolnego użytkownika (osoby adoptującej/schroniska)	Mieć kontrolę nad użytkownikami korzystającymi z systemu	MUST HAVE
	Zarządzać zgłoszeniami o nadużyciach	Pomagać rozstrzygać spory pomiędzy użytkownikami	COULD HAVE

2.2 Back sides (FURBS+)

User stories	**Criteria questions**	**Answers**
Jako schronisko chcę wstawiać ogłoszenie o dostępnym zwierzęciu do adopcji, żeby osoby adoptujące mogły zobaczyć ogłoszenie i zgłosić chęć adopcji zwierzęcia.	Czy mogę wstawić więcej niż jedno ogłoszenie na raz?	NIE
	Czy mogę zamieścić imię, wiek, płeć, gatunek, rasę, życiorys, cechy charakterystyczne, opis charakteru i zdjęcie zwierzątka w ogłoszeniu?	TAK
	Czy mogę podać datę wygaśnięcia ogłoszenia?	TAK
Jako osoba adoptująca chcę przeglądać i filtrować ogłoszenia interesujących mnie zwierzątek, żeby zobaczyć, jakie zwierzęta są dostępne do adopcji.	Czy mogę filtrować zwierzątka po ich umiejętności robienia fikołka?	MOŻE
	Czy mogę filtrować zwierzątka po nieobiektywnych kryteriach takich jak poziom uroczości?	NIE
	Czy przy przeglądaniu listy wyświetlane są podstawowe informacje takie jak imię, zdjęcie, gatunek, rasa itp.?	TAK

Jako osoba adoptująca chcę zobaczyć szczegóły na temat zwierzątka, żeby zdecydować, czy chcę je zaadoptować.	Czy mam wgląd w imię, wiek, płeć, gatunek, rasę, życiorys, cechy charakterystyczne, opis charakteru i zdjęcie zwierzątka?	TAK
	Czy mam wgląd w szczegółowe informacje odnośnie zamieszczającego ofertę adopcyjną, takie jak: dane kontaktowe, link do strony internetowej, lokalizacja?	TAK
	Czy mam wgląd w pełną historię leczenia, historię adopcyjną, dane poprzedniego właściciela?	NIE
Jako osoba adoptująca chcę zadeklarować chęć adopcji zwierzątka, żeby adoptować zwierzątko.	Czy mogę w każdej chwili wycofać złożoną deklarację?	TAK
	Czy kolejność złożenia deklaracji ma wpływ na adopcję?	NIE
	Czy mogę złożyć deklarację chęci adopcji kilku zwierzątek na raz?	MOŻE
Jako administrator chcę usunąć dowolne ogłoszenie, żeby uniknąć możliwości wystąpienia spamu.	Czy mogę wybrać więcej niż jedno ogłoszenie do usunięcia?	TAK
	Czy mogę dać ostrzeżenie użytkownikowi po usunięciu jego ogłoszenia?	NIE
	Czy mogę powiadomić użytkownika o usunięciu jego ogłoszenia?	TAK
Jako administrator chcę blokować/odblokować konto dowolnego użytkownika (osobę adoptującą/schronisko), żeby mieć kontrolę nad użytkownikami korzystającymi z systemu.	Czy mogę powiadomić użytkownika o zablokowaniu konta?	TAK
	Czy mogę odblokować konto na życzenie użytkownika?	TAK
	Czy mogę zablokować dowolnego użytkownika w dowolnym momencie?	TAK

3. Use Case Diagram



4. Współdzielone klasy i struktury

Są to wszystkie klasy i struktury, które są wykorzystywane w kilku modułach. Głównie reprezentują modele bazodanowe głównych nośników informacji.

Pet

Stanowi bazodanowy model zwierzątka. Zawiera wszystkie informacje o zwierzątku

ID	GUID
Shelter	Shelter
Name	string
Species	string
Breed	string
Birthday	DateTime
Description	string
Photo	byte[]

Address

Struktura zawierająca informacje o lokalizacji

Street	string
City	string
Province	string
PostalCode	string
Country	string

Announcement

Klasa Announcement stanowi bazodanowy model ogłoszenia. Zawiera wszystkie istotne informacje o ogłoszeniu.

ID	GUID	niezbędne do komunikacji z bazą danych, rozróżnienia między sobą dwa ogłoszenia
Author	Shelter	wskazanie na autora ogłoszenia
Pet	Pet	wskazanie na zwierzątko, którego dotyczy ogłoszenie
Title	string	
Description	string	
CreationDate	DateTime	
ClosingDate	DateTime	
Status	enum	status ogłoszenia- otwarte, zamknięte, w trakcie weryfikacji, do usunięcia
LastUpdateDate	DateTime	data ostatniej edycji ogłoszenia lub zmieny statusu

Filter

Zbiera wybrane przez użytkownika filtry i umożliwia przefiltrowanie ogłoszeń na etapie pobierania informacji z bazy danych.

Species	string	gatunek zwierzątka
Breed	string	rasa zwierzątka
Location	Address	lokalizacja schroniska
Age	TimeSpan	wiek zwierzątka
ShelterName	String	nazwa schroniska

AnnouncementProvider

Umożliwia pobranie ogłoszeń na podstawie zadanych kryteriów (filtru lub ID schroniska).

GetAnnouncements(GUID)	Announcement[]	na podstawie zadanego ID schroniska zwraca listę jego ogłoszeń
GetAnnouncements(Filter)	Announcement[]	na podstawie filtru zwraca listę ogłoszeń spełniających filtr

Application

Klasa Application stanowi bazodanowy model wniosku o adopcję.

ID	GUID	
User	User	
Announcement	Announcement	
DateOfApplication	DateTime	
IsWithdrawed	Bool	określa, czy wniosek został wycofany przez użytkownika
LastUpdateDate	DateTime	data ostatniej edycji wniosku
Withdraw()		wycofuje aplikację

User

Abstrakcyjna klasa User stanowi bazodanowy model konta użytkownika.

ID	GUID	
UserName	string	
PhoneNumber	string	
Email	string	
Address	Address	
AnnouncementProvider	AnnouncementProvider	
CreateReport(GUID, string)	bool	stworzenie zgłoszenia o nadużycie, przyjmuje identyfikator zgłaszanego obiektu (może to być adoptujący, schronisko lub ogłoszenie)

5. Moduł Adopcji

Moduł, który umożliwia stworzenie profilu adoptującego, adopcję wybranego zwierzęcia z ogłoszenia oraz dodania ogłoszenia do obserwowanych.

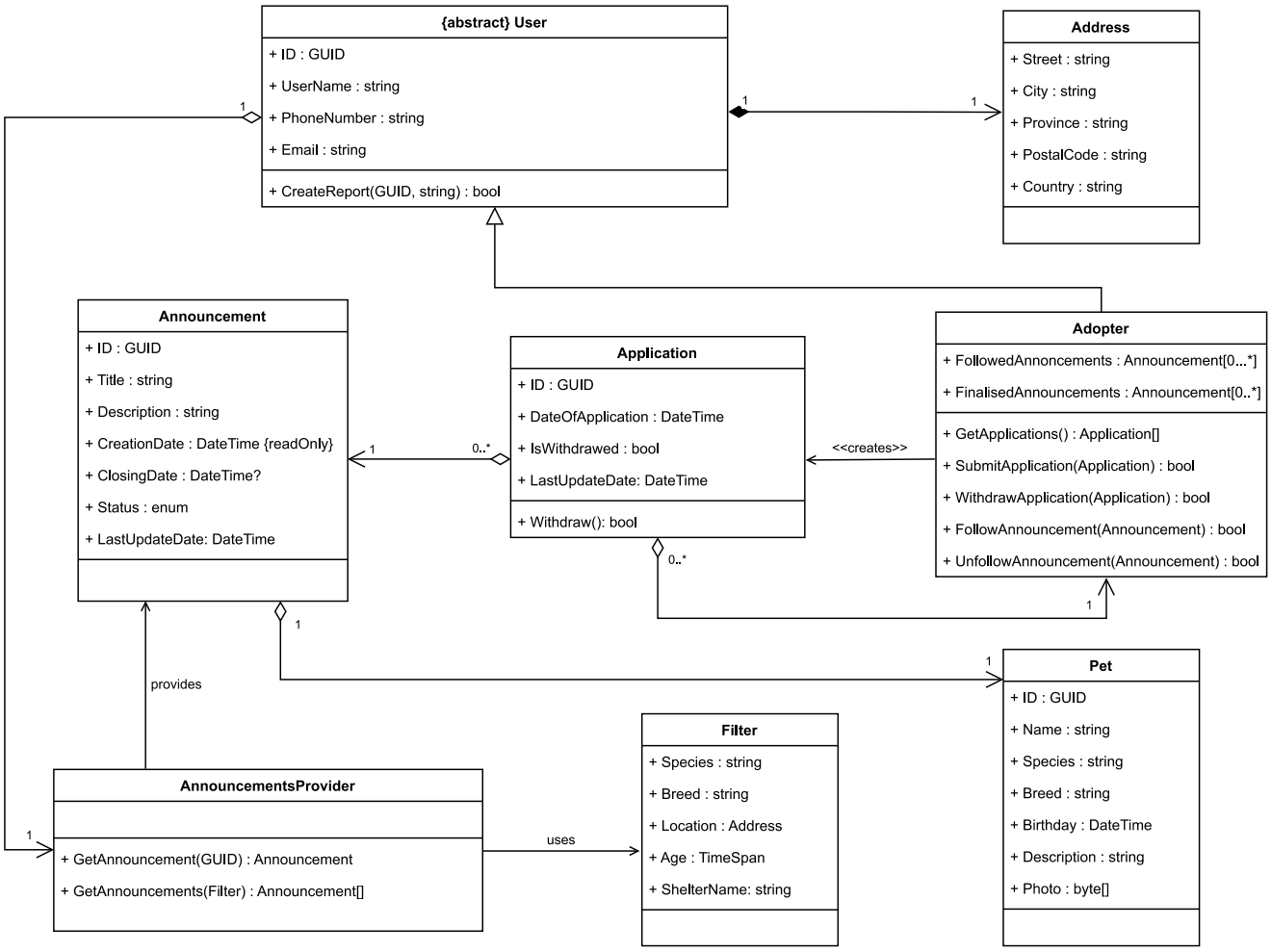
5.1 Klasy

Adopter : User

Dziedziczy po abstrakcyjnej klasie User. Stanowi bazodanowy model adoptującego. Zawiera wszystkie istotne informacje o adoptującym.

