Dokumentacja projektu WeatherApp

Frameworki biznesowe

Łukasz Kurant
Piotr Małek
Łukasz Lipiński

15 czerwca 2021

Opis projektu

Głównym założeniem projektu jak udostępnienie narzędzia do udostępniania stanu pogody przez użytkowników. Każda osoba może przypisać się do wybranej lokalizacji a następnie udostępniać z jej poziomu informacje o pogodzie w swoim otoczeniu (zwłaszcza w przypadkach anomalii pogodowych jak np. Burze). Po stronie użytkownika zostanie to przedstawione jako pin na interaktywnej mapie. Inni użytkownicy aplikacji mają możliwość oceny zgłoszonego pinu czy zgłoszenie go w przypadku łamania zasad.

Każdy użytkownik jest uwierzytelniany w procesie rejestracji i logowania.

Narzędzia użyte w projekcie

Backend

Spring Framework (Spring boot + Maven)

Hibernate

Spring security

JWT

Frontend

React.js

Leaflet.js (Obsługa map)

Axios

Baza danych

MySQL

Główne funkcjonalności w projekcie

Obsługa użytkownika

- Rejestracja
- Logowanie
- Zmiana hasła
- Zmiana lokalizacji użytkownika
- Role użytkowników (zwykły user, redaktor, administrator)
- Nadawanie i odbieranie roli użytkownika przez administratora

Obsługa pinów

- Tworzenie nowego pinu / Edycja / usuwanie
- Możliwość polubienia pina
- Zgłaszanie pinów nie spełniających wymogów

Obsługa zgłoszeń

Akceptowanie lub odrzucanie zgłoszenia

Role użytkowników

Zwykły użytkownik - możliwość zmiany hasła, dodawania pinów, zgłaszania pinów czy oceniania pinów innych.

Redaktor - uprawnienia zwykłego użytkownika + możliwość akceptowania lub odrzucania zgłoszeń użytkowników.

Administrator - uprawnienia redaktora + możliwość przypisywania i odbierania roli redaktora użytkownikom.

