Politechnika Warszawska
Wydział Matematyki i Nauk Informacyjnych

PRACA DYPLOMOWA INŻYNIERSKA
NA KIERUNKU INFORMATYKA

GEOPORTAL

GEOPORTAL

Autor:
Aneta Rosłan

Współautorzy:
Paweł Krawczyk
Maciej Wierzchowski

Promotor:
mgr inż. Michał Okulewicz

Opiekun naukowy:
prof. dr hab. Jacek Mańdziuk

Warszawa, styczeń 2013

....

podpis promotora podpis autora

Streszczenie

Jeden obraz mówi więcej niż tysiąc słów.

Tematem naszej pracy dyplomowej było stworzenie systemu przechowującego i udostępniającego dane przestrzenne — geoserwera (serwera mapowego). Za najważniejsze cele postawiliśmy sobie zgodność ze standardami popularnych usług, łatwość konfiguracji serwera, wydajność oraz możliwość rozbudowy systemu. W efekcie powstał WUTGeoserver, który spełnia nasze założenia i może być podstawą do stworzenia wielofunkcyjnego, rozbudowanego systemu geoinformacyjnego. Serwer ten jest nieodłączną częścią niniejszej pracy.

Ważną kwestią związaną z danymi przestrzennymi są opisane na początku niniejszej pracy podstawowe zagadnienia geodezyjno-kartograficzne: sposoby opisywania położenia obiektów na powierzchni Ziemi oraz tworzenia reprezentacji dwuwymiarowej tych danych (odwzorowania bryły ziemskiej na płaską powierzchnię papieru lub monitora). W kolejnych rozdziałach pokrótce przedstawiliśmy kilka istniejących aplikacji i systemów geoinformacyjnych ważnych dla naszej pracy ze względu na popularność i zastosowanie (do przeglądania, tworzenia i edycji danych przestrzennych). Omówiliśmy popularne formaty danych przestrzennych obsługiwanych w naszym serwerze oraz zaimplementowane standardy udostępniania tych danych przez sieć.

Istotnym założeniem pracy była zgodność ze standardami usług udostępniania danych przestrzennych. Ogólnie przyjętymi, udokumentowanymi standardami są Web Map Service (WMS) oraz Web Feature Service (WFS) opracowane przez Open Geospatial Consortium. Jednocześnie zależało nam na możliwości współdziałania WUTGeoservera z aplikacjami wykorzystywanymi w Polsce (zarówno powszechnie znanymi na całym świecie, jak QuantumGIS czy produkty firmy Google, jak i naszym krajowym geoportal.gov.pl). Jednak okazało się, że choć istnieją jednoznaczne standardy, to oprogramowanie — nawet oficjalnie je implementujące — nie zawsze działa zgodnie z nimi.

Dokumentacje: architektoniczna, techniczna i użytkownika stanowią dużą część naszej pracy pisemnej. Poza tym pełna dokumentacja Javadoc znajduje się na płycie z zamieszczonymi plikami źródłowymi. Staraliśmy się stworzyć możliwie szczegółową dokumentację, aby

ułatwić potencjalną rozbudowę systemu i ingerencję w jego kod. Powodem tej skrupulatności jest pierwotnie podjęta próba rozwinięcia istniejącego serwera open-source (Geoserver), z której zrezygnowaliśmy między innymi z powodu braku dokumentacji.

Całą aplikację napisaliśmy od podstaw w technologii Java, wykorzystując serwer JBoss i bazę danych PostgreSQS, a także szereg technologii i bibliotek omówionych w pracy. Zastosowaliśmy wzorzec master-slave aby stworzyć skalowalny system, umożliwiający zdefiniowanie węzła centralnego oraz węzłów potomnych w celu poprawy wydajności. Dołożyliśmy starań, aby naszą aplikację dało się łatwo wdrożyć, a osiągnięte rezultaty pod względem wydajności pozwalają nam sądzić, że znajdzie ona zastosowanie jako w pełni funkcjonalny system geoinformatyczny.

Abstract

A picture is worth a thousand words

The subject of our bachelor's thesis was to create the system, which stores and shares spatial data – geoserver (map server). Our most important goals were compliance with standards of popular services, ease of sever configuration, high performance and the possibility of future system development. As a result we have created the WUTGeoserver, which fulfills our goals and may be the basis for multifunctional and highly developed geoinformation system. This server is an integral part of this thesis.

An important matter associated with spatial data are described at the beginning of this paper basic geodetic and cartographic issues: how to describe the position of objects on the surface of the Earth, and create a two-dimensional representation of the data (mapping of the Earth globe on a flat surface of the paper or the monitor). In subsequent chapters we briefly presented several of existing applications and geoinformation systems important for our thesis due to their popularity and use to view, create and edit spatial data. We discussed popular data formats supported in our server and implemented standards for sharing these data over network. An important point of this project was the compliance with norms of services for sharing spatial data. Generally accepted, well documented standards are Web Map Service (WMS) and Web Feature Service (WFS) developed by Open Geospatial Consortium. At the same time we wanted that our WUTGeoserver interoperability with other applications used in Poland (commonly known around the world, such as QuantumGIS, as well as our national geoportal.gov.pl). However, it turned out that although there are clear standards, the software — even officially implementing them — do not always act in accordance with them.

Documentation: architectural, technical and user are large part of our paper work. Also the full Javadoc documentation is included on the CD with the source files of our application. We tried to make as detailed documentation as possible to facilitate potential system development and the interference in its source code. The reason our conscientiousness was the first attempt to develop existing ,open-source server (Geoserver), which we abandoned due to lack of documentation and technical support.

The whole application we have created from the basis in Java technology, using application server JBoss, PostgreSQL database and also a range of technologies and libraries described in this paper. We use master-slave architectural pattern to create a scalable system that allows defining central node and child nodes to improve performance. We have tried to provide that our application could be easily implemented. The results achieved in terms of performance allows us to believe that it will be used as a fully functional geoinformation system.

Spis treści

1 Wstęp

Współcześnie coraz większą popularnością cieszą się udostępniane w Internecie różnego rodzaju interaktywne mapy. Chyba najlepiej znanym przykładem jest działający od 2005 roku międzynarodowy serwis Google Maps[14]. Umożliwia on między innymi obejrzenie zdjęć lotniczych i satelitarnych Ziemi, widok mapy topograficznej wraz z siecią dróg, ma wbudowaną wyszukiwarkę obiektów po nazwach i adresach oraz opcję znajdowania trasy wzdłuż punktów. Do innych znanych serwisów tego typu należą międzynarodowe Bing[18] i OpenStreetMap[16] oraz polskie Zumi[17] i Targeo[19]. Popularnością cieszą się aplikacje takie jak Google Earth[15], które umożliwiają obejrzenie trójwymiarowego widoku Ziemi. Coraz częściej również administracja publiczna udostępnia mapy ogólnogeograficzne i tematyczne za pośrednictwem serwisów takich jak państwowy Geoportal[13] (mapy topograficzne), EkoMapa Ministerstwa Środowiska[20] (mapy środowiskowe) i Ikar Państwowego Instytutu Geologicznego[21] (mapy geologiczne).

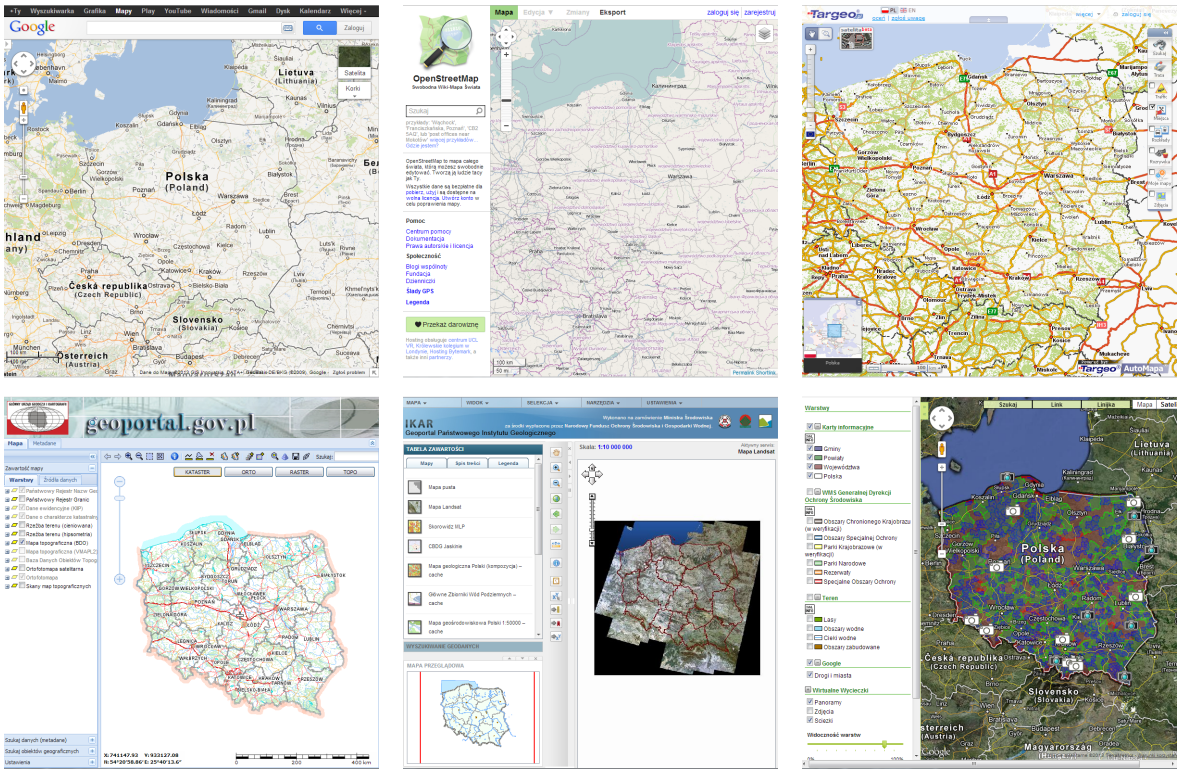
Niektóre z tych serwisów należą do przedsiębiorstw, niektóre są na licencji open source, jeszcze inne podlegają administracji rządowej i samorządowej. Łączy je fakt, że wszystkie udostępniają dane przestrzenne, głównie w postaci obrazów map.

Administracja państwowa posiada bardzo pokaźne zbiory różnych danych przestrzennych, które powoli w coraz szerszym zakresie są udostępniane. Jest to bardzo atrakcyjne dla obywateli, a jednocześnie generuje zapotrzebowanie na aplikacje serwerowe spełniające określone wymagania. W naszej pracy staraliśmy się uwzględnić specyfikę tej grupy odbiorców, biorąc pod uwagę zwłaszcza stawiane im wymagania prawne. Na poziomie krajowym kwestię danych przestrzennych reguluje Ustawa o infrastrukturze informacji przestrzennej[10]. Drugim bardzo ważnym dokumentem jest unijna dyrektywa INSPIRE [11], która jest próbą stworzenia spójnego jednolitego podejścia do kwestii wyszukiwania, przeglądania, pobierania i przekształcania danych przestrzennych.

Oprócz obowiązujących przepisów prawnych istnieją też międzynarodowe standardy formatów danych przestrzennych oraz usług ich udostępniania.

Celem niniejszej pracy było stworzenie aplikacji serwerowej, która udostępnia dane przestrzenne zgodnie ze wspomnianymi standardami, pozwalającej na import i eksport popularnych formatów, realizującej usługi wyszukiwania, przeglądania i pobierania zgodne z wytycznymi technicznymi dyrektywy INSPIRE. Serwer miał też mieć możliwość skonfigurowania go w postaci węzła centralnego (referencyjny bank danych) i węzłów potomnych (punkty dostępowe) w celu uzyskania skalowanego pod względem wydajności rozwiązania.

Pierwotnie zamierzaliśmy przystosować do własnych potrzeb istniejące rozwiązanie open-source, jednakże ostatecznie stworzyliśmy od podstaw własną aplikację o nazwie WUTGeoserver.



Rysunek 1.1: Popularne serwisy mapowe (od lewej od góry): Google Maps, Open Street Map, Targeo, Geoportal, Ikar2, Ekomapa

2 Mapy i dane przestrzenne

Zanim przejdziemy do technicznego opisu systemu, przedstawimy pokrótce problem opisywania wielkości i kształtu Ziemi, znajdujących się na niej obiektów oraz metod przedstawiania ich na mapach. Znajomość tych geodezyjnych i kartograficznych podstaw jest niezbędna do zrozumienia, czym tak naprawdę są archiwizowane dane i co się z nimi dzieje podczas działania aplikacji.

2.1. Mapy

Mapy są wygodnym narzędziem przedstawiającym powierzchnię Ziemi lub jej fragment jako obraz, na którym za pomocą powszechnie znanych reguł i określonych symboli odwzorowane są różne obiekty i zjawiska. Pozwalają na pierwszy rzut oka określić lokalizację i przestrzenne zależności między przedstawionymi elementami, co czyni z nich niezwykle cenny materiał informacyjny oraz atrakcyjny sposób przeglądania i przyswajania danych związanych z przestrzenią.

Przewłocki (2008) [2] definiuje **mapę** jako „zmniejszony, uogólniony, metamatycznie odwzorowany na płaszczyznę obszar określonego obszaru Ziemi, jednoznacznie orientujący w położeniu przestrzennym obiekty występujące na tym obszarze".