FollowedAnnouncements	Announcement[0..*]	lista obserwowanych ogłoszeń przez użytkownika
FinalisedAnnouncements	Announcement[0..*]	lista ogłoszeń, dzięki którym użytkownik zaadoptował zwierzątko
GetApplications()	Application[0..*]	lista złożonych wniosków o adopcję przez użytkownika
SubmitApplication(Application)	bool	umożliwia złożenie wniosku o adopcję wybranego zwierzęcia z ogłoszenia
WithdrawApplication(Application)	bool	umożliwia wycofanie wniosku o adopcję wybranego zwierzęcia z ogłoszenia, jeżeli nie został on jeszcze zaakceptowany przez schronisko
FollowAnnouncement(Announcement)	bool	umożliwia obserwowanie wybranego ogłoszenia
UnfollowAnnouncement(Announcement)	bool	umożliwia usunięcie ogłoszenia z obserwowanych

5.2 Diagram UML



6. Moduł schroniska

Moduł umożliwiający zamieszczanie, przeglądanie oraz śledzenie statusu ogłoszeń adopcyjnych. Zawiera również metody pozwalające akceptować lub odrzucać złożone do wystawionych ofert wnioski adopcyjne.

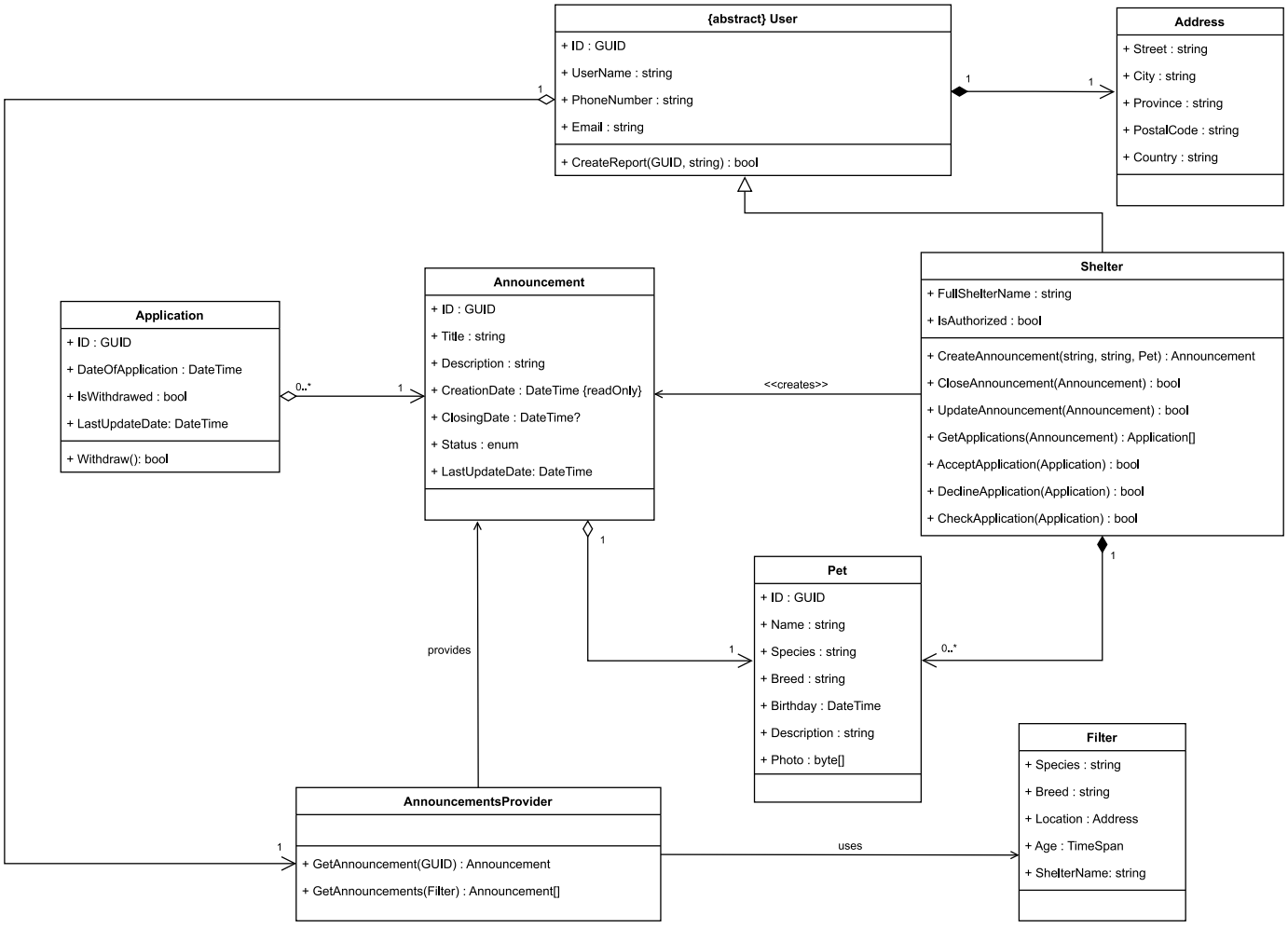
6.1 Klasy

Shelter : User

Zawiera dane na temat schroniska i udostępnia potrzebne metody.

FullShelterName	string	
IsAuthorized	bool	zmienna informująca o zautoryzowaniu schroniska przez administratora
Pets	Pet[]	lista zwierzątek w schronisku
CreateAnnoucement(string, string, Pet)	Announcement	umożliwia stworzenie nowego obiektu klasy Announcement po zapewnieniu tytułu, opisu i zwierzątka, którego dotyczy ogłoszenie.
CloseAnnoucement(Announcement)	bool	umożliwia zamknięcie istniejącego ogłoszenia.
UpdateAnnouncement(Announcement)	bool	umożliwia edycję ogłoszenia.
GetApplications(Announcement)	Application[]	zwraca listę zgłoszeń do danego ogłoszenia.
AcceptApplication(Application)	bool	pozwala zaakceptować wniosek adopcyjny.
DeclineApplication(Application)	bool	pozwala odrzucić wniosek adopcyjny.
CheckApplication(Application)	bool	pozwala zweryfikować zdolność adopcyjną potencjalnego nowego właściciela zwierzątka.

6.2 Diagram UML



7. Moduł administratora

Moduł pozwala na zarządzanie treściami zamieszczanymi przez użytkowników i schroniska.

7.1 Klasy

Admin

Zawiera metody potrzebne do zarządzania aplikacją.

BlockAdopter(GUID)	bool	blokuje adoptującego o podanym ID
UnblockAdopter(GUID)	bool	odblokowuje adoptującego o podanym ID
DeleteAnnouncement(GUID)	bool	usuwa ogłoszenie o podanym ID
AuthenticateShelter(GUID)	bool	potwierdza zweryfikowanie schroniska o podanym ID
BlockShelter(GUID)	bool	blokuje schroniska o podanym ID
UnblockShelter(GUID)	bool	odblokowuje schronisko o podanym ID
GetReports()	Report[0..*]	pobiera zgłoszenia o nadużycia

Abstract Report : Report

Abstrakcyjna klasa zawierająca informacje o zgłoszeniach o nadużycia.

ID	GUID	
Reason	string	powód zgłoszenia
CanBlockAdopter()	bool	metoda określająca, czy jako reakcję możemy zablokować adoptującego
CanBlockShelter()	bool	metoda określająca, czy jako reakcję możemy zablokować schronisko
CanDeleteAnnouncement()	bool	metoda określająca, czy jako reakcję możemy usunąć ogłoszenie
DeleteReport()	bool	usunięcie zgłoszenia

AdopterReport : Report

Implementacja klasy Report odpowiadająca zgłoszeniu adoptującego

AdopterID	GUID	id adoptującego
override CanBlockAdopter ()	bool	zwraca true
override CanBlockShelter()	bool	zwraca false
override CanDeleteAnnouncement()	bool	zwraca false

AnnouncementReport : Report

Implementacja klasy Report odpowiadająca zgłoszeniu ogłoszenia

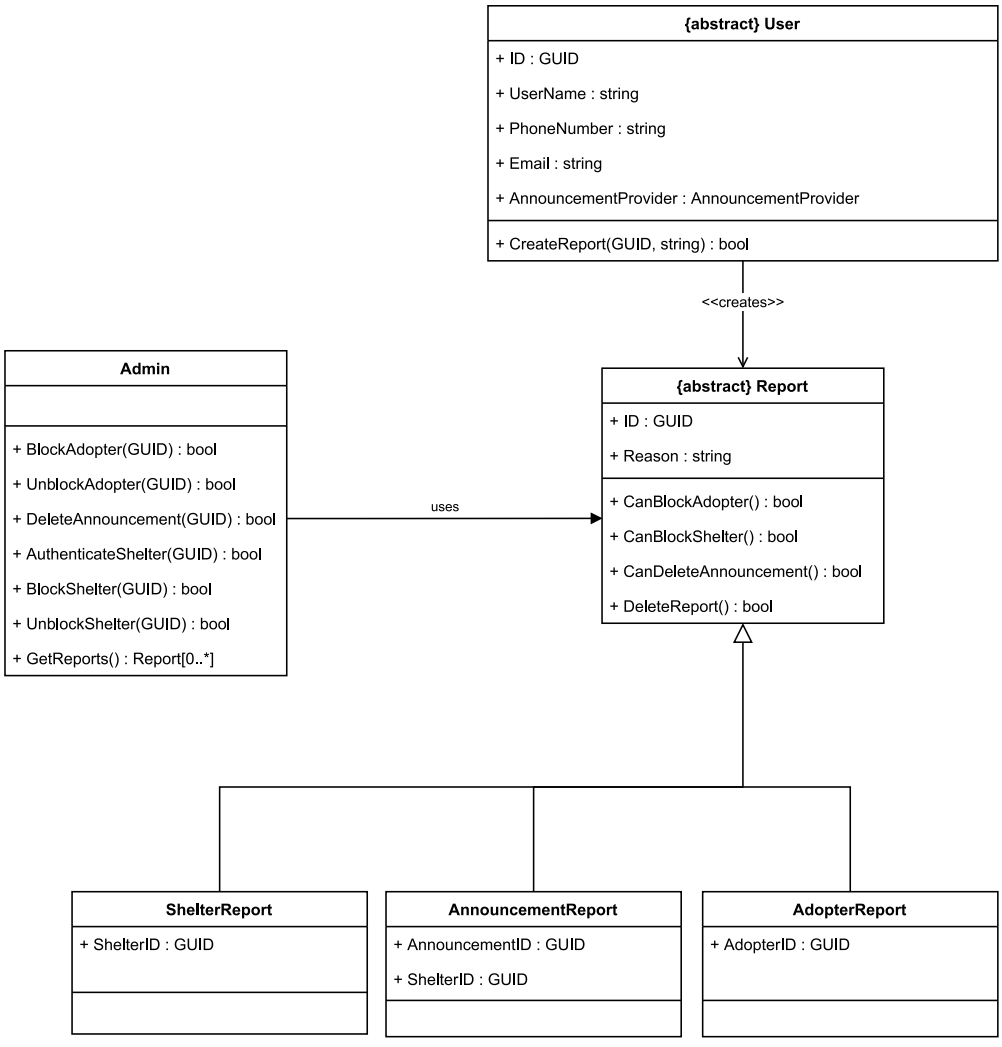
AnnouncementID	GUID	id ogłoszenia
ShelterID	GUID	id schroniska
override CanBlockAdopter ()	bool	zwraca false
override CanBlockShelter()	bool	zwraca true
override CanDeleteAnnouncement()	bool	zwraca true