Przypadki użycia

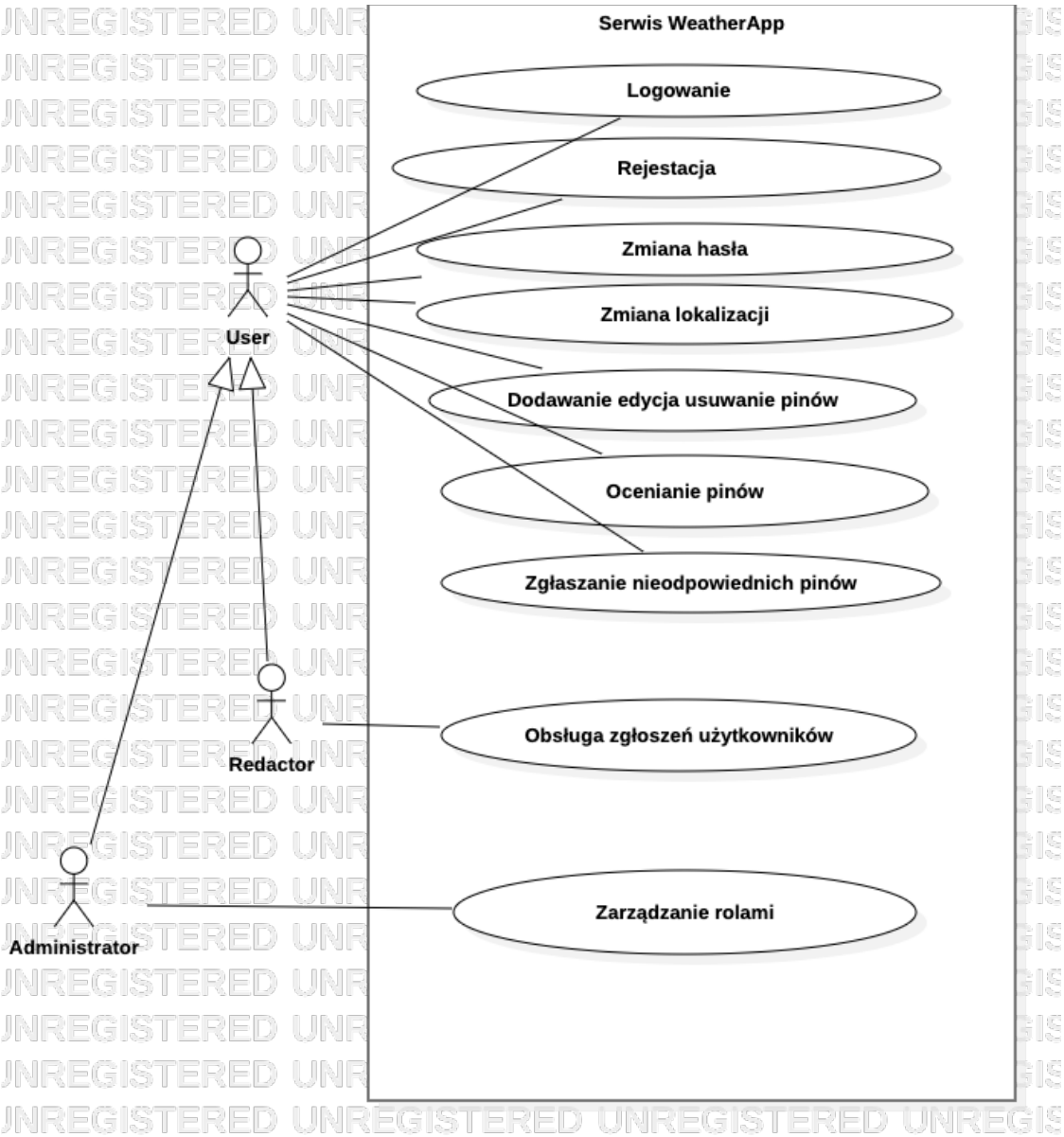
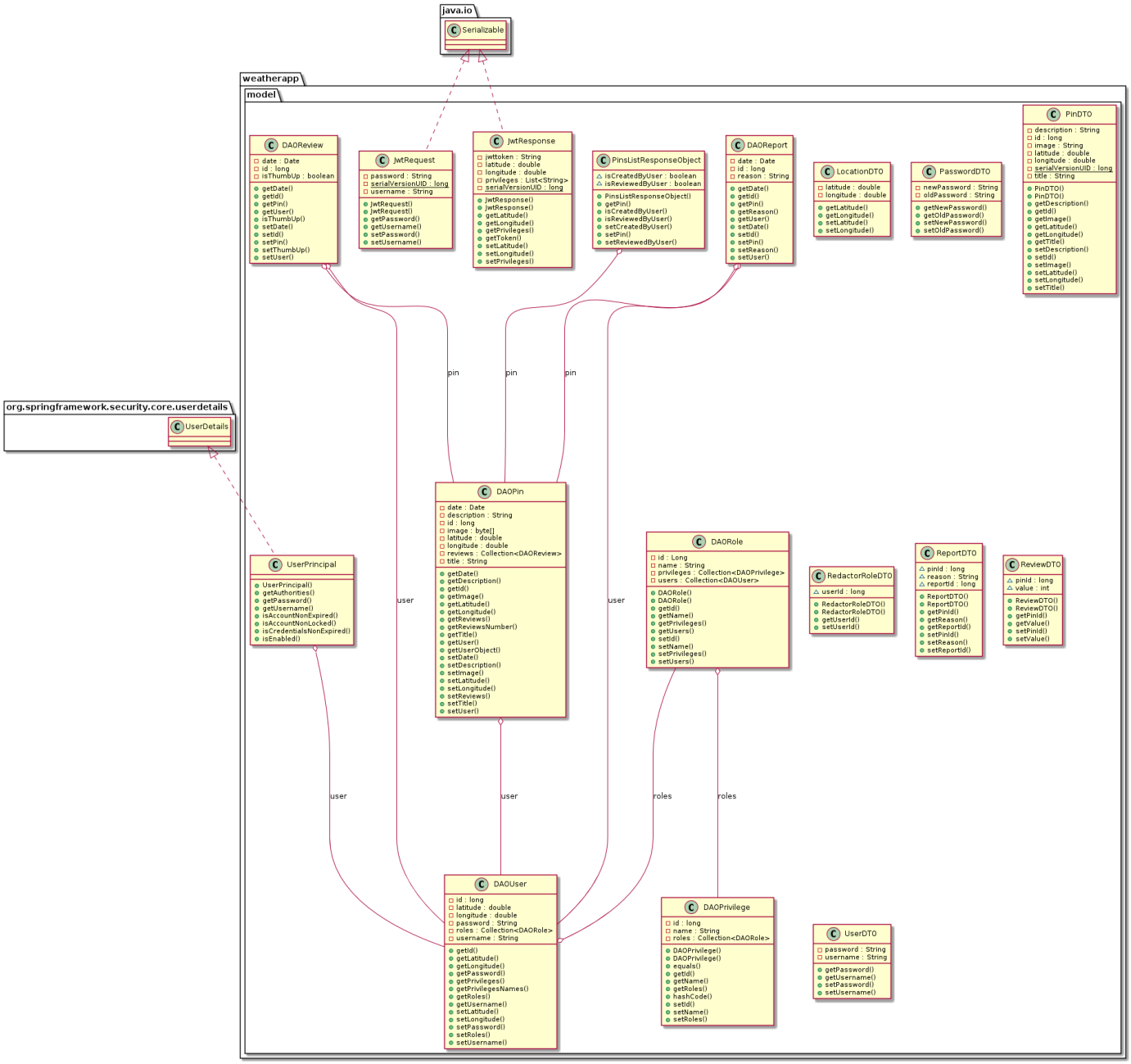