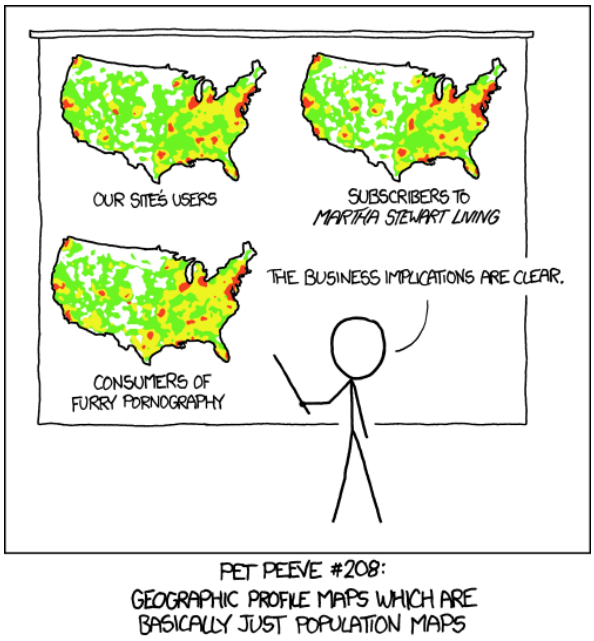
Mapy są powszechnie stosowane w wielu dziedzinach nauki. Od najmłodszych lat spotykamy się z nimi w szkole na zajęciach z geografii i historii. Wyniki większości prac geodezji są przedstawiane w postaci map. Niezbędne w pracy administracji choćby podczas ewidencji gruntów i budynków, wykorzystywane przez przedsiębiorstwa i instytucje w celu zarządzania nieruchomościami (rejestr infrastruktury, np. położenie sieci wodociągowych). Mapy są też powszechnie wykorzystywane do przeróżnych celów przez prywatnych użytkowników.

Mapa jest intuicyjnym sposobem przedstawienia danych przestrzennych. Jednak niesie ze sobą wiele ograniczeń — warunkiem jej czytelności i przydatności jest w dużej mierze ilość i zróżnicowanie prezentowanych danych[1]. Oznacza to, że ze zbioru pierwotnych danych

[1]Nie są to oczywiście jedyne problemy. Istotne są również przygotowanie danych, dobór odpowiedniej metody prezentacji oraz symboli i ich cech w kontekście konkretnych danych oraz całego obrazu (nałożenie się różnych kolorów lub tekstur). Zagadnieniem tworzeniem map zajmuje się nauka zwana kartografią.

zazwyczaj tylko niewielka część może zostać umieszczona na mapie, a i tak prawdopodobnie będzie po drodze poddana generalizacji.

Dawno minęły czasy, gdy mapy i dane na podstawie których powstawały miały jedynie postać papierową. W epoce komputerów coraz większa ich liczba przechowywana jest w postaci cyfrowej. Pozwala to na zgromadzenie dużej ilości dokładnych i precyzyjnych danych, które następnie można przetwarzać w celu uzyskania informacji pochodnej. W dalszej części pracy zajmować się będziemy tylko mapami i danymi w postaci cyfrowej.



Rysunek 2.1: Przykład map z niepoprawnie opracowanymi danymi (źródło: xkcd[24])

2.2. Dane przestrzenne

Ustawa o infrastrukturze informacji przestrzennej (2010)[10] określa, że **dane przestrzenne** są to „dane odnoszące się bezpośrednio lub pośrednio do określonego położenia lub obszaru geograficznego". Definicję tę doprecyzowuje Przewłocki (2008)[2] — jako „informacje o położeniu, geometrycznych własnościach i przestrzennych relacjach obiektów, które mogą być identyfikowane w odniesieniu do powierzchni Ziemi".

Na dane o obiekcie składają się informacje przestrzenne (geometria, położenie, relacje) oraz jego pozostałe atrybuty. Obiekty jednej lub kilku klas można grupować w warstwy. Przykładowo warstwa hydrologia może obejmować obiekty rzek, których geometrią będzie bieg rzeki, a atrybutami nazwa i klasa czystości.

2.2.1. Wykorzystanie

Dane przestrzenne są zbierane, archiwizowane a następnie wykorzystywane przez różne podmioty, zarówno administrację, instytucje, przedsiębiorstwa, jak i osoby fizyczne. Mają bardzo szerokie zastosowania, ograniczone jedynie ludzkimi potrzebami i wyobraźnią.

Jednym z najpotężniejszych używanych narzędzi są szeroko rozumiane analizy przestrzenne. Ich odpowiedni wybór i zastosowanie pozwalają przetworzyć posiadane dane w celu stworzenia zbiorczych opracowań, statystyk i raportów, kontroli i monitoringu oraz zasymulować scenariusze i warianty decyzji. Pozwalają oszacować skutki decyzji i wpływ różnych czynników na elementy rzeczywistości.

2.2.2. Analizy przestrzenne

Analizą przestrzenną jest „zastosowanie określonych operacji analitycznych na danych przestrzennych różnej postaci odniesionych do obiektów jednej lub kilku warstw informacyjnych systemu informacji przestrzennej z wykorzystaniem statystycznych i matematycznych technik dla pozyskania nowej informacji (informacji o charakterze jakościowym, ilościowym)" (Chmiel, 2010[5]).

Przykładowe analizy przestrzenne:

- Analiza obciążania sieci (ang. *network loading*) — modelowanie np. płynności ruchu
- Analiza optymalizacja poruszania się po sieci (ang. *route optimization*) – poszukiwanie optymalnego (najkrótszego np. w funkcji czasu, odległości) połączenia
- Analiza lokalizacji, czyli znalezienie najbliżej położonego obiektu (np. banku, sklepu, szpitalu) od wskazanego miejsca
- Analiza alokacji znajdująca wszystkie punkty oddalone (w czasie lub przestrzeni) od punktu początkowego o zadaną wartość
- Analiza trasowania wyznaczająca optymalną trasę przez n punktów
- Analizy zmian czasowych, np. zmian w użytkowaniu terenu

2.3. Opisywanie kształtu Ziemi

Ludzie od dawna próbują opisać i usystematyzować otaczającą ich rzeczywistość, w tym opisać kształt i wielkość Ziemi oraz położenie na niej różnych obiektów. Zagadnieniami tymi zajmuje się nauka zwana geodezją. Problem ten jest teżnieodłącznie związany z naszą pracą. Niezależnie od tego, czy jedynie archiwizujemy dane, czy też je udostępniamy i przetwarzamy,

aby były użyteczne, ich parametry przestrzenne muszą być odwzorowaniem rzeczywistości, zgodnym z powszechnymi standardami i obowiązującymi przepisami.

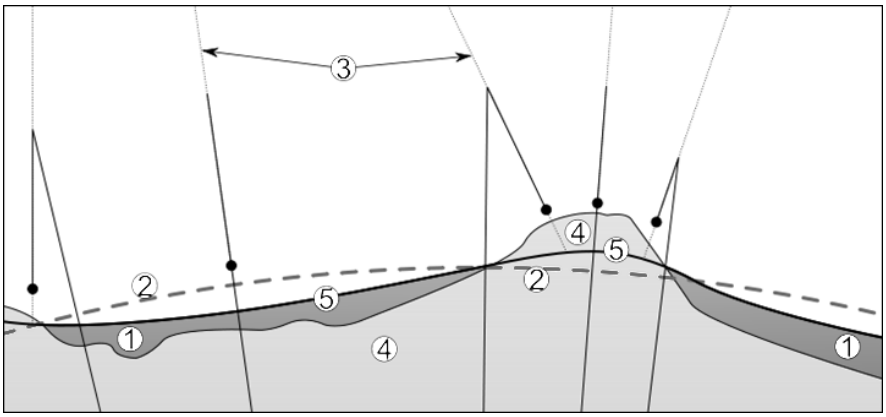
W skrócie na kwestię opisu położenia punktu składa się wybranie bryły przybliżającej kształt Ziemi (tzw. powierzchni odniesienia) i zdefiniowanie układu współrzędnych; ponadto pojawia się zagadnienie odwzorowania trójwymiarowej bryły na płaską powierzchnię kartki papieru czy ekranu monitora podczas wyświetlania (wydruku) danych jako mapy.

Dokładniej problem modelowania Ziemi i znajdujących się na niej obiektów opisany jest w niniejszym rozdziale.

2.3.1. Kształt Ziemi

Hipotezy o kulistym kształcie Ziemi sformułował już w VI w. p.n.e. grecki filozof Pitagoras, a Erastotenes w III w. p.n.e. wyliczył jej promień[2]. Koncepcję kulistego kształtu Ziemi zmodyfikował Newton. Stwierdził on, że ze względu na ruch obrotowy będzie to bryła spłaszczona na biegunach — tzw. elipsoida obrotowa.

Jednakże rzeczywisty kształt Ziemi wcale nie pokrywa się z elipsoidą. Bryłę, która dokładniej przedstawia kształt planety nazwano geoidą. Pokrywa się ona z powierzchnią mórz i oceanów i jest przedłużona w sposób umowny pod powierzchnią lądów. Jest to powierzchnia stałego potencjału siły ciężkości. Geoida jest bryłą bardzo nieregularną i opisanie jej matematycznym wzorem, nawet jeśli możliwe, byłoby całkowicie nieprzydatne w praktyce. Z tego względu w obliczeniach wykorzystuje się elipsoidę obrotową jako najbardziej zbliżoną do geoidy matematyczną powierzchnią odniesienia.



Rysunek 2.2: Przybliżanie kształtu Ziemi — elipsoida i geoida; Na obrazku zaznaczono: 1. ocean, 2. elipsoida, 3. pion kontynentalny, 4. kontynent, 5. geoida (źródło: [4])

[2]Obwód Ziemi wg obliczeń Erastotenesa wynosił około 40 tys. km (jest to tylko przybliżenie wyliczonej wartości, ponieważ użytą przez Greka jednostką miary był stadion, którego dokładna długość nie jest znana). Według współczesnych pomiarów wartość ta wynosi 40 007, 472 km[1].

2.3.2. Powierzchnia odniesienia

W zależności od skali opracowania i wielkości opisywanego obszaru powierzchnię Ziemi można przybliżać jako elipsoidę, kulę, a nawet płaszczyznę (dla bardzo małych obszarów[3]).

Do opracowań małoskalowych (w skalach 1 : 2 500 000 i mniejszych) jako powierzchnię odniesienia stosuje się kulę. Upraszcza to obliczenia, a niedokładność odwzorowania powierzchni Ziemi jest na tyle mała, że w takich skalach nie wpływa na dokładność rysowanej mapy.

W przypadku opracowań wielkoskalowych, o większej dokładności, wykorzystywane są elipsoidy obrotowe. Dla naszego projektu ważne są dwie elipsoidy, które są wykorzystywane w obsługiwanych przez nas układach: GRS'80 oraz WGS-84.

Elipsoida GRS'80

GRS'80 (ang. *Geodetic Reference System '80*) — Geodezyjny System Odniesienia zdefiniowany w 1980 roku. Jest to powierzchnia odniesienia używana w Polsce zgodnie z Rozporządzeniem Rady Ministrów (2000, 2012)[8][9], wykorzystywana w układach „1992" i „2000".

Elipsoida WGS-84

WGS-84 (ang. *World Geodetic System '84*) — popularny system odniesienia zdefiniowany w 1984 roku, powstał w wyniku drobnych modyfikacji elipsoidy GRS'80[4]. Elipsoida WGS-84 stała się podstawowym układem odniesienia w systemach nawigacji satelitarnej, jest też wykorzystywana przez popularny serwis mapowy Google Maps.

2.4. Układy współrzędnych

Geodezja wykorzystuje różne układy współrzędnych do opisania położenia punktu w przestrzeni. Najczęściej są to układy współrzędnych geograficznych lub geodezyjnych — opisująca położenie na powierzchni kuli lub elipsoidy oraz współrzędne prostokątne — opisujące położenie na płaszczyźnie.

[3] Jak podaje Kosiński (2010) [1] na obszarze koła o promieniu mniejszym niż 15,6 km błędy pomiarów są znikomo małe (do 15,6 mm przy pomiarach liniowych) i można uznać ten obszar za płaszczyznę. Jedynie na pomiary wysokości zakrzywienie Ziemi ma o wiele większy wpływ — można je zaniedbać dla obszaru koła o promieniu ok 300-400 m (błąd mniejszy niż 1 cm).

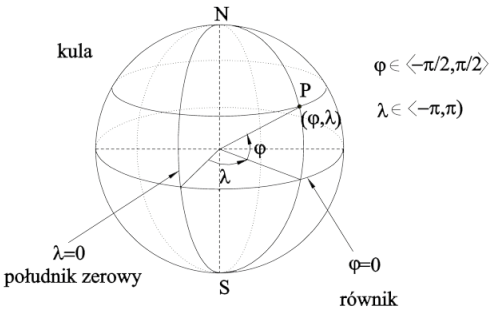
[4] Różnica w długości małej półosi elipsoid wynosi ok 1mm, co jest istotne jedynie podczas bardzo precyzyjnych pomiarach i obliczeniach

2.4.1. Współrzędne geograficzne ϕ i λ

Układ określający położenie punktu P(ϕ, λ) na kuli (Rysunek 2.3).

Szerokość geograficzna (ang. *latitude*) ϕ punktu P - kąt między punktem pionu w punkcie P (prosta łącząca punkt P ze środkiem kuli 0) z płaszczyzną równika. Przyjmuje wartości z zakresu $[0°; 90°]$ na północ od równika, $[0°; -90°]$ na południe.