ShelterReport : Report

Implementacja klasy Report odpowiadająca zgłoszeniu schroniska

ShelterID	GUID	id schroniska
override CanBlockAdopter ()	bool	zwraca false
override CanBlockShelter()	bool	zwraca true
override CanDeleteAnnouncement()	bool	zwraca false

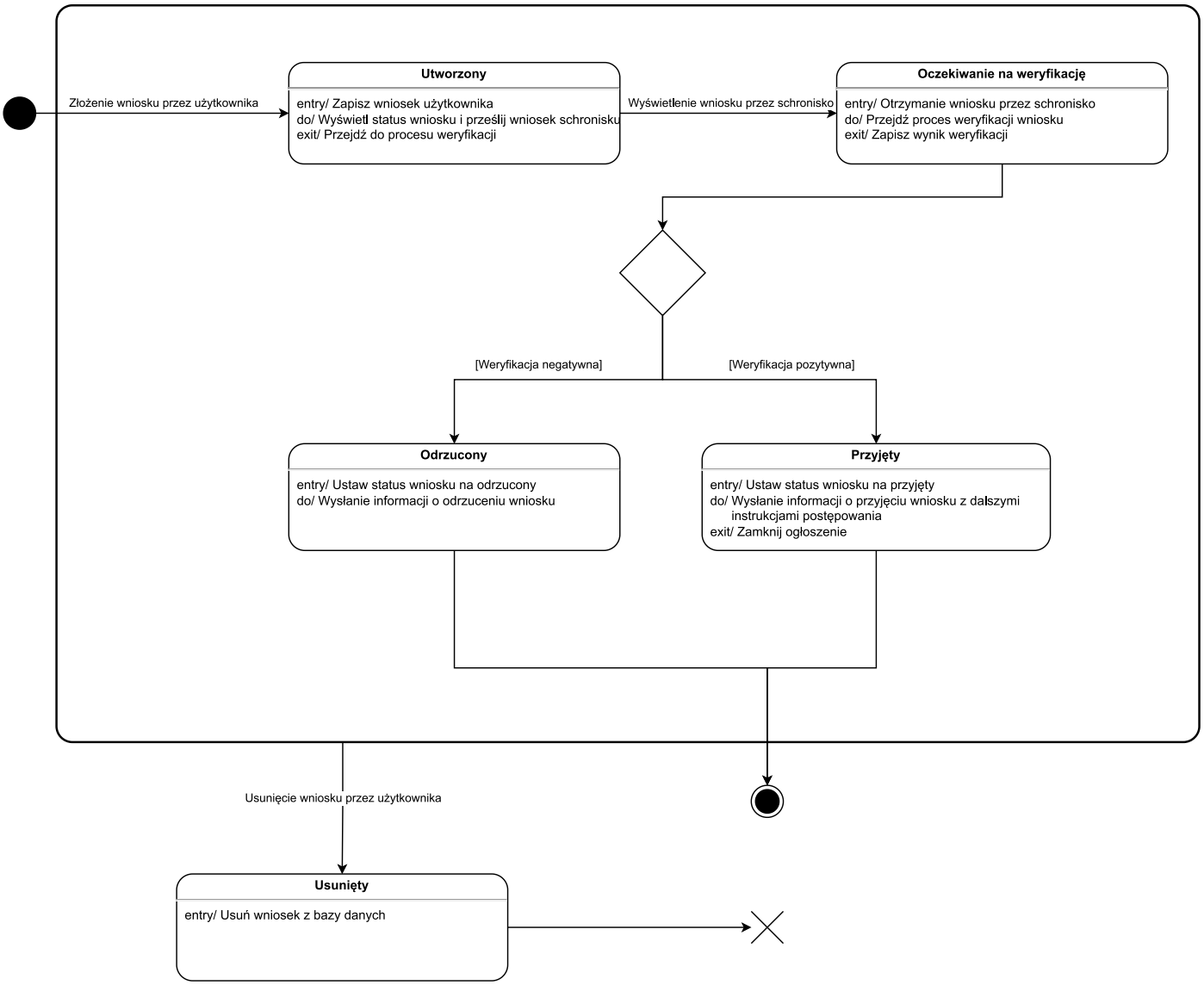
7.2 Diagram UML



8. Diagramy stanów

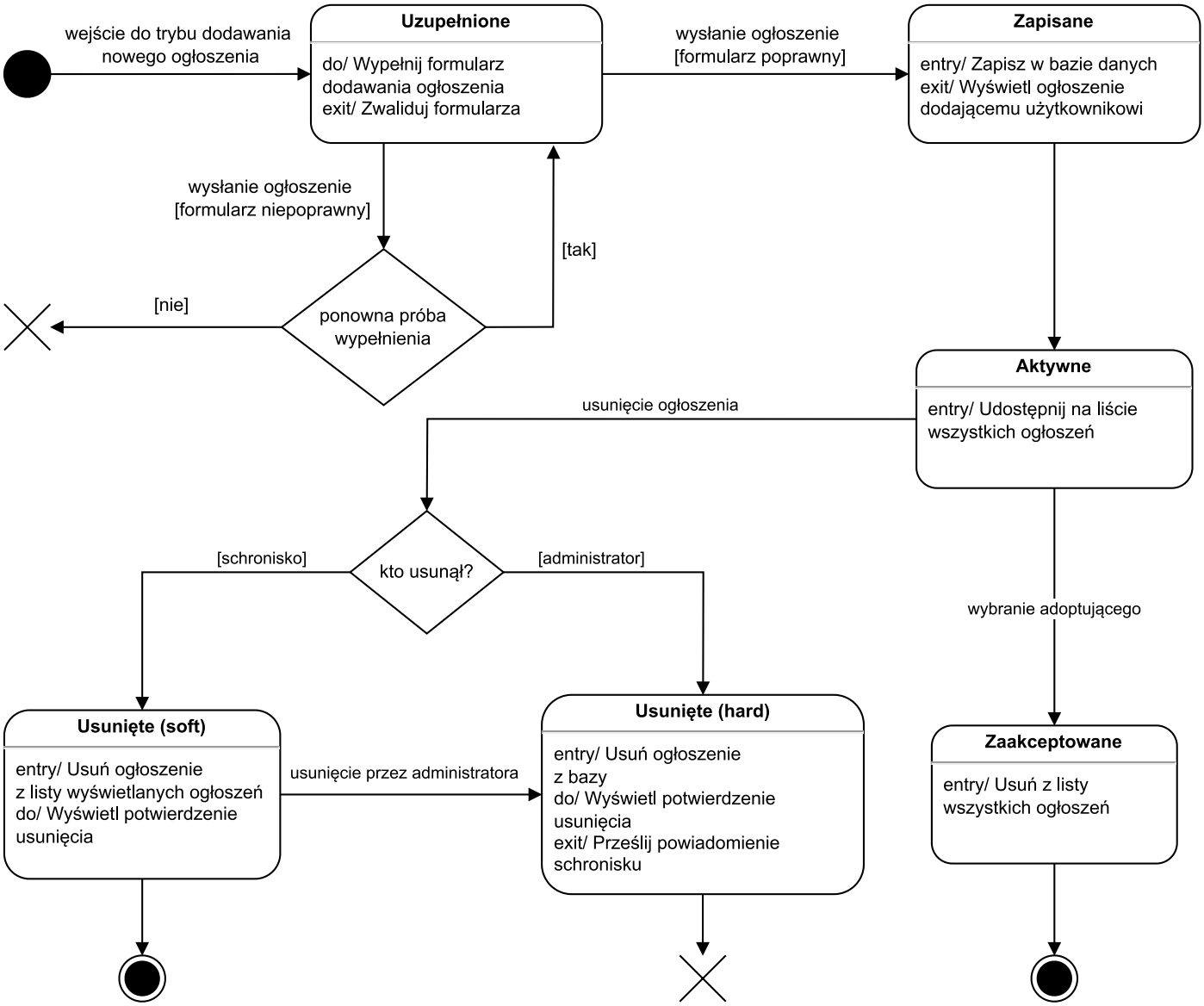
8.1 Aplikacja

Wniosek powstaje po wypełnieniu przez użytkownika formularzu aplikacji. Wówczas wniosek zostaje zapisany, jego status jest widoczny na liście wniosków użytkownika, a następnie przesłany do schroniska. W tym miejscu następuje rozpoczęcie procesu weryfikacyjnego użytkownika, czy jest odpowiednim kandydatem na adopcję zwierzątka. Po wyświetleniu wniosku przez schronisko zmienia on swój stan na „Oczekiwanie na weryfikację". Po prawidłowym przejściu tego procesu jego wynik jest zapisywany. Jeżeli użytkownik przeszedł weryfikację, to wniosek zostaje przyjęty. Użytkownik dostaje informację o przyjęciu wniosku z instruktażem dalszego postępowania, a ogłoszenie, którego wniosek dotyczył zostaje zamknięte. Z kolei, jeżeli użytkownik nie przeszedł procesu weryfikacji, to wniosek zostaje odrzucony, o czym użytkownik zostaje poinformowany. W każdym momencie wniosek może zostać usunięty.