Diagram UML klas modeli w projekcie

(Diagramy w dobrej jakości dostępne w katalogu z dokumentacją)



Spis modeli

DAOUser - modle użytkownika zawierający informacje potrzebne do jego uwierzytelnienia, ale również jego lokalizację oraz zbiór ról.

DAORole, DAOPrivilege - Każda rola w projekcie (User, Redaktor, Administrator) posiada własny obiekt roli. Do każdej roli są przepisywane uprawnienia, np.

- ***CREATE_PINES_PRIVILEGE*** - możliwość dodawania pinów
- ***MODERATE_PINES_PRIVILEGE*** - możliwość moderowania zgłoszeń pinów
- ***MODERATE_REDACTORS_PRIVILEGE*** - możliwość zarządzania rolami redaktorów.

UserPrincipal - Klasa konieczna do zarządzania użytkownikami z poziom Spring framework. Jest to kolejna warta na klasie DAOUser używanej w logice biznesowej aplikacji.

DAOPin - model pinu z relacją użytkownika o pogodzie. Zawiera informacje gdzie dana relacja się znajduje, zdjęcie, opis, itd.

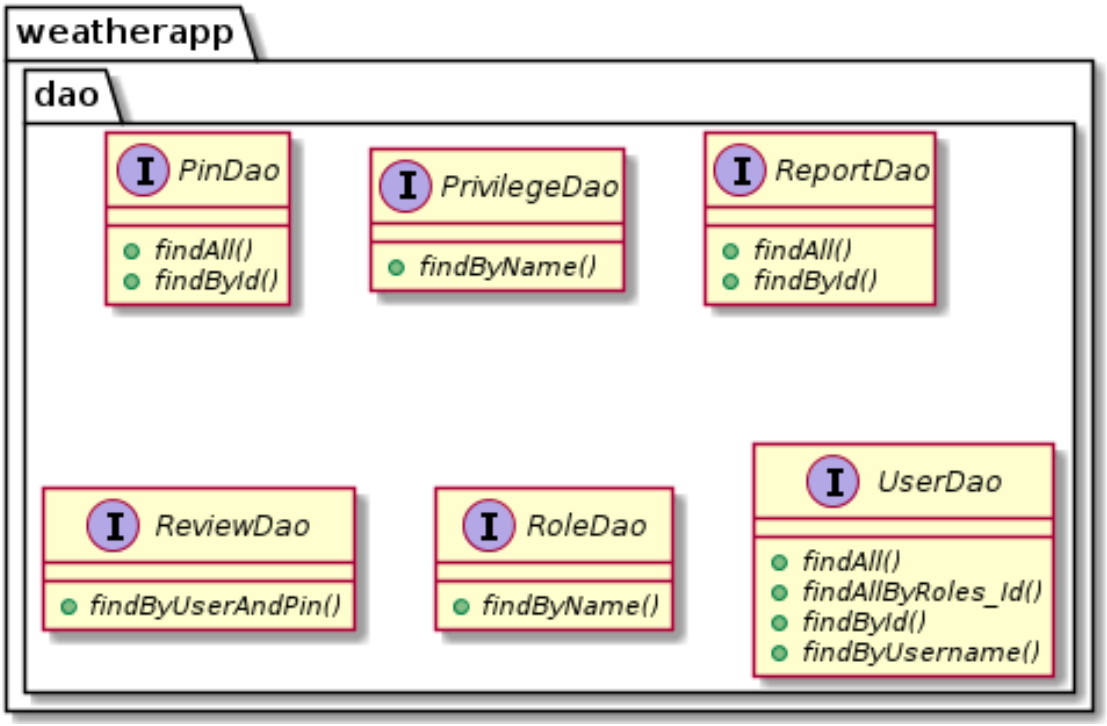
DAOReview - Klasa odpowiedzialna za przechowywanie informacji o ocenie pinu przez użytkownika

DAOReport - Klasa przechowująca informacje o zgłoszeniu pinu przez użytkownika, zawierająca informacje o pinie, użytkowniku czy szczegółach użytkownika.