Długość geograficzna (ang. *longitude*) λ punktu P - kąt dwuścienny między płaszczyzną południka punktu P a płaszczyzną przyjętego południka zerowego. Przyjmuje wartości z zakresu $[0°; 360°]$ lub $[-180°\ ; 180°]$ (dodatnie wartości są na wschód od południka zerowego).



Rysunek 2.3: Układ współrzędnych geograficznych (źródło: [4])

2.4.2. Współrzędne geodezyjne L i B

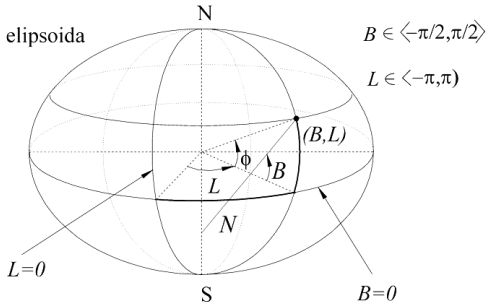
Współrzędne geodezyjne określają położenie punktu na elipsoidzie obrotowej (Rysunek 2.4).

Szerokość geodezyjna (elipsoidalna) B punktu P - kąt między normalną do powierzchni elipsoidy w punkcie P i płaszczyzną równika. Normalna ta nie przechodzi przez środek elipsoidy (z wyjątkiem punktów znajdujących się na równiku i biegunach).

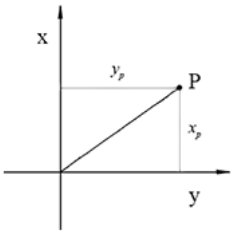
Długość geodezyjna (elipsoidalna) L punktu P - kąt dwuścienny między płaszczyzną południka punktu P a płaszczyzną przyjętego południka zerowego.

2.4.3. Współrzędne prostokątne x i y

Współrzędne prostokątne określają położenie punktu, gdy za powierzchnię odniesienia przyjmuje się płaszczyznę. Oznaczenie osi współrzędnych jest odwrotne, niż w kartezjańskim układzie współrzędnych, mianowicie oś współrzędnych x jest skierowana na północ, a oś współrzędnych y - na zachód (Rysunek 2.5).



Rysunek 2.4: Układ współrzędnych geodezyjnych (źródło: [4])



Rysunek 2.5: Układ współrzędnych prostokątnych (oprac. własne na podstawie [1])

2.5. Odwzorowanie kartograficzne (*map projection*)

Omówione do tej pory pojęcia odnoszą się do modelowania i zapisywania trójwymiarowej rzeczywistości. Jednak w praktyce dane nie są przedstawiane w trójwymiarze, tylko na płaszczyźnie (kartka papieru, ekran komputera). Do transformacji z układu 3D do 2D wykorzystywane są różne odwzorowania.

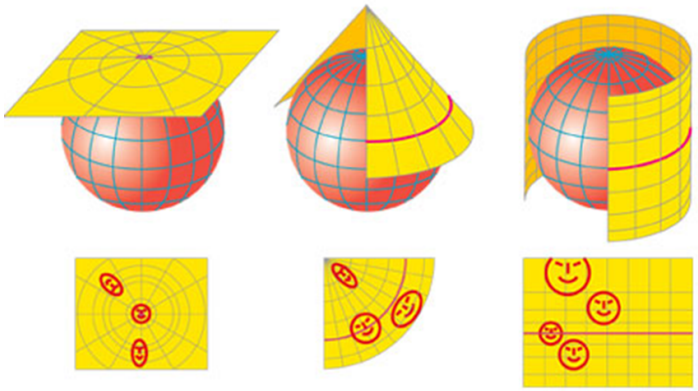
Odwzorowanie jest to określony matematycznie sposób odzwierciedlenia powierzchni odniesienia (elipsoidy, kuli) na płaszczyźnie ([4]).

Rzut kuli lub elipsoidy na powierzchnię zawsze wiąże się ze zniekształceniami odwzorowywanych kątów, odległości, powierzchni. W geodezji wykorzystuje się odwzorowania, które pozostawiają niektóre cechy niezniekształcone.

2.5.1. Rodzaj zniekształceń

Ze względu na rodzaj zniekształceń wyróżnia się grupy odwzorowań[1]:

- **równokątne (konforemne, ang. *conformal*)** — kąty na mapie opracowanej w tym odwzorowaniu są równe odpowiednim kątom w terenie. Stosowane na mapach topograficznych oraz dla marynarki, lotnictwa i wojska.



Rysunek 2.6: Przykłady zniekształceń powstałych w wyniku odwzorowania (źródło: [4])

- **równopolowe** — powierzchnie na mapie odpowiadają (proporcjonalnie do skali) powierzchniom w terenie. Stosowane na mapach gospodarczych.
- **równoodległościowe** — na tych mapach odcinki mierzone wzdłuż pewnych kierunków (np. wzdłuż południka lub równoleżnika) nie są zniekształcane w stosunku do wartości w terenie. Stosowane w niektórych mapach specjalnych, np. łączności radiowej.

2.5.2. Bryła rzutowania

Ze względu na rodzaj powierzchni, na której odtwarza się powierzchnię Ziemi jako oryginału, wyróżnia się odwzorowania płaszczyznowe, walcowe, stożkowe i umowne (wszystkie pozostałe).

Zależnie od położenia brył w stosunku do osi obrotu Ziemi rozróżnia się odwzorowania normalne (biegunowe), poprzeczne (równikowe, ang. *transverse*) i ukośne (horyzontalne).

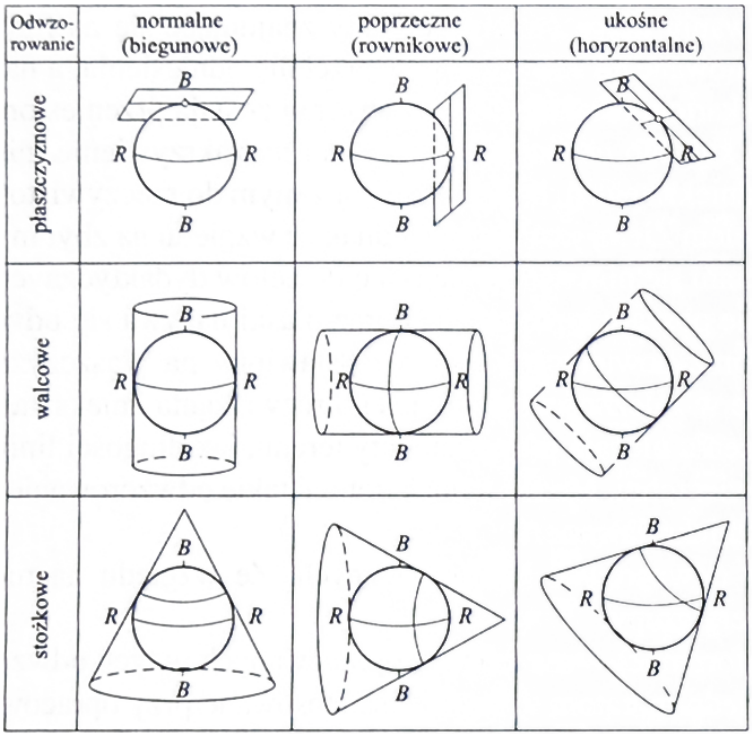
Ponadto powierzchnie mogą być ułożone stycznie do Ziemi , w pewnej odległości nad nią lub siecznie — przecinać bryłę ziemską (Rysunek 2.7).

2.5.3. Odwzorowanie Gaussa-Krügera

Jest to bardzo popularne odwzorowanie, wykorzystywane w Polsce już w czasach międzywojennych. Używano go niemal we wszystkich obowiązujących układach. Również obecnie wykorzystywane jest w obu obowiązujących w naszym kraju układach: „1992” (dla map średnio- i małoskalowych, używany m.in. w naszym krajowym geoportalu[13]) i „2000”.

O odwzorowaniu Gaussa-Krügera tak napisał Kociński (2010)[1]:

> Jest to odwzorowanie równokątne walcowe poprzeczne, w którym południk środkowy obszaru odwzorowuje się wiernie. Ponieważ całej powierzchni elipsoidy ziemskiej



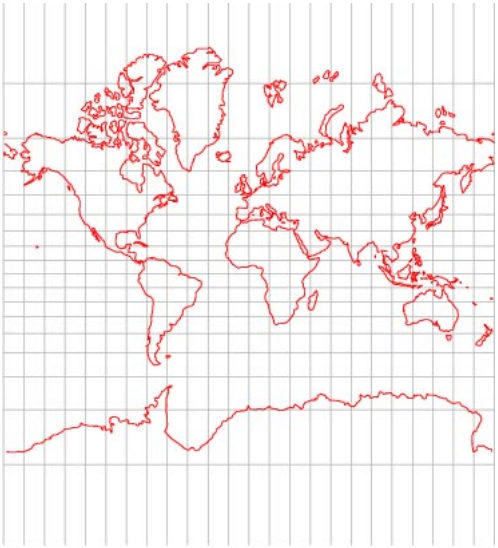
Rysunek 2.7: Odwzorowania kartograficzne (źródło: [2])

nie można odwzorować bez dużych zniekształceń, dlatego obszar Ziemi dzieli się na wąskie pasy południkowe i każdy z takich pasów odwzorowuje się oddzielnie, w oddzielnym układzie współrzędnych prostokątnych płaskich. Szerokość pasów ustalono tak, aby każdy z nich można było rozwinąć na płaszczyźnie i przedstawić na mapie bez praktycznie odczuwalnych zniekształceń, przekraczających stopień dokładności mapy. W tym celu powierzchnię elipsoidy dzieli się na 60 pasów południkowych po 6° każdy lub 120 pasów po 3° każdy.

2.5.4. Odwzorowanie Mercatora

Jest to odwzorowanie walcowe równokątne, opracowane w XVI w. przez flamandzkiego kartografa Gerharda Merkatora, używane głównie w mapach lotniczych i morskich.[22] Obecnie wykorzystywane w serwisach mapowych takich jak Google Maps, Bing Maps i OpenStreetMap[16]. Jest ono dokładne na równiku, ale im bliżej biegunów tym większe pojawiają się deformacje. Sam biegun nie może być odwzorowany, gdyż jego współrzędne odpowiadające szerokości geograficznej zbiegają do nieskończoności. Z tego względu wspomniane serwisy wyświetlają mapy Ziemi do około 85° szerokości geograficznej.

Zalety tego odwzorowania wymienił Bing Maps[25]: zachowuje kształt relatywnie małych obiektów (istotne przy wyświetlaniu zdjęć lotniczych, by nie zniekształcić kwadratowych budynków na prostokątne) oraz zapewnia zachowanie kierunków (północ zawsze jest na górze mapy).



Rysunek 2.8: Ziemia w odwzorowaniu Mercatora (źródło: [23])

2.6. Systemy odniesień przestrzennych

Kociński (2010)[1] definiuje, że „na system odniesień przestrzennych składa się elipsoida obrotowa jako naturalne odniesienie wszystkich punktów powierzchni Ziemi oraz sposób odwzorowywania powierzchni tej elipsoidy na płaszczyźnie".

W Polsce obowiązuje **państwowy system odniesień przestrzennych** określony w Rozporządzeniu Rady Ministrów z dnia 8 sierpnia 2000 r. w sprawie państwowego systemu odniesień przestrzennych (Dz.U. 2000 nr 70 poz. 821)[8]. Według [3] system ten obejmuje między innymi:

- geodezyjny układ odniesienia, stosujący elipsoidę GRS'80
- układ współrzędnych płaskich prostokątnych „2000" — stosowany do mapy zasadniczej
- układ współrzędnych płaskich prostokątnych „1992" — dla map urzędowych o skali mapy 1:10 000 i mniejszej

2.6.1. Kody EPSG

EPSG Geodetic Parameter Set[5] — baza danych zawierająca m.in. parametry elipsoid używanych jako powierzchnie odniesienia oraz układów odniesienia; oznaczane są one kodem liczbowym unikalnym dla typu zasobu. Kody EPSG są powszechnie wykorzystywane do identyfikacji układów współrzędnych w oprogramowaniu przestrzennym, między innymi w naszym serwerze.

2.6.2. Układ „1992”

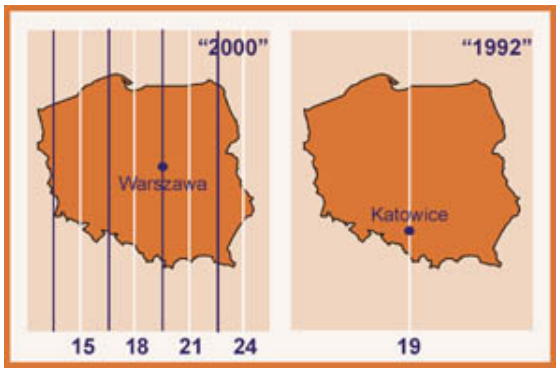
Kod EPSG projekcji: 2180

Jeden z obowiązujących w Polsce układów współrzędnych. Opiera się na odwzorowaniu Gaussa-Krügera elipsoidy WGS-84 w jednym pasie o szerokości 10°. Jest to jednolity układ współrzędnych całego kraju. Występują w nim duże zniekształcenia długościowe (od -70 cm/km do +90 cm/km), przez co układ nie może być wykorzystywany dla map w dużej skali. Używany do map topograficznych w skali 1:10 000 i mniejszych.