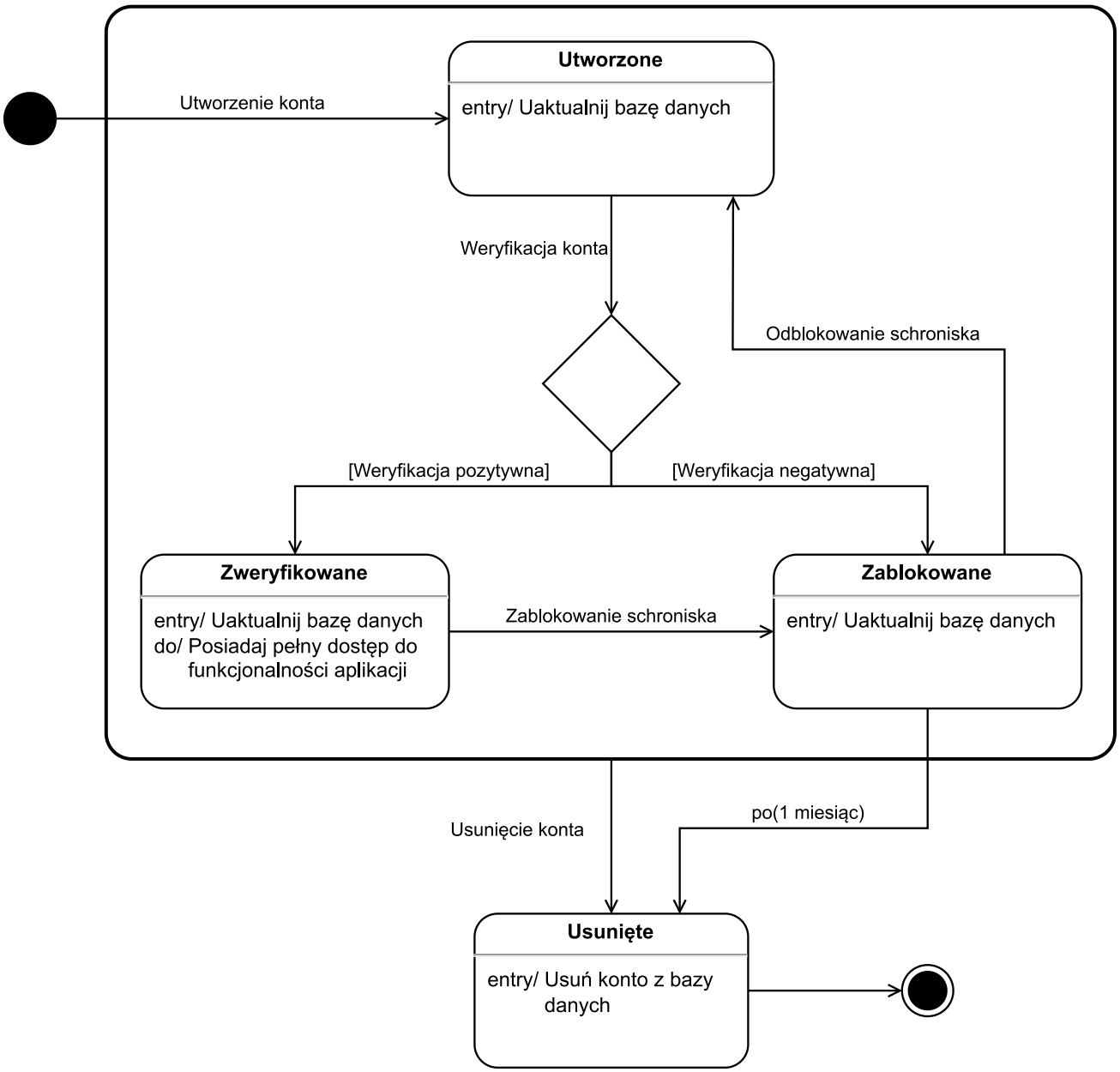
8.2 Ogłoszenie

Schronisko tworzy ogłoszenie przez przejście do dedykowanego formularza, poprawne wypełnienie go i przesłanie. W przypadku niepoprawnej walidacji ogłoszenia, może wypełniać je ponownie do momentu rezygnacji z uzupełniania wniosku lub podania poprawnych wartości w formularzu. Ogłoszenie zostaje zapisane do bazy. Wyjście ogłoszenia ze stanu aktywnego może zostać spowodowane dwiema akcjami - ogłoszenie może zostać usunięte przez schronisko lub administratora, oraz wniosek adopcyjny powiązany z nim może zostać zaakceptowany. W pierwszym przypadku usunięcia przez administratora, ogłoszenie zostanie usunięte z bazy. W pozostałych przypadkach, ogłoszenie zmieni status odpowiednio na usunięte lub zaakceptowane.



8.3 Schronisko

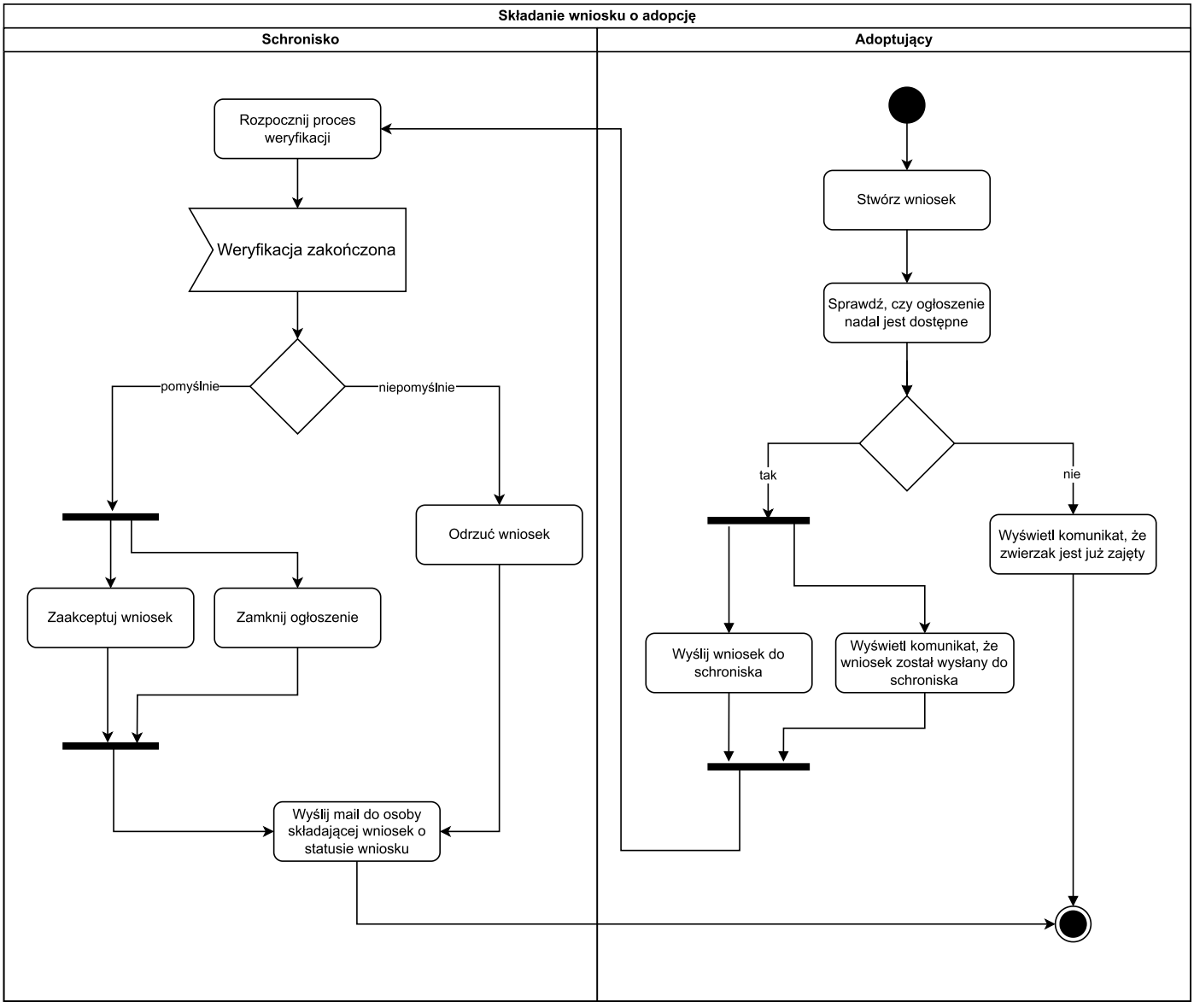
Schronisko zostaje utworzone w procesie rejestracji konta. Wtedy zapisywane jest ono do bazy danych. Administrator musi stworzone konto zweryfikować – jeżeli schronisko przejdzie ten proces pozytywnie zostaje zweryfikowane, co jest zapisywane do bazy danych, a użytkownik ma dostęp do pełnej funkcjonalności aplikacji. W przeciwnym wypadku schronisko jest blokowane. Schronisko zweryfikowane również może zostać zablokowane przez administratora. W stanie zablokowanym schronisko może odwołać się i wrócić do stanu utworzonego. Z dowolnego stanu schronisko może zostać usunięte. Zablokowane konto zostaje automatycznie usunięte po jednym miesiącu.