JwtRequest / JwtResponse - klasy będące wejściem i wyjściem dla requestów związanych z uwierzytelnieniem użytkownika z użyciem tokenów JWT.

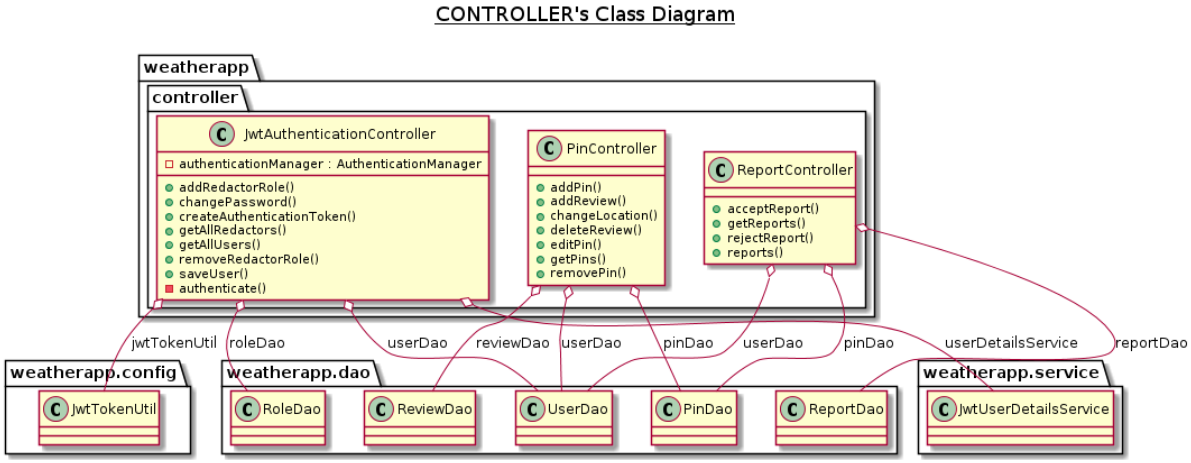
Klasy z przyrostkiem „*DTO” - klasy używane do serializacji obiektów przysyłanych do endpointów lub z nich wysyłanych.

DAO's Class Diagram



Interfejsy DAO służą do tworzenia definicji zapytań do bazy danych i umożliwiające obsługę komunikacji z bazą danych.

Schemat kontrolerów



Kontrolery to inaczej zbiory metod (endpointów) w których zawiera się cała logika biznesowa aplikacji.

JwtAuthenticationController - odpowiada za obsługę użytkowników, sesje, generowanie tokenów JWT, obsługę ról czy zmianę hasła.

PinController - zawiera zbiór metod odpowiedzialnych za działania na pinach (relacjach użytkowników).

ReportController - odpowiada za obsługę zgłoszeń pinów przez użytkowników.

Opis enepointów

Szczegółowy spis endpointów wraz z przykładami użycia znajduje się w katalogu z dokumentacją.

Wersjonowanie API

W przypadku generowania nowej wersji Ani, najlepszym scenariuszem jest podmiana wersji w linku, np.

https://www.example.com/api/1/products

Wewnętrzna wersja API korzysta z formatu 1.2.3, więc wygląda następująco:

MAJOR.MINOR.PATCH

Major: wersja główna używana w identyfikatorze URI i oznacza istotne zmiany w interfejsie API. Wewnętrznie nowa wersja główna oznacza utworzenie nowego interfejsu API, a numer wersji jest używany do kierowania do właściwego hosta.
Minor i patch: są niewidoczne dla klienta i używane wewnętrznie do aktualizacji zgodnych z poprzednimi wersjami.

Uważamy że w przypadku tak małej aplikacji jest to najprostszy sposób na umożliwienie klientom dostęp do zasobów, jednocześnie mając na względzie ż może to powodować niepotrzebne rozgałęzienia kodu.

W naszej aplikacji obecnie nie mamy tak skonfigurowanego wersjonowania ze względu na proste lokalne środowisko testowe (wersjonowanie mogłoby zostać dodane na poziomie proxy).