2.6.3. Układ „2000”

Kody EPSG projekcji: 2176, 2177, 2178, 2179 (dla każdego z pasów)

Drugi z obowiązujących w Polsce układów współrzędnych. Tak samo jak układ „1992” wykorzystuje odwzorowanie Gaussa-Krügera elipsoidy WGS-84. W tym układzie obszar kraju podzielony jest na cztery pasy południkowe o szerokości 3° długości geograficznej każdy, oznaczone numerami 5, 6, 7 i 8. Zniekształcenia długościowe wynoszą od -7,7 cm/km do +7 cm/km.



Rysunek 2.9: Układy współrzędnych płaskich państwowego systemu odniesień przestrzennych (źródło: [3])

[5]Nazwa pochodzi European Petroleum Survey Group [EPSG] — organizacji naukowej istniejącej w latach 1986–2005, która opracowała i rozpowszechniła tę bazę danych.

2.6.4. Google Web Map

Kod EPSG projekcji: 3857

Układ wykorzystywany przez Google Maps. Wykorzystuje elipsoidę WGS-84 do bezpośredniego zapisywania współrzędnych punktów i obiektów. W momencie rysowania mapy wykorzystywane jest odwzorowanie Mercatora.

3 Serwisy mapowe i aplikacje

Napisany przez nas WUTGeoserver nie jest programem, który może istnieć tylko dla siebie. Podstawowym założeniem jest to, by znalazł miejsce wśród istniejących na rynku rozwiązań. Oznacza to, że nie wystarczy, by spełniał abstrakcyjne założenia — ważne jest, aby poradził sobie w rzeczywistym środowisku. Sprawdziliśmy kompatybilność naszego serwera z kilkoma serwisami i aplikacjami opisanymi poniżej.

3.1. Istniejące serwisy mapowe

3.1.1. Google Maps

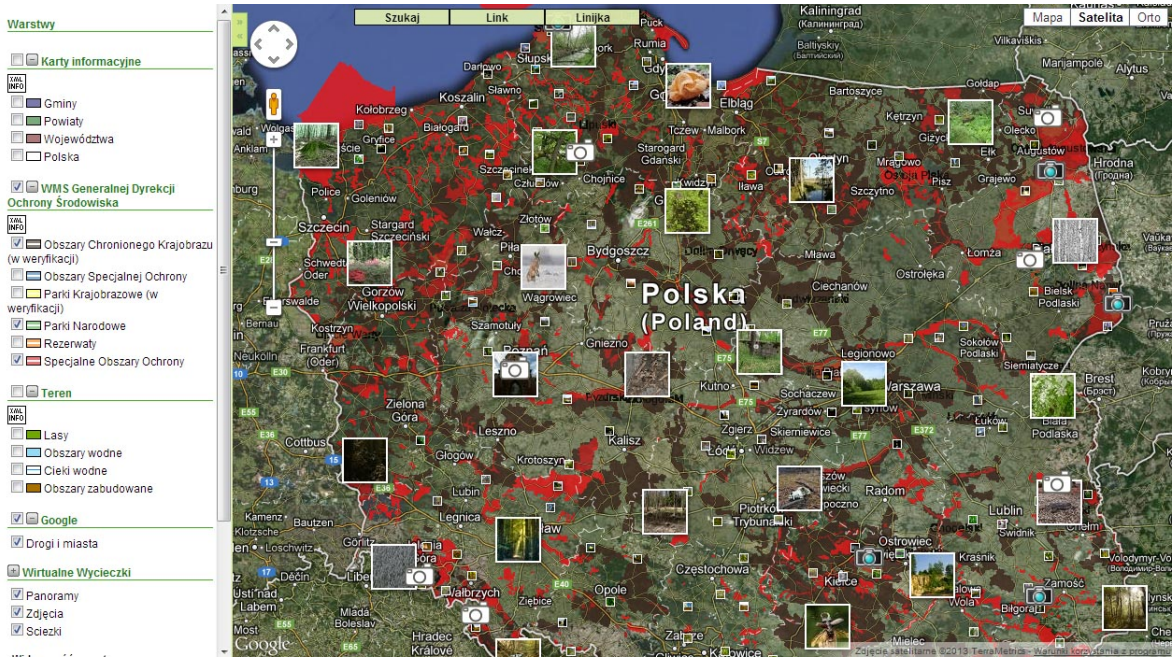
Internetowy serwis mapowy firmy Google działający od 2005 roku. Umożliwia oglądanie map i zdjęć lotniczych i satelitarnych, wyszukiwanie obiektów i coraz więcej pokrewnych funkcji, takich jak widok pod kątem 45° czy street view (panoramy 360°).

Do Google Maps udostępnione jest API (aktualnie rozwijana jest wersja 3), które umożliwia w łatwy i szybki sposób umieszczenie mapy na swojej stronie oraz dostosowanie jej do potrzeb użytkownika. Interfejs ten daje ogromne możliwości, czego przykładem może być EkoMapa[20], w której na udostępniane przez amerykańską firmę zdjęcia lotnicze i satelitarne nałożone zostały mapy udostępniane przez Generalną Dyrekcję Ochrony Środowiska, oraz własne warstwy (karty informacyjne Ekoagregatu) i znaczniki (zdjęcia z serwisu Panoramio, panoramy i ścieżki). Dostęp do API odbywa się za pomocą JavaScript.

Bardzo istotne z punktu widzenia użytkowników jest możliwość nieodpłatnego korzystania zarówno z map jak i API dla zastosowań niekomercyjnych, przy pewnych ograniczeniach dotyczących dziennej liczby zapytań.

Jednym z elementów naszej pracy było napisanie biblioteki JavaScript rozszerzającej Google Maps API o funkcje umożliwiające wygodne dołączenie do dostarczonego produktu map udostępnionych w standardzie WMS (opisany w 4.2.1) z zewnętrznych serwisów.

Biblioteka ta następnie została wykorzystana w naszym serwerze w tzw. aplikacjach użytkownika, które umożliwiają szybkie stworzenie prostej kompozycji map udostępnianych przez różne serwisy WMS. Podkładem tych kompozycji oraz silnikiem są właśnie Google Maps oraz ich API w wersji 3.



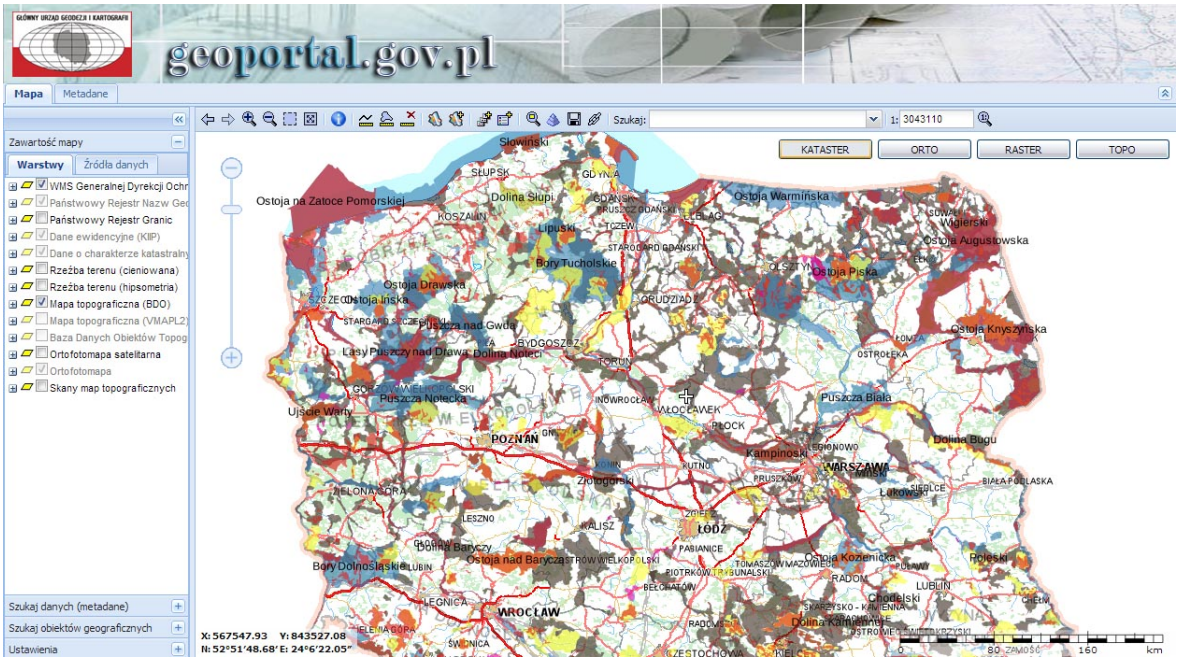
Rysunek 3.1: EkoMapa — przykład wykorzystania serwisu Google Maps (źródło: [20])

3.1.2. Geoportal.gov.pl

Jedną z wytycznych unijnej Dyrektywy INSPIRE[11] było stworzenie przez każde państwo członkowskie geoportalu — portalu internetowego, który zapewnia dostęp do usług danych przestrzennych. W Polsce jego rolę pełni GEOPORTAL.GOV.PL[13] (Geoportal). Geoportal udostępnia dane przestrzenne, między innymi granice województw, powiatów i gmin Polski z Państwowego Rejestru Granic[1].

Z punktu widzenia naszej pracy bardzo istotny jest klient mapowy geoportalu. Umożliwia on wyświetlenie kompozycji map w układzie „1992", za których tło mogą służyć m.in. zdjęcia satelitarne i ortofotomapa oraz mapy topograficzne. Na geoportalu znaleźć można dane udostępniane przez sam geoportal, a także inne podmioty administracji. Istnieje też możliwość podania klientowi dowolnego adresu serwera WMS i nałożenia na mapę kolejnej warstwy. To był właśnie kolejny test naszej aplikacji — sprawdzenie, czy udostępniamy dane zgodnie ze standardami i czy jesteśmy w stanie porozumieć się z serwisem mapowym, który standardy te również implementuje.

[1]Uwaga: dane przestrzenne podlegają ochronie prawnej. Przed wykorzystaniem zasobów udostępnianych przez Geoportal lub inny serwer należy zapoznać się z jego regulaminem.



Rysunek 3.2: Klient mapowy Geoportal.gov.pl (źródło: [13])

3.2. Aplikacje

3.2.1. Quantum GIS

Quantum GIS (QGIS) jest oprogramowaniem geoinformacyjnym, tzn. służącym do wprowadzania, gromadzenia, przetwarzania oraz wizualizacji danych geograficznych, udostępnionym na licencji GNU GPL. Wśród jego możliwości znajduje się między innymi wczytywanie danych z serwerów WMS i WFS. Analogicznie jak w przypadku geoportalu próby podłączenia WUTGeoservera do QGIS były testem zgodności i poprawności implementacji standardów.

3.2.2. Google Earth

Aplikacja Google umożliwia przeglądanie danych przestrzennych (w tym trójwymiarowych) na modelu kuli ziemskiej. Umożliwia wyszukiwanie, nawigację, posiada bazę modeli budynków i najbardziej znanych budowli świata. To co nas interesowało to możliwość szybkiego i wygodnego tworzenia danych przestrzennych w formacie KML obsługiwanym przez nasz serwer. Korzystaliśmy z tej aplikacji, aby stworzyć przykładowe dane do zaimportowania, a także aby sprawdzić, czy dane udostępnione przez nas jako KML są poprawnie rozpoznawane przez aplikację Google.

4 Standardy OGC

Open Geospatial Consortium [OGC] jest międzynarodową organizacją założoną w 1994 roku, zrzeszającą ponad 470 firm, agencji rządowych i uczelni wyższych, które współpracują nad rozwijaniem i implementacją otwartych standardów dla danych i usług przestrzennych. Standardem OGC jes dokument uznany za pomocą kompromisu i zaakceptowany przez członków OGC, zawierający zasady i wskazówki w celu osiągnięcia optymalnej zgodności i interoperacyjności w danym zagadnieniu.

Istotą tych standardów jest ich status — tworzone wspólnie przez ekspertów z różnych środowisk, są popularne i powszechnie stosowane, więc dostosowanie się do nich zapewnia zgodność z dużą częścią istniejących rozwiązań geoinformacyjnych. W niniejszym rozdziale przedstawiliśmy zaimplementowane przez nas standardy OGC: formaty GML i KML oraz WMS i WFS specyfikujące usługi udostępniania danych przestrzennych za pomocą protokołu HTTP. Trzy z tych standardów — GML, WMS i WFS — zostały przyjęte jako standardy International Organization for Standardization (ISO).

4.1. Formaty danych przestrzennych

Jednym z podziałów danych przestrzennych jest podział na dane wektorowe i rastrowe. Geometria danych **wektorowych** jest opisana przez zbiory punktów o znanych współrzędnych, natomiast dane w formacie **rastrowym** to uporządkowany zbiór komórek (pikseli) pokrywający pewną część powierzchni[3]. WUTGeoserver pozwala na import danych wektorowych w formatach GML i KML.

4.1.1. Geography Markup Language (GML)

Oparty na XML format zapisu danych przestrzennych opracowany przez Open Geospatial Consortium. Pliki zapisywane są z rozszerzeniem .gml lub .xml. Pierwsza wersja została opublikowana w 2000 roku. GML jest formatem wymiany danych pomiędzy różnymi aplikacjami systemów informacji geograficznej. Jest to rozbudowany standard, obejmujący dane wektorowe, rastrowe i inne.