9. Diagramy aktywności

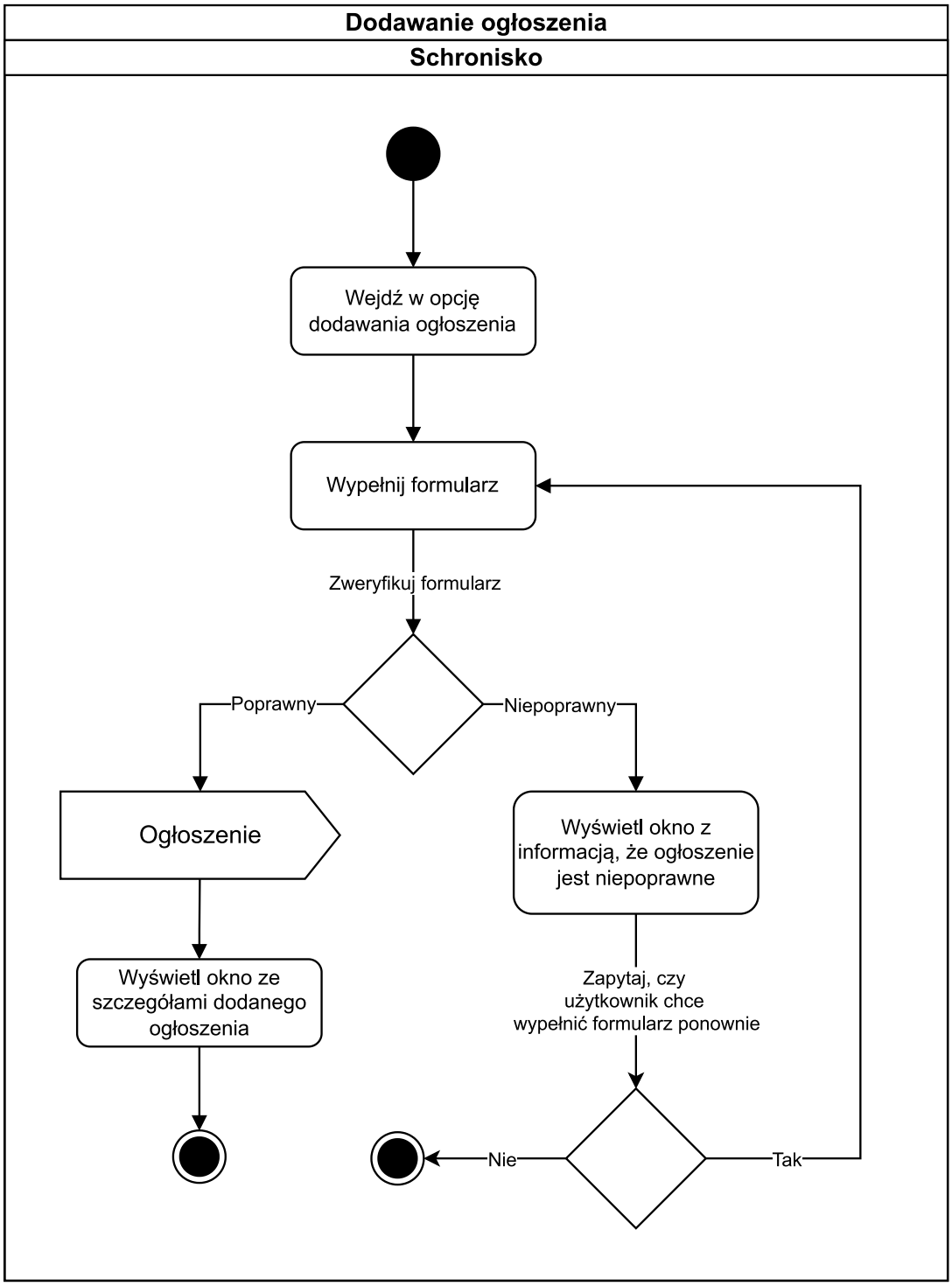
9.1 Złożenie wniosku o adopcję

Adoptujący składa wniosek o adopcję zwierzaka. Następnie jest sprawdzane, czy ogłoszenie jest nadal dostępne. Jeżeli nie, to adoptujący otrzymuje komunikat, że zwierzak jest już zajęty. W przeciwnym przypadku równocześnie wysyłany jest wniosek do schroniska oraz jest wyświetlany komunikat, że wniosek został wysłany do schroniska. Później schronisko rozpoczyna proces weryfikacji wniosku. Po pewnym czasie otrzymywane jest zdarzenie, że weryfikacja zakończyła się. Jeżeli została zakończona pomyślnie, to wniosek jest akceptowany, a ogłoszenie zamykane. W przeciwnym przypadku wniosek jest odrzucany. Na koniec zostaje wysłany mail do osoby składającej wniosek z informacją o aktualnym statusie wniosku.



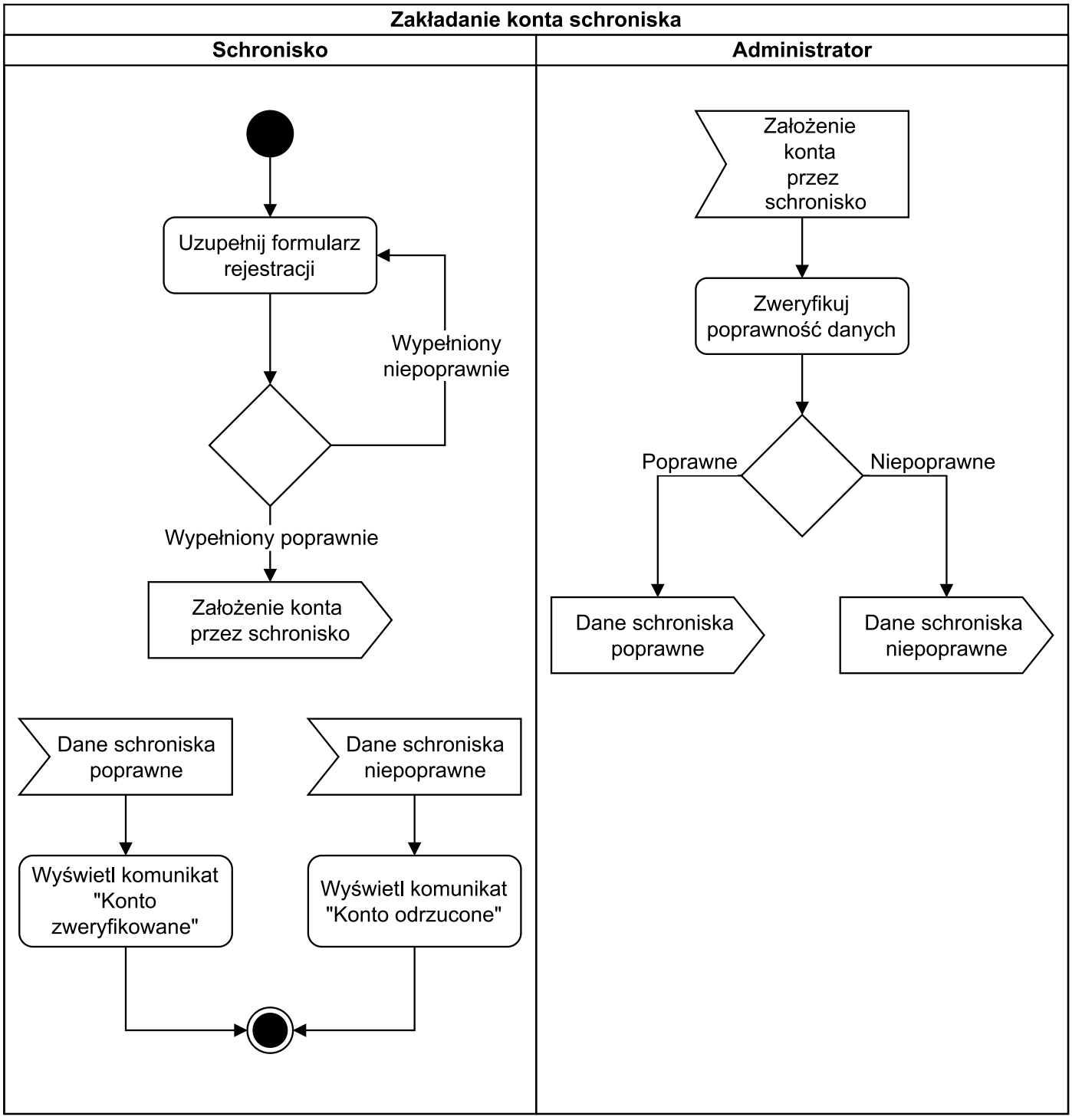
9.2 Tworzenie ogłoszenia

Schronisko otwiera okno dodawania nowego ogłoszenia i wypełnia formularz. Po zaakceptowaniu naniesionych danych następuje weryfikacja formularza. W przypadku pomyślnej weryfikacji wysyłane zostaje zdarzenie w postaci obiektu Ogłoszenia i wyświetlane jest okno ze szczegółami dodanego ogłoszenia. W przeciwnym przypadku (formularz niepoprawny) wyświetlane jest okno z opisem błędu. Jeśli użytkownik zechce poprawić swój formularz ma do tego możliwość. Może też zupełnie zrezygnować z dodawania ogłoszenia.



9.3 Zakładanie konta schroniska

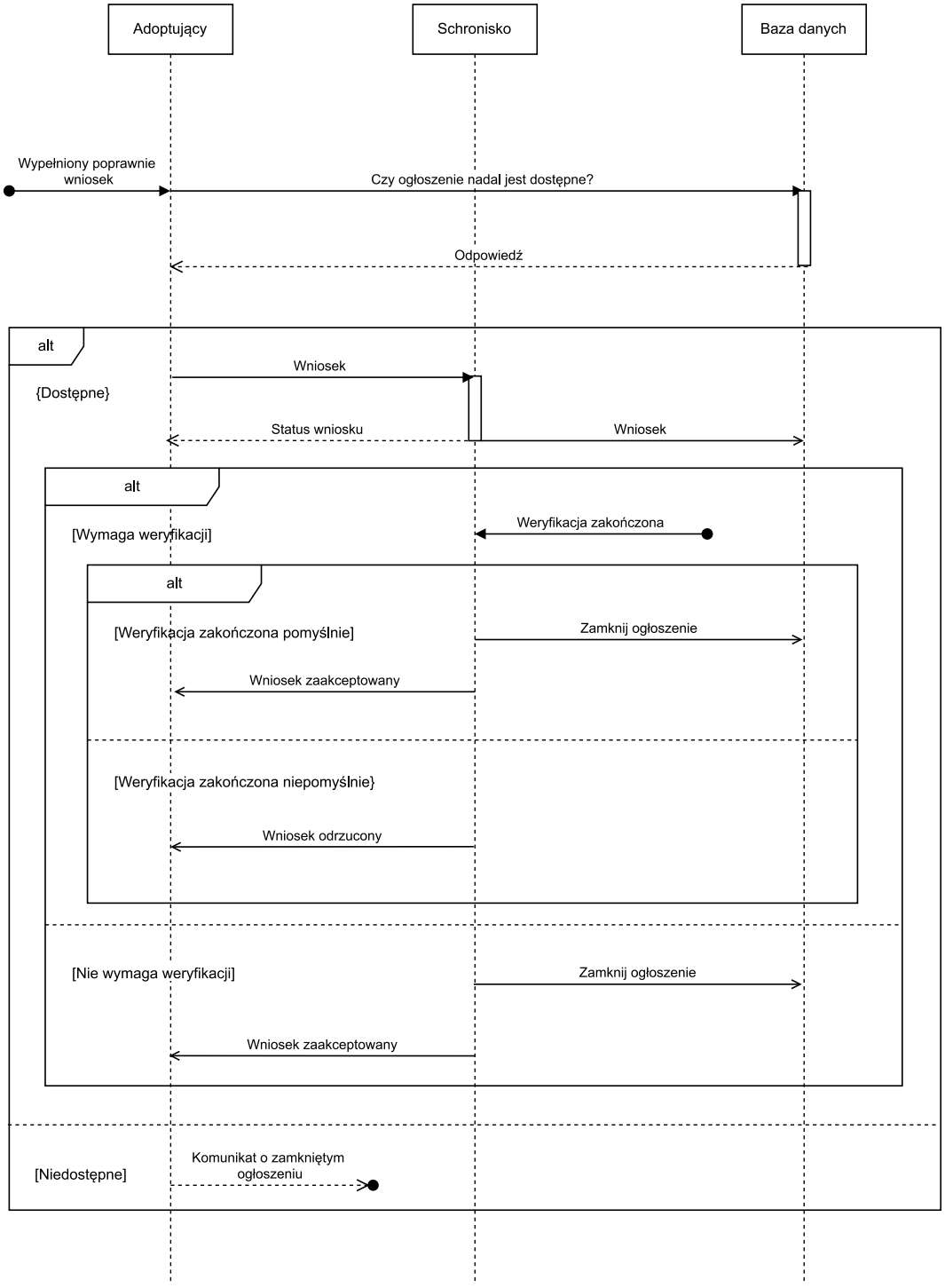
Schronisko wypełnia formularz rejestracyjny. Jeżeli został wypełniony niepoprawnie to należy uzupełnić go ponownie. Jeżeli został wypełniony poprawnie to wysyłane jest zdarzenie o założeniu konta. W odpowiedzi Administrator weryfikuje poprawność danych – potwierdza prawdziwość schroniska – i wysyła odpowiednie zdarzenia – dane poprawne lub nie. W odpowiedzi na nie użytkownikowi wyświetlany jest odpowiedni komunikat.



10. Diagramy sekwencji

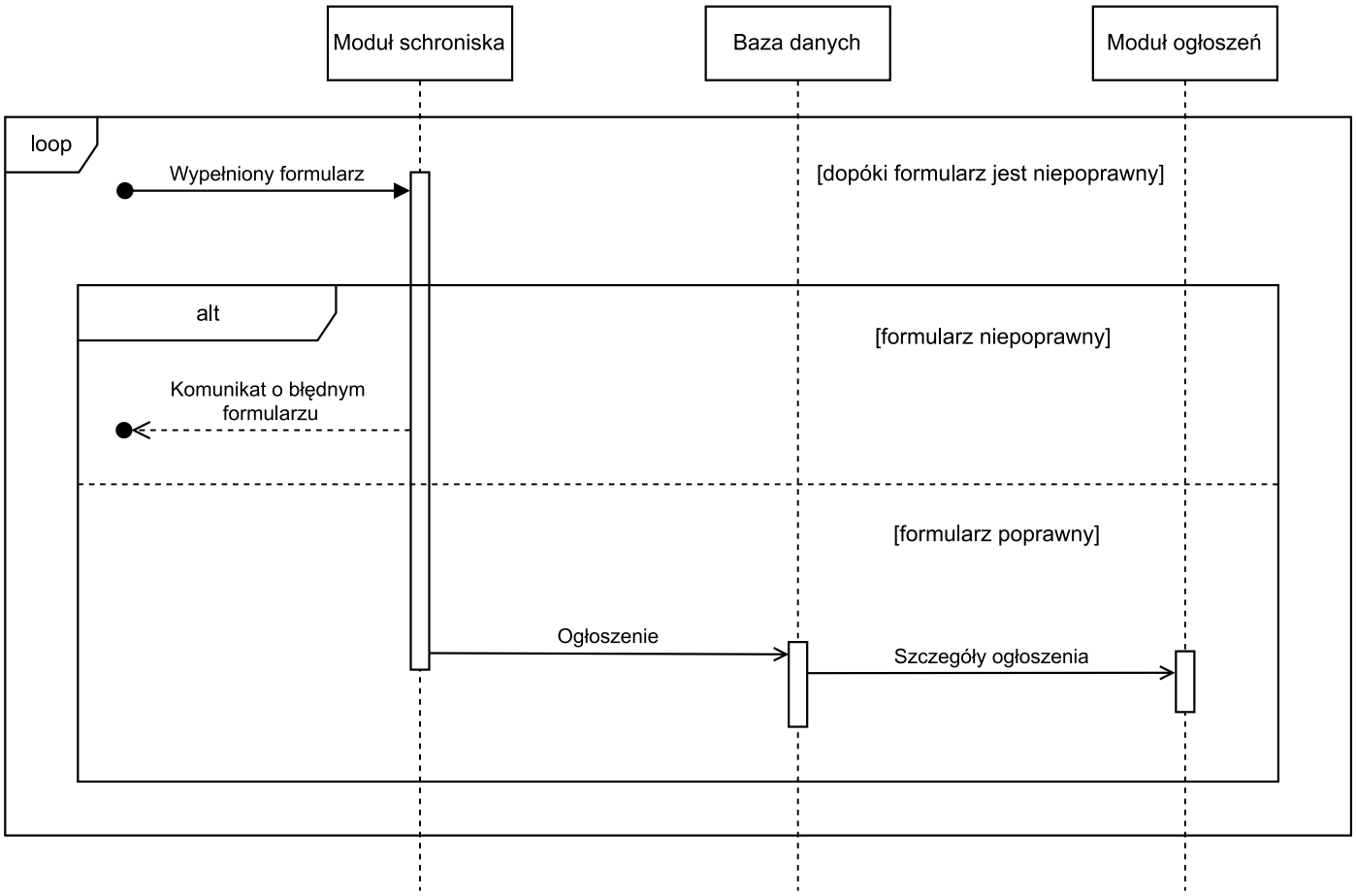
10.1 Składanie wniosku

Po poprawnym wypełnieniu wniosku przez adoptującego idzie zapytanie do bazy danych, czy ogłoszenie nadal jest otwarte. Jeżeli tak, to sprawdzane jest, czy użytkownik powinien przejść proces weryfikacji. Przy odpowiedzi twierdzącej po pozytywnym rozpatrzeniu weryfikacji ogłoszenie jest zamykane w bazie danych, a adoptujący dostaje odpowiedź, że jego wniosek został zaakceptowany. Przy odpowiedzi przeczącej otrzymuje informację, iż jego wniosek został odrzucony. Jeżeli użytkownik nie wymaga weryfikacji ogłoszenie jest od razu zamykane, a odpowiedź o zaakceptowaniu wniosku wysyłana do adoptującego. W przypadku, kiedy ogłoszenie nie jest już dostępne, zostaje wyświetlony stosowny komunikat i nie podejmowane są dalsze kroki.



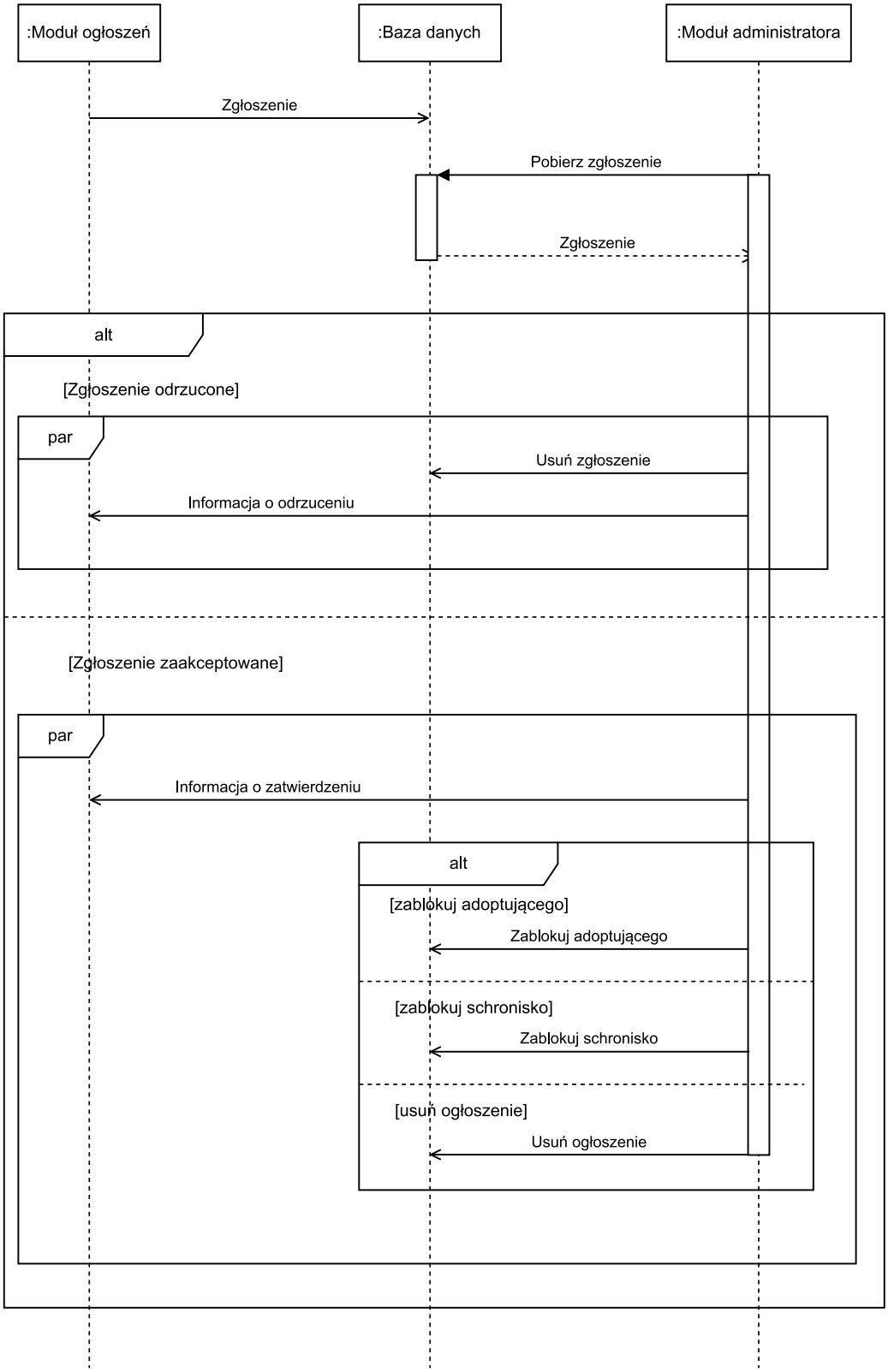
10.2 Dodawanie ogłoszenia

Schronisko wypełnia formularz. Jeśli formularz został wypełniony poprawnie, ogłoszenie zostaje dodane do bazy danych i wyświetlone zostają informacje ze szczegółami dodanego ogłoszenia. Jeżeli formularz był wypełniony błędnie, zostanie wyświetlony komunikat o błędach. Cały proces jest powtarzany do poprawnego wypełnienia formularza (lub rezygnacji z procesu przez schronisko).



10.3 Obsługa zgłoszeń

Użytkownik poprzez współdzielony moduł ogłoszeń dodaje zgłoszenie. Administrator przez moduł ogłoszeń pobiera zgłoszenie i decyduje o jego następstwie. Jeżeli jest ono odrzucone to równocześnie informowany jest o tym zgłaszający oraz zgłoszenie usuwane jest z bazy. W przeciwnym wypadku zgłaszający dostaje informacje o zatwierdzeniu i dzieje się jedna z możliwych akcji: zablokowanie adoptującego, zablokowanie schroniska lub usunięcie ogłoszenia.