Nasza aplikacja obsługuje standard GML w wersji 3.

4.1.2. Keyhole Markup Language (KML)

Jest to format zapisu przestrzennych danych wektorowych, oparty na rozszerzalnym języku znaczników XML. Pliki zapisywane są z rozszerzeniem .kml i .kmz. Format stworzony został przez firmę Keyhole Inc. (stąd nazwa języka), którą w 2004 roku przejął Google. W 2008 roku został oficjalnie przyjęty jako jeden ze standardów OGC. Wykorzystywany jest przede wszystkim w Google Maps i Google Earth.

Do opisu geometrii używa układu współrzędnych geograficznych WGS-84.

W odróżnieniu od GML zawiera w sobie również opis wyglądu (sposobu wyświetlania) obiektów — jego przeznaczeniem jest przede wszystkim wizualizacja obiektów.

4.2. Usługi udostępniania danych przestrzennych

Najważniejszą funkcjonalnością serwera jest — obok przechowywania danych — ich udostępnianie. Wśród standardów OGC dwie najbardziej rozpowszechnione usługi udostępniania danych to Web Map Service i Web Feature Service. Obie udostępniają dane przechowywane jako wektorowe, tylko pierwsza z nich dostarcza wygenerowany wynikowy obraz rastrowy mapy, a druga faktycznie pliki w formacie wektorowym.

4.2.1. Web Map Service (WMS)

Stworzony przez Open Geospatial Consortium (OGC) standard udostępniania map w postaci rastrowej za pomocą interfejsu HTTP. Parametry zapytań przesyłane są w postaci par klucz-wartość.

WMS wykorzystuje dane przestrzenne, aby na podstawie podanych parametrów wygenerować obraz mapy. Udostępniane są jedynie wynikowe obrazy, a nie oryginalne dane.

Standard definiuje trzy rodzaje zapytań: GetCapabilities, GetMap i dodatkowe GetFeatureInfo, z których pierwsze dwa muszą być obowiązkowo obsługiwane.

Zapytanie **GetCapabilities** służy do uzgodnienia wersji standardu obsługiwanego przez serwer i klienta oraz jest źródłem metadanych o usłudze. Domyślnie w formacie XML, zawiera informacje o dostawcy danych, usłudze oraz samych danych (lista dostępnych warstw i ich parametry, niezbędne do poprawnego wygenerowania kolejnych zapytań).

Zapytanie **GetMap** jest prośbą o fragment mapy o podanych parametrach. Nazwy i wartości różnią się w drobnych, ale bardzo istotnych szczegółach zależnie od podanej wersji. Różnice między nimi omówiliśmy dalej. WUTGeoserver implementuje wersję 1.3.0, więc do

niej głównie się odnieśliśmy. Obowiązkowo należy podać m.in. rodzaj usługi (SERVICE), zapytanie (REQUEST), wersję (VERSION), format wynikowy obrazka (FORMAT) i jego wielkość w pikselach (WIDTH, HEIGHT), a także:

- LAYERS — rozdzielona przecinkami lista nazw warstw udostępnianych przez WMS
- STYLES — rozdzielona przecinkami lista nazw styli, jakie należy zastosować do odpowiadających im w liście LAYERS warstw. Więcej o stylach i naszym rozszerzeniu tego parametru w rozdziale 9.5
- CRS — układ współrzędnych geograficznych lub prostokątnych, np. EPSG:2180
- BBOX — prostokąt ograniczający, czyli współrzędne ograniczające rysunek mapy, podane w układzie współrzędnych zdefiniowanym w parametrze crs

Opcjonalne zapytanie **GetFeatureInfo** poozwala uzyskać bardziej szczegółowe informacje o obiekcie znajdującym się w danym punkcie mapy.

Wersja 1.1.1 a 1.3.0

Między wersją 1.1.1 a 1.3.0 WMS nastąpiła pozornie drobna, ale poważna w skutkach zmiana — mianowicie zmieniono znaczenie parametru BBOX (*bounding box*). Wartości tego parametru podaje się w postaci czterech liczb oddzielonych przecinkami, w kolejności: min x, min y, max x, max y. Liczby te są współrzędnymi w układzie współrzędnych podanym w parametrze CRS. Różnica pojawia się w orientacji osi X i Y.

W wersjach do włącznie 1.1.1 zakładano, że oś współrzędnych X rośnie w kierunku wschodnim, a oś współrzędnych Y w kierunku północnym. Niezależnie od odwzorowania i przyjętego układu współrzędnych, kolejność parametrów wyglądała tak samo: zachodnie ograniczenie, południowe, wschodnie, północne.

W wersji 1.3.0 zmieniono podejście. Orientacja osi współrzędnych nie jest już narzucona przez OGC, a zależy od użytego odwzorowania (parametru CRS). Oś X odpowiada pierwszej współrzędnej, a Y drugiej. Oznacza to, że np. w przypadku układu wykorzystywanego przez Google (EPSG:3857) kolejność nadal jest taka sama (zachód, południe, wschód, północ). Jednak układy „1992”, „2000” i WGS-84 orientują pierwszą oś współrzędnych w kierunku północnym, więc w ich przypadku kolejność będzie zamieniona: południowe ograniczenie, zachodnie, północne, wschodnie.

4.2.2. Web Feature Service (WFS)

Standard OGC stworzony w celu udostępniania danych w postaci wektorowej. Podobnie jak WMS korzysta z interfejsu HTTP, a zapytania podawane są w postaci par klucz-wartość.

Usługa WFS udostępnia dane w formatach wektorowych, np. nasz aplikacja pozwala na ściągnięcie plików w formatach GML i KML. Prócz danych geometrycznych klient otrzymuje w ten sposób również atrybuty obiektów. Pozwala to na przeprowadzanie znacznie szerszych analiz. Możliwe jest np. wyszukiwanie po atrybutach albo po ułożeniu elementów względem siebie. W ramach standardu WFS możliwe jest zaawansowane filtrowanie i przeprowadzanie operacji na zbiorach danych.

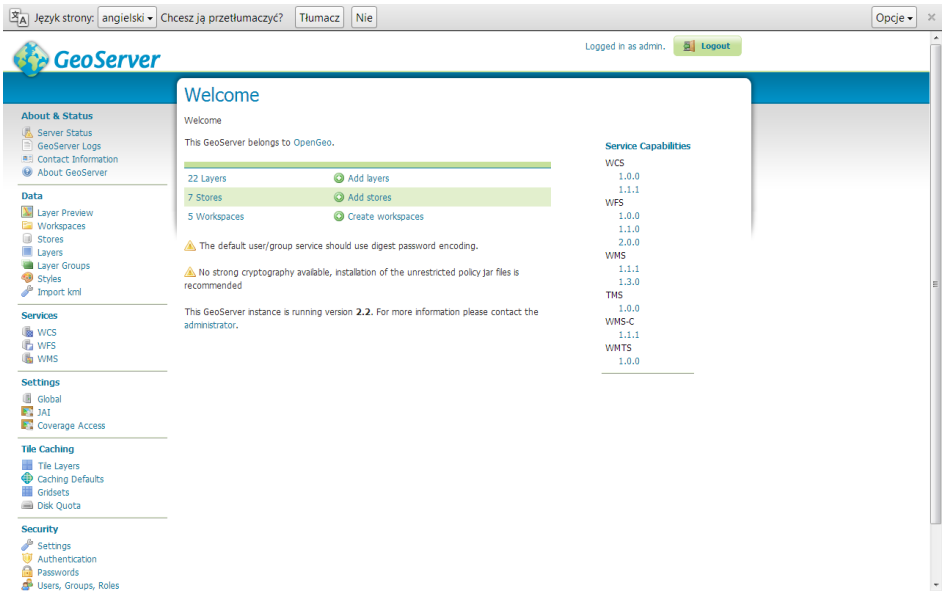
Nasz serwer implementuje WFS w wersji 1.1.0 Więcej informacji znajduje się w rozdziale Instrukcja użytkownika 9.7.

5 Geoserver

W pierwszych tygodniach pracy nad projektem zapoznawaliśmy się z istniejącymi rozwiązaniami z otwartym dostępem do źródeł. Skupiliśmy swoją uwagę na aplikacji Geoserver[26] i bibliotece GeoTools[27]. Raport z analizy tego oprogramowania przedstawiamy w niniejszym rozdziale.

5.1. Raport z analizy kodu źródłowego aplikacji

Do stworzenia serwera mapowego potrzebowaliśmy oprogramowania umożliwiającego przechowywanie danych przestrzennych i udostępnianie ich w postaci usług WMS i WFS. Jedną z popularnych darmowych aplikacją spełniających te kryteria jest Geoserver[26]. Implementuje on standardy OGC takie jak WFS, WCS czy WMS. Posiada bardzo dużo możliwości, jednak wciąż jest niedoskonały. Wśród głównych wad od strony użytkownika końcowego wymienić należy mało intuicyjny i „ciężki" interfejs (Rysunek 5.1), niedziałające opcje (np. eksport warstw do formatu KML, nieprawidłowe przekształcanie projekcji, niedziałający prawidłowo import z zewnętrznych serwisów WFS, nawet gdy są to usługi wystawiane przez Geoserver) oraz trudności we wdrożeniu.



Rysunek 5.1: Interfejs użytkownika aplikacji Geoserver

Rozbudowa Geoservera według naszych wstępnych planów miała polegać na poprawieniu zawartych w nim błędów, rozszerzeniu interfejsu użytkownika i dodaniu nowych funkcjonalności, takich jak: import warstw z plików KML, import danych punktowych z plików CSV i ich interpolacja, rozbudowa cache'a o różne tryby pracy, stworzenie mechanizmu replikacji dla architektury master/slave i kaskadowego importowania danych z zewnętrznych serwisów WFS. Geoserver napisany został w technologii Java i oparty na szkielecie Spring. Do przetwarzania danych przestrzennych wykorzystuje otwartą bibliotekę GeoTools[27].

Wstępna analiza pobranego z repozytorium kodu źródłowego Geoserver'a pokazała jednak wiele dodatkowych trudności z jakimi musielibyśmy się zmierzyć. Pomimo faktu, iż nie było problemów z samą kompilacją kodu i uruchomieniem aplikacji ze źródeł, to w samym kodzie zauważyliśmy sporo błędów. Ponad 100 z istniejących testów jednostkowych od początku kończyło się niepowodzeniem. Bez działających testów jednostkowych traciliśmy kontrolę nad integracją aplikacji z naszymi modułami. Poprawianie tych testów natomiast wymagałoby bardzo dużego nakładu czasu. Kolejne problemy pojawiły się przy próbie napisania pierwszych prostych modułów do Geoservera. Okazała się bowiem, że jego kod źródłowy jest bardzo słabo udokumentowany, korzysta z wielu bibliotek zewnętrznych i sam w sobie jest bardzo obszerny. Podstawowa funkcjonalność — dodawanie nowych warstw i typów warstw z poziomu kodu, do której potrzebowaliśmy łatwego dostępu — wymagała bardzo wielu modyfikacji w samym Geoserverze, jak i w bibliotece GeoTools.

Biorąc pod uwagę ograniczone zasoby jakimi dysponowaliśmy nie mogliśmy sobie pozwolić na tak wiele poprawek, jakich wymagał na tę chwilę Geoserver, bez dostępu do dokumentacji technicznej. Uzgodniliśmy wspólnie z promotorem, że najlepszym rozwiązaniem będzie rezygnacja z pomysłu wykorzystania tej aplikacji i stworzenie całego serwera od podstaw.

6 WUTGeoserver

W tym rozdziale przedstawiamy krótki opis naszego ostatecznego podejścia do założeń tematu pracy inżynierskiej. Rezygnując w całości z wykorzystania darmowych rozwiązań postanowiliśmy napisać własny serwer mapowy, który nazwaliśmy WUTGeoserver. Poniżej przedstawiamy charakterystykę stworzonej aplikacji w kontekście założeń pracy inżynierskiej oraz wykorzystane technologie w celu realizacji tego projektu. Szczegóły implementacyjne, architektura aplikacji oraz dokumentacja techniczna i użytkownika WUTGeoservera zostaną przedstawione w kolejnych rozdziałach.

6.1. Opis funkcjonalności aplikacji i wykorzystane technologie

WUTGeoserver jest aplikacją serwerową świadczącą ulugi udostępniania danych przestrzennych w postaci wektorowej (WFS) i rastrowej (WMS). Umożliwia import i eksport danych w formacie GML i KML, a także import danych punktowych w formacie CSV i przedstawianie ich w postaci zinterpolowanej na mapie. Program współpracuje z popularną aplikacją QuantumGis[29] oraz z oficjalnym polskim serwisem mapowym Geoportal[13]. Ponadto WUTGeoserver umożliwia tworzenie prostych aplikacji do przeglądania map. Podczas tworzenia aplikacji skupiliśmy naszą uwagę na kwestiach wydajnościowych. Stworzyliśmy odpowiednią architekurę serwerową i zastosowaliśmy mechanizmy cache'owania i replikacji.

6.1.1. Podstawowe technologie

Nasza aplikacja wykorzystuje następujące technologie:

- Java SE 1.7 [32]
- J2EE (Java 2 Platform, Enterprise Edition [33]
- Java Servlet 3.0 [34] niewielka aplikacja w technologii Java działająca po stronie serwera WWW w modelu żądanie-odpowiedź, umożliwiająca dynamiczne generowanie odpowiedzi w dowolnym formacie (dokumenty HTML, XML, pliki graficzne, itp.). API serwletów zawiera się w API Javy i ma do niego pełen dostęp. Uruchamiane są w środowisku serwera aplikacji (np. GlassFish, JBoss) albo kontenera webowego (np. Apache Tomcat).