11. Rozwiązania technologiczne

Moduł schroniska, adoptującego i administratora zostanie wdrożony w postaci aplikacji webowej i/lub mobilnej. Zdecydowaliśmy się nie wdrażać aplikacji desktopowej, ponieważ uważamy, że nie jest ona konieczna w przypadku naszych modułów oraz że przeznaczenie naszej aplikacji pasuje bardziej pod profil aplikacji webowej i mobilnej. Jeżeli jednak byłaby taka potrzeba, to zalecamy zaprojektować aplikację we frameworku Flutter. Nasze zalecenia do implementacji pozostałych części:

Język programowania	C# / TypeScript
Serwer	ASP.NET Core
Hosting dla serwera	Azure App Service
Framework dla aplikacji webowej	Angular
Hosting dla aplikacji webowej	Azure App Service
Framework dla aplikacji mobilnej	Flutter
Baza danych	Azure SQL Database
Framework ORM	Entity Framework Core

12. Niezawodność komponentów

Każdy moduł po wysłaniu żądania, oczekuje przez pewien predefiniowany czas na odpowiedź (np. 1s). Jeżeli odpowiedź nie nadejdzie, request zostaje ponowiony określoną ilość razy. W przypadku braku odpowiedzi, użytkownik zostaje poinformowany o niepowodzeniu operacji poprzez stosowny komunikat warstwy prezentacyjnej. Wszystkie requesty do modułów zostają wysłane po akcji użytkownika, dlatego mogą zostać przez niego ponowione w innym momencie. Wszystkie moduły komunikują się ze sobą przy użyciu modułu bazy danych. Sprawia to, że jest to kluczowy moduł potrzebny do sprawnego działania systemu. Jego awaria powoduje problemy w działaniu całej aplikacji. Aby zwiększyć niezawodność całego systemu sugerujemy rozwiązanie opierające się na zastosowaniu rozproszonej bazy danych opartej na replikacji transakcji. Wówczas jeżeli jedna baza danych ulegnie awarii, druga automatycznie przejmie rolę tej uszkodzonej i będzie obsługiwać wszystkie zapytania. Pozostałe moduły nie będą w stanie poprawnie funkcjonować z powodu braku komunikacji. Użytkownicy nie mają możliwości naprawy działania i muszą oczekiwać na naprawę usterki przez administratora.

13. Sposoby symulacji aplikacji użytkowników

Aplikacja umożliwia łatwą symulację działania wielu użytkowników. Odpowiednie konfiguracje agentów pozwoliłyby odwzorować akcję podejmowane normalnie przez człowieka:

- agent schroniska mógłby w określonych odstępach czasu dodawać losowo generowane ogłoszenia oraz losowo akceptować/odrzucać wnioski;
- agent adoptującego mógłby w określonych odstępach czasu pobierać listę ogłoszeń i aplikować na losowe z nich.

Moduł administratora to jedyny element, którego istota bazuje na semantycznym znaczeniu informacji, więc trudnym jest oddzielenie go od człowieka, jednak samo działanie dałoby się zasymulować stosując losową weryfikację zgłoszeń.

14. Komunikacja w systemie

14.1 Wprowadzenie

System dzieli się na cztery moduły: adoptującego, schroniska, administratora oraz współdzielony moduł ogłoszeń. Najważniejszym modułem ze względu na komunikację jest moduł ogłoszeń, ponieważ w dużej mierze stanowi on łącznik pomiędzy pozostałymi modułami. Moduły posiadają REST API, wysyłają oraz odbierają requesty HTTP. Komunikacja wewnętrzna odbywa się natomiast w zdecydowanej większości przypadków za pośrednictwem bazy danych. Zapewnia ona aktualność danych, moduły pobierają dane z bazy i na ich podstawie wykonują swoje zadania. Baza danych nie może samodzielnie wysyłać requestów do innych modułów systemu, jedynie odpowiadać na przychodzące zapytania.

14.2 Dokumentacja API

Zakładanie konta adoptującego

Po uruchomieniu aplikacji użytkownik otrzymuje możliwość zalogowania się lub założenia konta adoptującego albo schroniska. Po wybraniu opcji założenia konta adoptującego użytkownik podaje wymagane dane: imię i nazwisko, email, numer telefonu, hasło oraz adres. Po zatwierdzeniu wysyłany jest request POST /adopter, a moduł bazy po weryfikacji danych tworzy nowy rekord w tabeli adoptujących. Serwer odpowiada informacją, czy udało się utworzyć nowe konto lub czy wystąpiły jakieś problemy. Adoptujący następnie zostaje zalogowany do aplikacji.

Zakładanie konta schroniska

Po uruchomieniu aplikacji użytkownik otrzymuje możliwość zalogowania się lub założenia konta adoptującego albo schroniska. Po wybraniu opcji założenia konta schroniska użytkownik podaje wymagane dane: imię i nazwisko, email, numer telefonu, hasło oraz adres. Po zatwierdzeniu wysyłany jest request POST /shelter, a moduł bazy po weryfikacji danych tworzy nowy rekord w tabeli schronisk. Serwer odpowiada informacją, czy udało się utworzyć nowe konto lub czy wystąpiły jakieś problemy. Schronisko następnie zostaje zalogowane do aplikacji, ale bez możliwości wstawiania ogłoszeń, dopóki nie zostanie zautoryzowane przez administratora. Administrator po zweryfikowaniu schroniska oznacza je jako zweryfikowane przez request PUT /shelter/{shelterId}

Przetwarzanie wniosku o adopcję

Zalogowany użytkownik ma możliwości złożenia wniosku o adopcję jako odpowiedź na ogłoszenie wystawione przez pewne schronisko. W pierwszej kolejności wysyłany jest request GET na /announcements/{announcementId}/available, aby sprawdzić, czy ogłoszenie jest nadal aktualne. Jeżeli nie jest, to użytkownik dostaje stosowny komunikat i jest przekierowywany na stronę z listą ogłoszeń. Wniosek jest przesyłany poprzez request POST na /applications. W przypadku, gdy dana osoba była już wcześniej weryfikowana przez schronisko, to automatycznie ogłoszenie jest zamykane, a sam wniosek zaakceptowany poprzez request PUT na /applications/{applicationId}/accept. W przeciwnym razie schronisko rozpoczyna swój własny proces weryfikacji adoptującego. Serwis nie wnika w przebieg tej weryfikacji. Jej wynik schronisko przesyła za pomocą requestu PUT na /adopter/{adopterId}. W przypadku pozytywnego rozpatrzenia ogłoszenie jest zamykane, a adoptujący poinformowany drogą mailową o możliwości odbioru zwierzątka.

Dodawanie ogłoszenia

Każde schronisko w aplikacji może stworzyć ogłoszenie z ofertą oddania zwierzątka do adopcji. Wykorzystywana jest do tego metoda POST / announcements, która w swoim ciele przekazuje ID ogłoszenia. ID połączone jest z danymi na temat ogłoszenia: ID, tytuł, opis, data utworzenia i nullowalna data zamknięcia oraz ID zwierzątka, które zostało utworzone już wcześniej.

Werfykacja adoptującego

W trakcie procesu adopcji użytkownik, jeżeli nie adoptował wcześniej zwierzątka, musi zostać zweryfikowany przez schronisko. Odbywa to się poza aplikacją. Przedstawiciel schroniska spotyka się z potencjalnym adoptującym, sprawdza czy w jego mieszkaniu panują odpowiednie warunki itp. Następnie wywoływana jest

metoda POST /adopter/{adopterId}/authorize, która zapisuje w bazie informację o pomyślnym przejściu weryfikacji i pozwala dokończyć proces adopcji.

Obsługa zgłoszeń

Każdy adoptujący w aplikacji może stworzyć zgłoszenie na temat ogłoszenia lub schroniska, a schronisko może zgłosić adoptującego. Wykorzystywana jest do tego metoda POST /reports, która w swoim ciele przekazuje informacje o zgłoszeniu: jego typ (określa, czy dotyczy ono ogłoszenia, schroniska czy adoptującego), identyfikator celu (odpowiednio ogłoszenia, schroniska lub adoptującego, w zależności od typu), wiadomość opisującą problem oraz stan „new". Administrator wykorzystuję metodę GET /reports, żeby przeglądać zgłoszenia w stanie new i następnie może je zaakceptować lub odrzucić wykorzystując metodę PUT /reports przekazując nowy stan w ciele: „accepted" lub „declined". W przypadku zaakceptowania wykorzystywana jest odpowiednia metoda PUT /shelter/{shelterId}, /announcements/{announcementId} lub /adopter/{adopterId} do zablokowania odpowiedniego schroniska, ogłoszenia lub adoptującego.

Usuwanie danych

Wszelkie operacje usuwania powinny być wykonywane poprzez zmianę stanu obiektu w bazie na usunięty poprzez odpowiednią metodę PUT. Polepsza to działanie modułu administratora, ponieważ treści nieodpowiednie będą mogły być wciąż weryfikowane przez administratora, nawet po ich usunięciu przez użytkownika. Wpisy w bazie, które mają stan usunięty są automatycznie usuwane po 1 miesiącu od ich ostatniej modyfikacji w celu zapobiegnięcia zaśmiecaniu bazy danych.