- JSP (Java Server Pages) [35] — technologia umożliwiająca tworzenie dynamicznych dokumentów WWW w formatach HTML, XHTML, DHTML oraz XML z wykorzystaniem języka Java. Możemy patrzeć na JSP jako na obudowę serwletów o wysokim poziomie abstrakcji. W procesie translacji dokument JSP zamieniany jest na odpowiedni serwlet. Strony JSP w połączeniu z technologią JSF stanowią w naszej aplikacji niezależną warstwę widoku w schemacie projektowym MVC (Model-View-Controller).
- JSF 2.0 (Java Server Faces) [36] — bazujący na technologii Java szkielet aplikacji, który upraszcza tworzenie interfejsu użytkownika do aplikacji Java EE. Wykorzystuje JSTL (JavaServer Pages Standard Tag Library) oraz technologię EL (Unified Expression Language) pozwalającą na dostęp do obiektów klas typu Managed Bean z poziomu kodu strony JSP.
- Hibernate 4.0 [37] — mapowanie obiektowo-relacyjne (ORM) dla języka Java pozwalające na realizację warstwy dostępu do danych (ang. *persistance layer*). Zapewnia on przede wszystkim translację danych pomiędzy relacyjną bazą danych a klasami — encjami Javy. Opiera się na wykorzystaniu opisu struktury danych za pomocą języka XML, dzięki czemu można mapować obiekty, bezpośrednio na istniejące tabele bazy danych. Do komunikacji z bazą danych Hibernate udostępnia bazujący na języku SQL (Structured Query Language) język HQL (Hibernate Query Language). Język ten pozwala m.in. na tworzenie zapytań do bazy danych odwołując się do tabel i kolumn jak do odpowiadających im obiektów i pól w aplikacji. Dodatkowo Hibernate zwiększa wydajność operacji na bazie danych dzięki buforowaniu i minimalizacji liczby przesyłanych zapytań.
- Hibernate Spatial [38] — dodatek do technologii Hibernate rozszerzający jej funkcjonalność o wsparcie dla obiektów geometrii JTS (Java Topology Suite) oraz relacji geometrycznych. Integruje się z rozszerzeniem PostGIS [39] dla bazy danych PostgreSQL [40].
- JBOSS Application Server 7 [41] — serwer aplikacyjny

7 Architektura WUTGeoservera

W tym rozdziale przedstawiamy opis i schematy architektury implementowanej aplikacji serwerowej. Przedstawiamy podział aplikacji na warstwy oraz zastosowany wzorzec architektoniczny Master/Slave. Ponadto opisujemy strukturę kodu źródłowego wynikającą z zastosowanych technologii i model aplikacji w kontekście komunikacji z bazą danych.

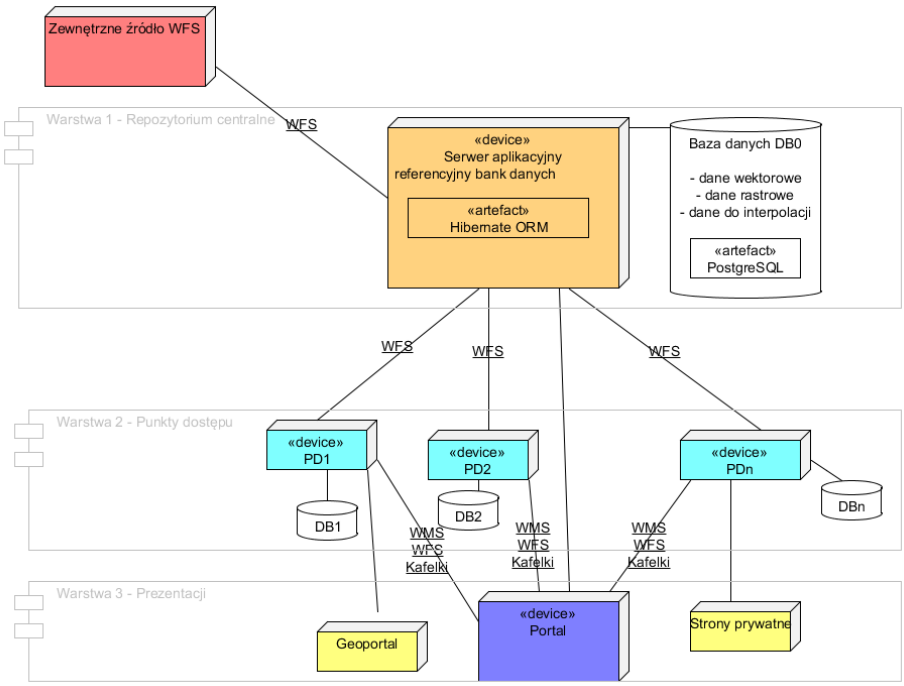
7.1. Architektura serwera

W celu zapewnienia skalowalnego pod względem wydajności rozwiązania serwer ma możliwość skonfigurowania go w postaci węzła centralnego (referencyjny bank danych) i węzłów potomnych (punkty dostępowe). Każdy serwer posiada własną bazę danych zsynchronizowaną z serwerem centralnym poprzez mechanizm replikacji. Serwery powiązane są ze swoimi bazami danych za pośrednictwem mechanizmu mapowania obiektowo – relacyjnego (ORM). Ponadto każdy serwer udostępnia usługi WMS, WFS i kafelki stanowiące komponenty warstwy prezentacji systemu. Klient uzyskuje dostęp do usług poprzez aplikację www, geoportal lub tworząc własne strony. Kompletna architektura aplikacji przedstawiona została na diagramie (Rysunek 7.1).

7.2. Model aplikacji

Głównym założeniem projektu modelu aplikacji WUTGeoserver jest podział struktury aplikacji na warstwę logiki biznesowej i warstwę prezentacji, jednocześnie dążąc do rozwiązania maksymalnie uniwersalnego i zgodnego ze standardami projektowania aplikacji J2EE. Do tego celu aplikacja została oparta na szkielecie JSF, który implementuje wzorzec MVC. Całość logiki biznesowej została zaimplementowana w komponentach modelu. Warstwa widoku podzielona została na moduły w zależności od reprezentowanej usługi. Możemy tak więc wydzielić:

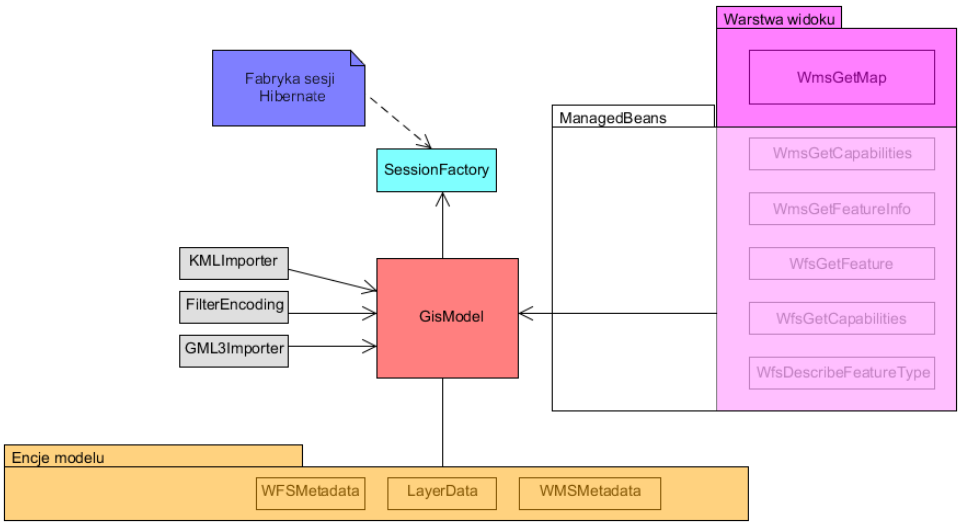
- interfejs graficzny www
- metadane zwracane w formacie XML.
- pliki graficzne zwracane przez serwis WMS.



Rysunek 7.1: Architektura WUTGeoservera

- dane wektorowe zwracane w formacie GML lub KML przez serwis WFS

W celu integracji widoku z modelem wykorzystujemy główne technologie dostarczane przez framework JSF 2.0, tj. mechanizm wstrzykiwania zależności, ManagedBeans oraz Unified Expression Language (EL). Natomiast do mapowania obiektowo-relacyjnego bazy danych na klasy-encje modelu wykorzystujemy bibliotekę Hibernate 4.0 wraz z dodatkiem Hibernate-Spatial. W celu zapewnienia przejrzystości kodu wszystkie metody zapisujące i odczytujące obiekty z bazy danych umieszczone zostały w klasie GisModel będącej ManagedBean o zasięgu aplikacyjnym (Application scope). Klasa ta wraz z klasą pomocniczą SessionFactoryHelper i zestawem bibliotek Hibernate stanowią więc warstwę dostępu do danych. Rozwiązanie takie pozwala na łatwy dostęp do bazy danych w każdym miejscu aplikacji oraz zapewnia, że wszystkie wymagane biblioteki i właściwości zostaną zainicjalizowane przed ich wykorzystaniem. Pozostałe klasy typu ManagedBean wykorzystane w aplikacji to: ServiceExceptionReport mająca zasięg sesji i służąca do zbierania informacji o błędach dotyczących zapytań użytkownika oraz klasy typu Request/Response mające zasięg pojedynczego zapytania i służące do weryfikacji poprawności parametrów zapytań oraz generowania odpowiedzi. Model aplikacji przedstawiony został na diagramie (Rysunek 7.2).



Rysunek 7.2: Model klas aplikacji Geoportal

8 Dokumentacja techniczna

Po doświadczeniu z brakiem dobrej dokumentacji kodu Geoserver'a postanowiliśmy dokładnie szczegółowo strukturę naszego kodu pozostawiając tym samym innym programistom mozliwość dalszej rozbudowy naszej aplikacji. W niniejszym rozdziale przedstawiamy dokładny opis najważniejszych klas aplikacji podzielony na warstwy dostępu do danych, logiki, widoku i modelu. Większość klas przedstawiona została na diagramach. Opisujemy także kluczowe metody pod względem pełnionych przez nie funkcji. Więcej szczegółów, takich jak opisy poszczególnych pól, czy parametrów metod znaleźć można w załączonej na płycie cd dokumentacji Javadoc. W dalszej części rozdziału przedstawiamy techniczne aspekty poszczególnych funkcjonalności.

Wszystkie wymieniane biblioteki są opisane w Rozdziale 10.

8.1. Struktura aplikacji

8.1.1. Warstwa dostępu do danych

wut.GISModel — klasa odpowiedzialna za całą komunikację z bazą. Postanowiliśmy wyraźnie oddzielić komunikację z bazą od pozostałej części aplikacji, dlatego wszystkie wykorzystywane zapytania są obudowane w metody tej klasy. Oprócz tego obsługuje także usuwanie plików z kafelkami, oraz tworzenie kafelków dla serwisu GetTile. Większość funkcji to proste metody pracujące na obiektach z bazy danych. Więcej informacji w załączonych Javadocs. Diagram klasy znajduje się w Załącznikach 14.1. Najważniejsze metody:

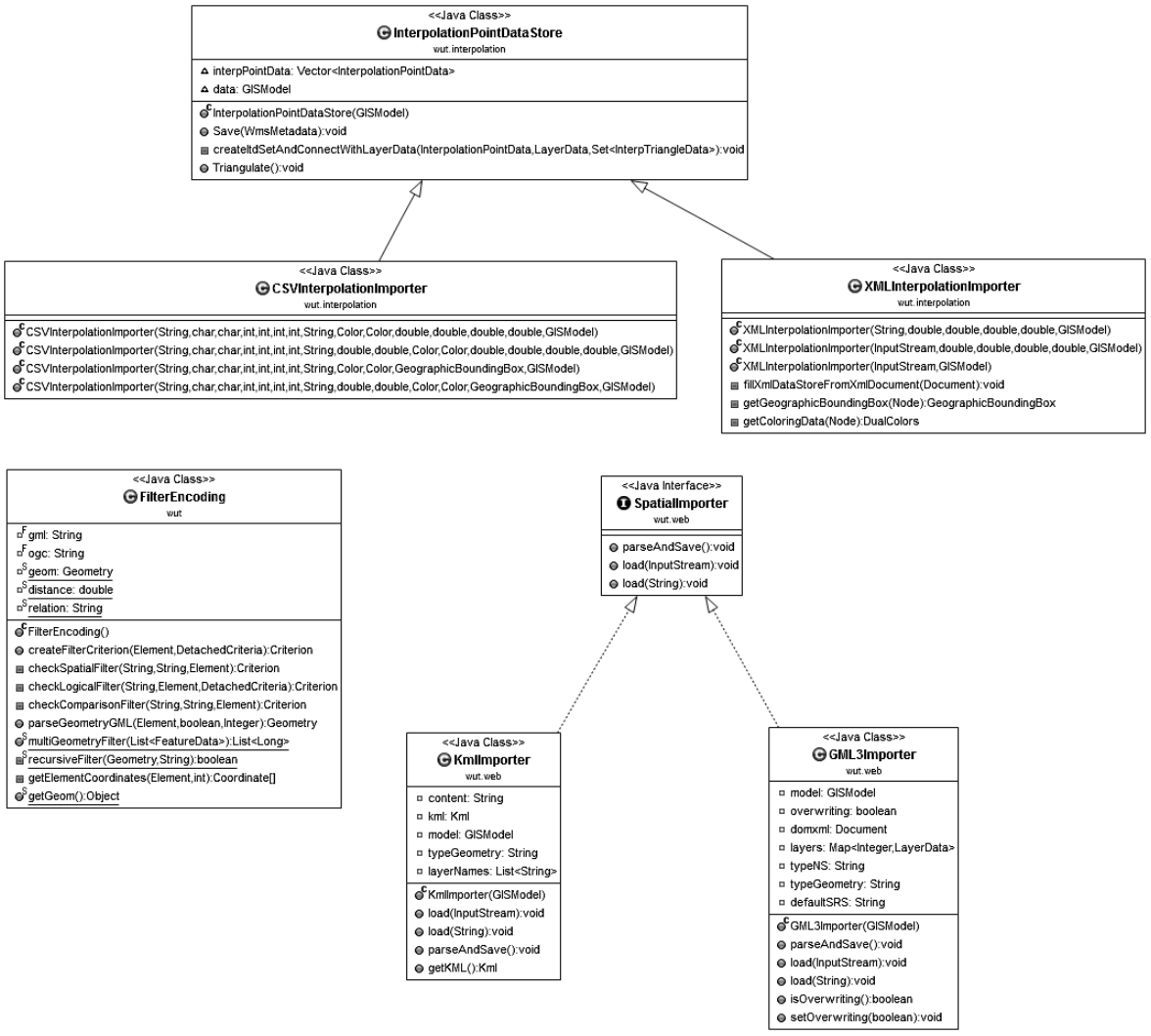
- getBBoxForFeatureType — zwraca BoundigBox dla zadanej warstwy i zadanej projekcji. W przypadku, gdy warstwa nie posiadda zdefiniowanego prostokąta ograniczającego dla tej projekcji to metoda ta wykonuje transformacji istniejącego BoundingBoxa. Uwaga: Operacja ta zwróci prostokąt ograniczający nie mniejszy od BoundingBox'a.
- getFeatureById — obsługuje zapytania z podanym parametrem FEATUREID, tworząc zapytanie do bazy danych o obiekt FeatureData z odpowiednią wartością pola GMLid i ewentualnie wywołując procedurę ST_TRANSFORM Postgisa na zwracanej geometrii w celu transformacji układu współrzędnych

- getFeatureByTypename — obsługuje zapytania z podanym parametrem TYPENAME, tworząc zapytanie o obiekty FeatureData należące do obiektu LayerData z odpowiednią wartością pola wfsFeatureName i ewentualnie wywołując procedurę ST_TRANSFORM Postgisa na zwracanej geometrii w celu transformacji układu współrzędnych.
- getFeatureWithBBoxTransform — obsługuje zapytania z podanym parametrem BOUNDINGBOX, tworząc zapytanie do bazy danych o obiekty FeatureData, których geometria przecina się z zadanym prostokątem. Do tego celu wykonywana jest procedura INTERSECT Postgisa
- getFeatureWithFilter — obsługuje zapytania z podanym parametrem FILTER. Metoda ta, oprócz nazw warstw i nazwy docelowej projekcji, jako parametr przyjmuje obiekt Criteria utworzony za pomocą klasy FilterEncoding w momencie walidacji parametrów zapytania przez WfsGetFeature_Request oraz nazwę atrybutu po którym wykonywane jest filtrowanie. W przypadku filtrów geometrycznych dodatkowo wykonywane jest filtrowanie geometrii typu GeometryCollection wykorzystując metodę multiGeometryFilter klasy FilterEncoding
- getEXFeatureData — pobiera wszystkie geometrie z podanej warstwy przecinające się z BoundingBoxem podanym jako parametr, oraz — jeżeli parametr TRANSFORM jest ustawiony na true — przeprowadza transformację między układami współrzędnych z natywnego danej geometrii na podany w BoundingBox
- getEXInterpTriangleData — analogicznie jak getEXFeatureData, tylko pobiera obiekty wykorzystywane przy obrazach z danymi interpolacyjnymi
- saveLayer — metoda zapisująca warstwę, może zapisywać dane geometryczne warstwy, oraz jej metadane dla serwisów WMS oraz WFS

8.1.2. Kontroler — najważniejsze klasy logiki biznesowej

wut.FilterEncoding — klasa odpowiedzialna za parsowanie filtrów i geometrii GML zarówno przy imporcie plików jak i dla zapytań GetFeature.

- checkComparisonFilter — generuje część zapytania filtrującą rezultat względem wartości atrybutów obiektów
- checkSpatialFilter — generuje część zapytania filtrującą rezultat względem relacji geometrycznej z zadaną figurą
- checkLogicalFilter — umożliwia łączenie filtrów zapytań spójnikami logicznymi AND i OR



Rysunek 8.1: Diagram klas kontrolera interpolacji i importu plików

- multiGeometryFilter, która oddzielnie obsługuje geometrie złożone (typu GeometryCollection), jako że relacje geometryczne i transformacje są dla nich niedozwolone przez PostGIS. Metoda ta porównuje kolekcje geometrii korzystając jedynie z mietod dostarczonych przez JTS i nie tworzy ona żadnych dodatkowych zapytań HQL.
- parseGeometryGML — metoda ta jako parametr przyjmuje korzeń drzewa DOM opisującego geometrię w formacie GML i zwraca obiekt JTS Geometry odpowiadający tej geometrii

wut.web.GML3Importer — klasa do importu danych z plików GML. Wykorzystuje DOM w celu reprezentacji dokumentu XML w pamięci. Do konwertowania geometrii wykorzystuje klasę FilterEncoding.

wut.web.KMLImporter — klasa do importu danych z plików KML. Wykorzystuje bibliotekę JAK (*Java API for KML*) w celu stworzenia reprezentacji dokumentu KML w pamięci.

wut.interpolation.InterpolationPointData — klasa opisująca jedną warstwę zawierającą dane interpolacyjne. Wykorzystywana przy imporcie. Zawiera w sobie dane o projekcji w postaci numeru EPSG, nazwę, wartości minimalne i maksymalne, listę InterpolationColorData (opis gradientu), geograficzny prostokąt ograniczający oraz kolekcję zawierającą punkty z atrybutami. Klasa ta potrafi przeprowadzić triangulację na danych zawartych w tym obiekcie.

wut.interpolation.InterpolationPointDataStore — klasa opisująca zestaw obiektów InterpolationPointData, której zadaniem jest zapis ich do bazy danych. Najważniejsza metoda to Save — która zapisuje wszystkie InterpolationPointData zawarte w tym obiekcie do bazy danych, najpierw przeprowadzając na nich triangulację

wut.interpolation.CSVInterpolationImporter — klasa dziedzicząca po InterpolationPointDataStore, której celem jest wczytywanie plików CSV. Wykorzystuje do tego bibliotekę OpenCSV 10. Zawiera kilka typów konstruktorów, których najważniejszymi parametrami są: ścieżka do pliku, delimiter (symbol rozdzielający pola, domyślnie przecinek), rodzaj cudzysłowiów w pliku, numery indeksów pól z wartością oraz pól z długością i szerokością geograficzną.

wut.interpolation.XMLInterpolationImporter — klasa dziedzicząca po InterpolationPointDataStore, której celem jest wczytywanie plików XML. Wykorzystuje do tego Java DOM API for XML zawarte w bibliotece standardowej języka Java.

wut.replication.ExternalWFSReplication — Encja przechowująca dane serwera zewnętrznego wraz z informacjami o warstwach przechowywanymi w obiektach klasy wut.replication.FeatureTypeInfo oraz wybrane przez użytkownika warstwy do replikacji. Ponadto klasa ta posiada obiekt klasy ReplicationScheduler (Rysunek 8.2). Najważniejsze metody ExternalWFSReplication to:

- parseGetCapabilities — próbuje połączyć się z serwerem zewnętrznym o podanym adresie URL za pomocą biblioteki Apache HttpComponents Client (HttpClient), następnie wysyła zapytanie GetCapabilities i parsuje otrzymany dokument XML odczytując niezbędne informacje o warstwach

- parseGetFeature — wysyła za pomocą biblioteki HttpClient zapytanie GetFeature z odpowiednio ustawionym parametrem typename i pobiera plik GML z opisem warstwy do pamięci, następnie porównuje wersje danych i dokonuje ewentualnej replikacji
- startReplication — tworzy obiekt ReplicationScheduler, zapisuje do bazy danych wszystkie dane niezbędne do odtworzenia swojego stanu po restarcie i przeprowadza pierwszą replikację wywołując metodę performReplication
- performReplication — pobiera z bazy danych informacje o warstwach na podstawie nazw i wywołuje metodę getFeature dla otrzymanej listy FeatureTypeInfo

wut.replication.ReplicationInterceptor — klasa dziedzicząca po hibernate.EmptyInterceptor zbierająca informacje o wprowadzanych zmianach w bazie danych i propagująca te zmiany po wszystkich serwerach z puli slave'ów (Rysunek 14.2).

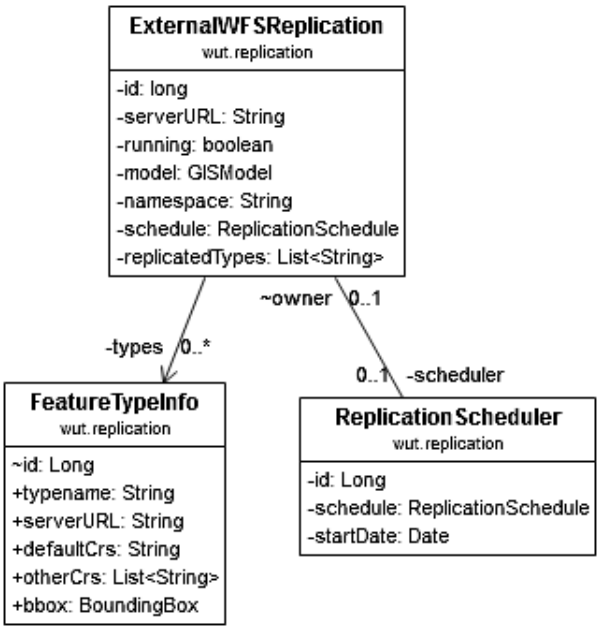
wut.replication.ReplicationListener — Metoda listen tej klasy uruchamiana jest w oddzielnym wątku i oczekuje na połączenie TCP na zadanym porcie w celu dalszego obsłużenia przesyłanych poleceń (Rysunek 14.2).

wut.replication.ReplicationScheduler — implementuje interfejs Runnable, uruchamiana w oddzielnym wątku porównuje co 10 minut czas wykonania ostatniej aktualizacji z czasem bieżącym. W przypadku upłynięcia określonego przez użytkownika okresu zleca aktualizację poprzez wywołanie metody performReplication swojego właściciela — obiektu ExternalWFSReplication (Rysunek 8.2).

wut.replication.ServerManager — klasa odpowiadająca za komunikację między serwerami w architekturze master-slave, w tym za tworzenie puli serwerów potomnych, kopiowanie bazy danych nowo podłączonym do serwera głównego i kontrolę wersji danych. Jest to klasa typu Singleton (Załącznik, rysunek 14.2). Najważniejsze metody tej klasy to:

- addSlave — dodaje nowo podłączony serwer do puli serwerów potomnych i rozpoczyna kopiowanie własnej bazy danych przesyłając do tego serwera plik zrzutu. Ponadto, po zatwierdzeniu przez slave'a otrzymania kopii bazy serwer główny tworzy nową fabrykę sesji (Hibernate SessionFactory) dla połączenia ze zdalną bazą danych. Komunikacja serwerów odbywa się poprzez protokół TCP/IP.
- createSFConfig — tworzy plik konfiguracyjny dla zdalnego połączenia Hibernate'a.

- dump_db — tworzy zrzut lokalnej bazy danych do pliku SQL wywołując narzędzie pg_dump
- init — inicjalizuje obiekt ServerManager na podstawie pliku properties
- join — inicjalizuje połączenie z serwerem głównym i pobranie kopii jego bazy danych
- restore_db — odtwarza bazę danych z pobranego pliku SQL wywołując narzędzie psql



Rysunek 8.2: Diagram klas ExternalWFSReplication, ReplicationScheduler i FeatureTypeInfo z pakietu wut.replication

8.1.3. Klasy widoku

Klasy kontrolujące widok aplikacji, czyli odpowiedzialne za interpretowanie zapytań użytkownika i generowanie odpowiedzi w postaci graficznej lub jako dokumenty XML, zostały podzielone na dwie logiczne części. Pierwszą z nich są klasy powiązane z usługą WMS odpowiedzialne za renderowanie map, drugą natomiast zestaw ManagedBeanów integrujących model aplikacji z plikami JSP odpowiedzialnymi za generowanie dokumentów XML.