14.3 Kod RAML

```
#%RAML 1.0
title: PetShareApi
baseUri: https://share.pet/api
version: 1.0

securitySchemes:
  ApiKeyAuth:
    type: apiKey
    in: header
    name: api-key
    description: Autoryzacja kluczem API zawierającym informację o
    o roli posiadacza. Istniejące role: adoptujący, schronisko, administrator.

schemas:
  User:
    description: Użytkownik - klasa
    type: object
    properties:
      IsAuthorized:
        type: boolean
        example: true
        description: Czy użytkownik przeszedł weryfikację
      UserName:
        type: string
        example: Piotr Nowak
        description: Nazwa adoptującego
      PhoneNumber:
        type: string
        example: 512945243
        description: Numer telefonu
```

```
    Email:
      type: string
      example: p.nowak@gmail.com
      description: Adres email
    Address:
      allOf:
      - $ref: #/schemas/Address
      description: Adres zamieszkania
  required:
    - UserName
    - PhoneNumber
    - Email
    - Address
Address:
  description: Adres - klasa
  type: object
  properties:
    Street:
      type: string
      example: Wesoła 4
      description: Ulica
    City:
      type: string
      example: Warszawa
      description: Miasto
    Province:
      type: string
      example: Mazowieckie
      description: Województwo
    PostalCode:
      type: string
      example: 12-345
      description: Kod pocztowy
    Country:
      type: string
      example: Polska
      description: Kraj
  required:
    - Street
    - City
    - Province
    - PostalCode
    - Country
Application:
  description: Aplikacja - klasa
  type: object
  properties:
    ID?:
      type: integer
      description: ID aplikacji
```

```
    DateOfAplication:
      type: datetime
    IsWithdrawed:
      type: boolean
    LastUpdateDate:
      type: DateTime
      example: 2018-02-28T16:41:41.090Z
      description: Data ostatniej aktualizacji wniosku
    User:
      allOf:
      - $ref: #/schemas/User
    Announcement:
      allOf:
      - $ref: #/schemas/Announcement
    required:
      - DateOfAplication
      - User
      - LastUpdateDate
      - Announcement
Announcement:
  description: Ogłoszenie - klasa
  type: object
  properties:
    ID?:
      type: integer
      description: ID schroniska
    Title:
      type: string
      description: Tytuł ogłoszenia
      example: Oddam kotka
    Decription:
      type: string
      description: Opis ogłoszenia
      example: Oddam kotka do dobrego domu ładnego
    CreationDate:
      type: DateTime
      example: 2018-02-28T16:41:41.090Z
      description: Data wystawienia ogłoszenia
    ClosingDate:
      type: DateTime
      example: 2018-02-28T16:41:41.090Z
      description: Data zamknięcia ogłoszenia
    Status:
      enum: [Otwarte, Zamknięte, W trakcie weryfikacji, Usunięte]
      type: string
      example: Otwarte
      description: Status ogłoszenia
    LastUpdateDate:
      type: DateTime
      example: 2018-02-28T16:41:41.090Z
```

```
      description: Data ostatniej aktualizacji ogłoszenia
    IDPet:
      type: integer
      description: ID zwierzątka
      $ref: #/types/Pet
    required:
      - ID
      - CreationDate
      - Status
      - LastUpdateDate
      - IDPet
Pet:
  description: Zwierzątko - klasa
  type: object
  properties:
    ID:
      type: integer
      description: ID zwierzątka
    Name:
      type: string
      description: Imię zwierzątka
      example: Puszek
    Species:
      type: string
      description: Gatunek zwierzątka
      example: kot
    Breed:
      type: string
      description: Rasa zwierzątka
      example: norweski leśny
    Birthday:
      type: DateTime
      example: 2018-02-28T16:41:41.090Z
      description: Data urodzin zwierzątka
    Description:
      type: string
      description: Opis zwierzątka
      example: Ładny kotek ale wredny
    Photo:
      description: Zdjęcia zwierzątka
      type: byte[]
Report:
  description: Zgłoszenie - struktura
  type: object
  properties:
    reportType:
      type: string
      enum: [adopter, announcement, shelter]
      example: announcement
      description: Typ zgłoszenia (kogo dotyczy)
```

```yaml
      targetId:
        type: integer
        example: 1
        description: identyfikator celu zgłoszenia
      message:
        type: string
        example: Obraźliwy opis ogłoszenia :(
        description: Opis zgłoszenia
      state:
        type: string
        enum: [new, declined, accepted]
        example: new
/announcements:
  get:
    description: Zwróć wszystkie ogłoszenia spełniające filtry
    queryParameters:
        species:
          description: Gatunek zwierzątka
          type: string
          required: false
          example: Pies
        breed:
          description: Rasa zwierzątka
          type: string
          required: false
          example: Norweski leśny
        location:
          description: Miejscowość schroniska
          type: string
          required: false
          example: Warszawa
        age:
          description: Wiek zwierzątka w dniach
          type: int
          required: false
          example: Warszawa
        shelterName:
          description: Nazwa schroniska
          type: string
          required: false
          example: Schronisko na Paluchu
    responses:
      200:
        description: Zwrócono ogłoszenia
        body:
          application/json:
            type: Announcement[]
      400:
        description: Pobieranie ogłoszeń nie powiodło się
      401:
```

```
      description: Brak dostępu
post:
  description: Utworzenie ogłoszenia
  requestBody:
    description: Dane nowego ogłoszenia
    required: true
    body:
      application/json:
        type: Announcement
  responses:
    201:
      description: Ogłoszenie zostało dodane
      body:
        application/json:
          type: integer
          description: Identyfikator ogłoszenia
    400:
      description: Ogłoszenie nie zostało utworzone
/{announcementId}:
  get:
    description: Zwróć ogłoszenie o podanym id
    responses:
      201:
        description: Zwrócono ogłoszenie
        body:
          application/json:
            type: Announcement
      400:
        description: Pobieranie ogłoszenia nie powiodło się
      401:
        description: Brak dostępu
  put:
    description: Uaktualnia ogłoszenie, używane do blokowania
    ApiKeyAuth: [Admin]
    content:
      application/json:
        schema:
          $ref: #/schemas/Announcement
    responses:
      200:
        description: OK
        content:
          application/json:
            schema:
              $ref: #/schemas/Announcement
      403:
        description: Brak uprawnień administratora
  /available:
    get:
      description: Czy oferta jest nadal aktualna?
```

```yaml
        security:
          ApiKeyAuth: [Adopter]
        responses:
          200:
            description: Oferta aktualna
          201:
            description: Oferta nieaktualna
          400:
            description: Sprawdzenie stanu dostępności nie powiodło się
          401:
            description: Brak dostępu
/applications:
  post:
    description: Utworzenie nowego wniosku
    requestBody:
      description: Dane nowego wniosku
        required: true
        content:
          application/json:
            schema:
            $ref: #/schemas/Application
        responses:
          200:
            description: Aplikacja została stworzona
            content:
              application/json:
                schema:
                  $ref: #/schemas/id
                  description: Identyfikator aplikacji
          400:
            description: Aplikacja nie została utworzona
  /{applicationId}:
    /accept:
      put:
        description: Wniosek zaakceptowany
            security:
            ApiKeyAuth: [Shelter]
            responses:
            200:
              description: Ok
            400:
              description: Zmiana statusu nie powiodła się
            401:
              description: Brak dostępu
    /reject:
      put:
        description: Wniosek odrzucony
            security:
            ApiKeyAuth: [Shelter]
            responses:
```

```
            200:
              description: Ok
            400:
              description: Zmiana statusu nie powiodła się
            401:
              description: Brak dostępu
/reports:
  get:
    ApiKeyAuth: [Admin]
    decription: Gets all reports without answer
    responses:
      200:
        description: OK
        body:
          application/json:
            schema:
              type: schemas.Report[]
      403:
        description: Brak uprawnień administratora
  post:
    description: Dodaje nowe zgłoszenie
    body:
      application/json:
        type: Report
    responses:
      200:
        description: OK
      400:
        description: Bad request
      408:
        description: Timeout
  put:
    decription: Uaktualnia zgłoszenie
    ApiKeyAuth: [Admin]
    content:
      application/json:
        schema:
          $ref: #/components/schemas/Report
    responses:
      200:
        description: OK
        content:
          application/json:
            schema:
              $ref: #/components/schemas/Report
      403:
        description: Brak uprawnień administratora
/shelter:
  get:
    description: Pobiera wszystkie schroniska
```

```yaml
  body:
    application/json:
      type: Shelter[]
  responses:
    200:
      description: OK
    400:
      description: Bad request
    408:
      description: Timeout
post:
  description: Dodaje nowe schronisko
  body:
    application/json:
      type: Shelter
  responses:
    200:
      description: OK
    400:
      description: Bad request
/{shelterId}:
  get:
    description: Pobiera schronisko o danym id
    content:
      application/json:
        schema:
          $ref: #/schemas/Shelter
    responses:
      200:
        description: OK
        content:
          application/json:
            schema:
              $ref: #/schemas/Shelter
      400:
        description: Bad request
  put:
    description: Uaktualnia schronisko, używane do blokowania i weryfikacji
    ApiKeyAuth: [Admin]
    content:
      application/json:
        schema:
          $ref: #/schemas/Shelter
    responses:
      200:
        description: OK
        content:
          application/json:
            schema:
              $ref: #/schemas/Shelter
```

```
        403:
          description: Brak uprawnień administratora
/adopter:
  get:
    description: Pobiera wszystkich adoptujących
    ApiKeyAuth: [Admin]
    body:
      application/json:
        type: Adopter[]
    responses:
      200:
        description: OK
      400:
        description: Bad request
      403:
        description: Brak uprawnień administratora
  post:
    description: Dodaje nowego adoptującego
    body:
      application/json:
        type: Adopter
    responses:
      200:
        description: OK
      400:
        description: Bad request
      408:
        description: Timeout
  /{adopterId}:
    get:
      description: Pobiera adoptującego o danym id
      content:
        application/json:
          schema:
            $ref: #/schemas/User
      responses:
        200:
          description: OK
          content:
            application/json:
              schema:
                $ref: #/schemas/User
        400:
          description: Bad request
    put:
      description: Uaktualnia użytkownika, używane do blokowania i weryfikacji
      ApiKeyAuth: [Admin]
      content:
        application/json:
          schema:
```

```
            $ref: #/schemas/User
      responses:
        200:
          description: OK
          content:
            application/json:
              schema:
                $ref: #/schemas/User
        403:
          description: Brak uprawnień administratora
    /authorize:
      post:
        description: Zakończenie weryfikacji adoptującego
        ApiKeyAuth: [Shelter]
        responses:
          200:
            Description: OK
          403:
            description: Brak uprawnień schroniska
/pet:
  post:
    description: Dodaje nowego zwierzaka
    body:
      application/json:
        type: Pet
    responses:
      200:
        description: OK
      400:
        description: Bad request
      408:
        description: Timeout
  get:
    description: Zwraca wszystkie zwierzęta dla danego schroniska
    ApiKeyAuth: [Shelter]
    body:
      application/json:
        type: Pet[]
    responses:
      200:
        description: OK
      400:
        description: Bad request
      403:
        description: Brak uprawnień schroniska
  /{petId}:
    get:
      description: Zwraca zwierzątko o danym id
      body:
        application/json:
```

```
      type: Pet
  responses:
    200:
      description: OK
    400:
      description: Bad request
put:
  description: Uaktualnia dane zwierzaka
  ApiKeyAuth: [Shelter]
  content:
    application/json:
      schema:
        $ref: #/schemas/Pet
  responses:
    200:
      description: OK
      content:
        application/json:
          schema:
            $ref: #/schemas/Pet
    403:
      description: Brak uprawnień schroniska
```