Renderer

wut.wms.WMSGetMap oraz **wut.wms.CSGetMap** — serwlety odpowiedzialne odpowiednio za zapytania GetMap oraz GetMap z opisem stylu w zapytaniu. Tworzą one

obiekty klasy WMSGetMap_Request lub CSGetMap_Request, które zostają przekazane do obiektu klasy RenderManager metodami run/runCS. Diagramy klas znajdują się w Załączniku 14.3

wut.wms.GetTile — serwlet odpowiedzialny za zapytania o kafelki, przekazuje interesujące nas parametry do obiektu klasy LayerRenderManager metodami runTile.

wut.wms.RenderManager — klasa odpowiedzialna za zarządzanie rysowaniem odpowiedzi na zapytania GetMap oraz CSGetMap. Dla każdej warstwy, która znajduje się w żądaniu GetMap tworzy obiekt LayerRenderRequest, który jest przekazywany obiektowi klasy LayerRenderManager. W przypadku zapytania CSGetMap dodatkowo przetwarza style zapisane w postaci XML na listę obiektów StyleDefinition i wykorzystuje je do stworzenia LayerRenderRequest. Zawiera metody:

- createStyleDefinitions — metoda tworząca na bazie obiektu String listę definicji styli, w przypadku, gdy nastąpi błąd jest on zwracany poprzez obiekt ServiceExceptionReport
- createCustomStyleDefinition — metoda tworząca na bazie węzła dokumentu XML definicje stylu
- getExistingStyle — metoda pobierająca z bazy danych istniejący styl, na bazie odpowiedniego elementu węzła dokumentu XML
- createLayerRenderRequest — tworzy obiekt klasy LayerRenderRequest na bazie podanych mu parametrów. W tym obiekcie jest umieszczana większość danych potrzebnych do stworzenia odpowiedzi na zapytanie
- run oraz runCS — funkcje odpowiedzialne za rozpoczęcie procesu rysowywania mapy

wut.wms.LayerRenderManager — klasa odpowiedzialna za zarządzanie wyrysowywaniem obrazu danej warstwy, zajmuje się także wyrysowywaniem kafelków oraz obsługą cache'a. W trakcie działania albo pobiera obraz z dysku, albo formułuje RenderRequest do obiektu klasy Renderer z potrzebnym mu wycinkiem obrazu w celu wyrysowania go. Najważniejsze metody:

- runTile — zwraca kafelek z warstwy o podanej nazwie i współrzędnych. W trakcie tego procesu kafelek zostaje wczytany z dysku, lub — w przypadku gdy nie istnieje — wyrysowywany oraz zapisany

- run — zwraca obraz mapy na bazie obiektu LayerRenderRequest. W trakcie tego procesu obraz może zostać wyrysowany lub wczytany z dysku
- cachedRender — zwraca obraz mapy pozyskany z wykorzystaniem cache'u. Sprawdza, które kafelki są potrzebne do narysowania mapy, po czym pozyskuje każdy z tych kafelków i składa w obraz wynikowy
- getTileRender — sprawdza czy odpowiedni kafelek istnieje i próbuje go wczytać z dysku, jeżeli nie istnieje na dysku to zostaje wyrysowany i zapisany na dysku przy użyciu metody createTiledLayerFile
- createTiledLayerFile — pobiera BoundingBox wycinka odpowiadającego temu kafelkowi oraz wykorzystuje metodę createTileFileInner do pozyskania obrazu
- createTileFileInner — tworzy obraz w pamięci, po czym sprawdza, czy można zapisać dane do odpowiedniego pliku. Następnie wykorzystując obiekt klasy Renderer wyrysowuje obraz, który w kolejnym kroku jest zapisywany na dysku (jeżeli to możliwe), a następnie zwracany
- getBBfromXYZ — wylicza prostokąt ograniczający dla kafelka z odpowiednimi współrzędnymi x, y, zoom
- tiledRender — sprawdza czy istnieje na dysku plik z obrazem odpowiedniego wycinka warstwy oraz próbuje go wczytać, w!momencie gdy nie istnieje to zostaje wyrysowany i zapisany na dysku przy użyciu metody createTileFile
- createTileFile — analogiczne do createTiledLayerFile, główna różnica to nazwa tworzonego pliku

wut.wms.Renderer — klasa odpowiedzialna za wyrysowanie danych geometrycznych oraz interpolacyjnych przy użyciu obiektu Graphics2D. Wypełnienie, obrysowanie figur geometrycznych oraz gradient do danych interpolacyjnych otrzymujemy z obiektu klasy DrawableStyle. Najważniejsza metoda: run — funkcja, która wyrysowuje dane zlecenie — obiekt klasy RenderRequest — na obiekcie Graphics2D przekazanym jako parametr

wut.interpolation.drawing.TriangleGradientPaint oraz **wut.interpolation.drawing.TriangleGradientPaintContext** — klasy odpowiedzialne za wypełnianie trójkątów gradientem na bazie wartości interpolowanych liniowo. Najważniejszy jest konstruktor klasy TriangleGradientPaint. Jako parametry przyjmuje:

- Polygon — wielokąt będący trójkątem
- Tablica double — trójelementowa tablica opisująca wartości w wierzchołkach

- Lista InterpolationColorData — lista definiująca gradient

W tym konstruktorze są wyliczane parametry płaszczyzny, która odpowiada interpolacji tych wartości, te parametry są potem używane do wyliczenia wartości dla każdego z tych punktów.

Generator dokumentów XML

Dokumenty XML (poza KML) są dynamicznie generowane za pomocą technologii JSP i JSF. Schemat dokumentu poprzez mechanizm EL jest wypełniany danymi, które znajdują się w odpowiednich klasach — ManagedBean modelu widoku (Rysunek w załączniku 14.4). Do prawidłowego wyświetlania dokumentów GML wykorzystywane są dodatkowo klasy implementujące interfejs FeatureFormat. Odpowiadają one za transformację danych geometrii na odpowiadające im ciągi znaków zgodne z formatem GML 3. Klasa Feature sprzęga dane obiektów (FeatureData) ze sposobem ich wyświetlania (FeatureFormat). Lista obiektów będących wynikiem zapytania GetFeature generowana i przechowywana jest przez ManagedBean WfsGetFeature_Response.

wut.wfs.WfsGetFeature_Request — ManagedBean odpowiadający za pobranie parametrów zapytania (poprzez zastosowanie mechanizmu wstrzyknięć zależności). Najważniejszą metodą tej klasy jest metoda validate dokonująca wstępnego sprawdzenia poprawności parametrów. Dodatkowo klasa ta posiada szereg właściwości typu „set" wstępnie parsujących parametry i zwracających je w wygodnej postaci.

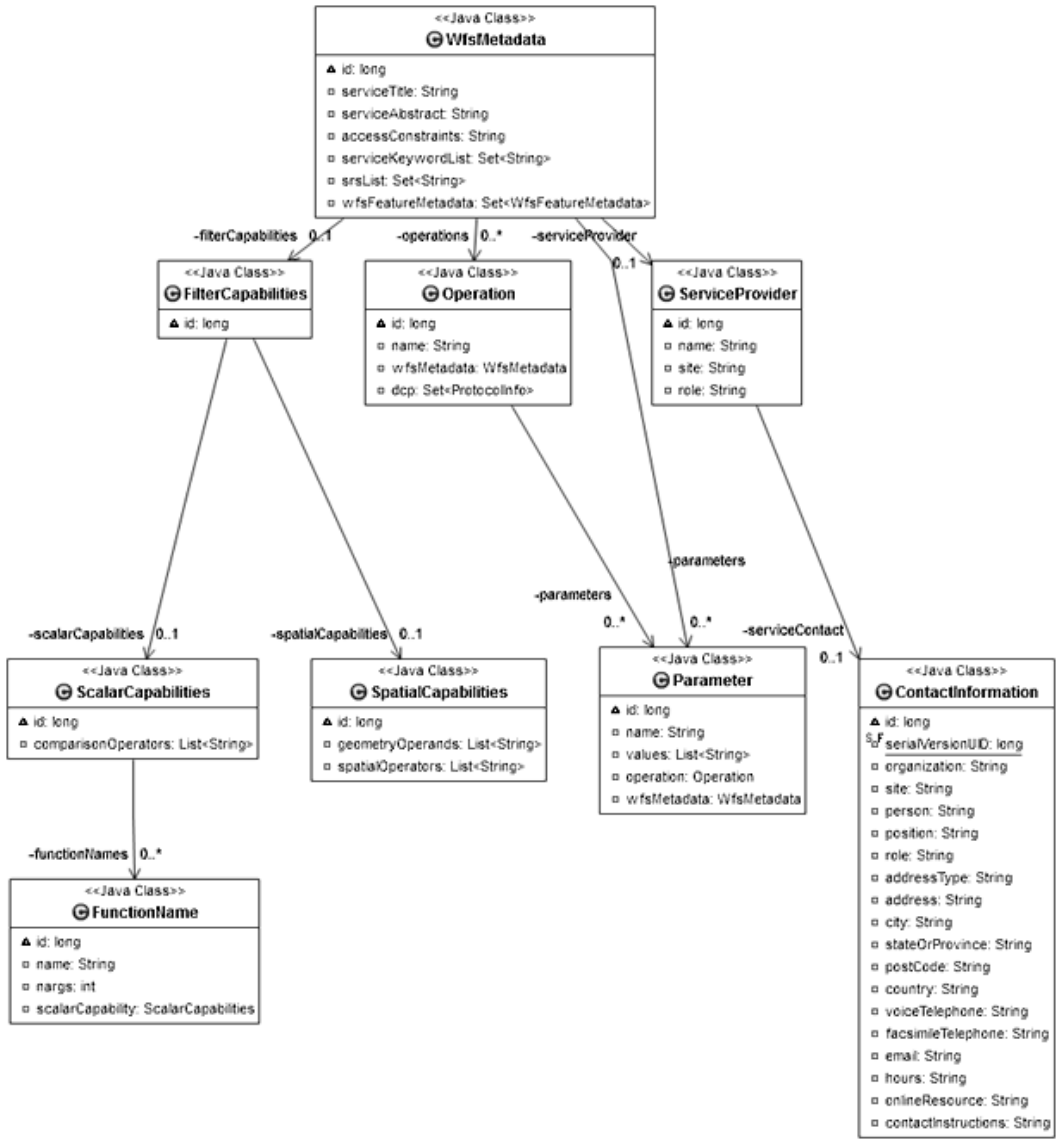
wut.wfs.WfsGetFeature_Response — ManagedBean będący klasą modelu widoku dla dokumentów feature.jsp. Cała logika tej klasy wykonuje się automatycznie w chwili utworzenia jej obiektu, czyli w momencie, gdy w odpowiedzi na zapytanie, w odpowiednim dokumencie JSP pojawia się odwołanie do tego Beana. Dzieje się tak dzięki zastosowaniu mechanizmu wstrzyknięć zależności (w tym wypadku Beanów GisModel, wfsGFReq, serviceException) oraz specjalnej metody init objętej adnotacją @PostConstruct. Metoda ta zaczyna wykonywać się natychmiast po utworzeniu obiektu. Dzięki temu mechanizmowi mamy pewność, że wszystkie pola klasy zostały już zainicjalizowane. W metodzie init analizowane są parametry zapytania w celu pobrania odpowiednich obiektów potrzebnych do odpowiedzi. Następnie wywoływana jest jedna z metod getFeature z GisModelu (8.1.2) , a jej wynik — lista obiektów klasy Feature, zapisywany w polu members. Na koniec, w przypadku takiej konieczności, otrzymana lista jest sortowana. Samym wyświetleniem dokumentu XML zajmuje się serwlet utworzony automatycznie na bazie pliku JSP.

8.1.4. Model danych aplikacji

Do przechowywania danych wykorzystujemy bazę danych PostgreSQL z dodatkiem PostGIS rozszerzającym tą bazę danych o możliwości przestrzenne. Do komunikacji z bazą danych korzystamy z biblioteki do mapowania obiektowo relacyjnego Hibernate z dodatkiem HibernateSpatial rozszerzającym go o możliwości przestrzenne, żeby móc wykorzystać udogodnienia oferowane nam przez rozszerzenie PostGIS.

8.1.5. Opis encji

Serwis WFS

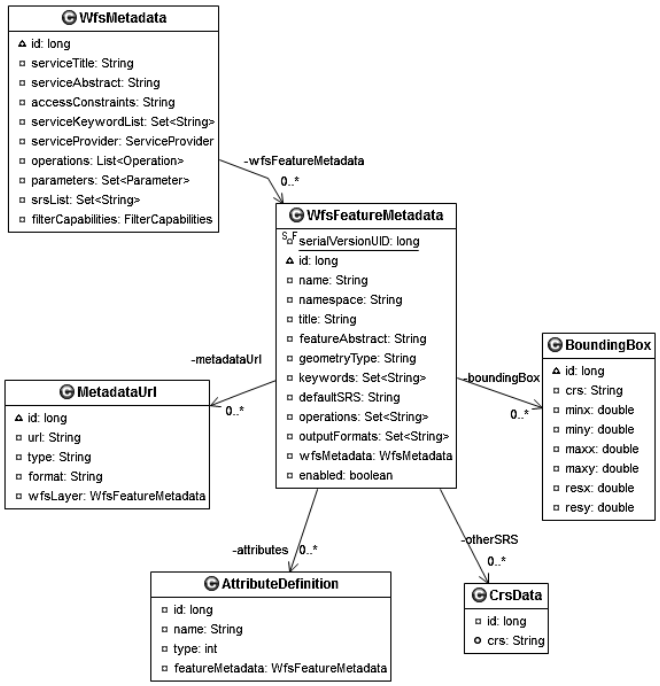


Rysunek 8.3: Diagram asocjacji dla klasy WfsMetdata

WfsMetadata — opis serwisu WFS. Zawiera między innymi:

- operations — obsługiwane zapytania
- filterCapabilities — filtry obsługiwane przez dany serwis
- wfsFeatureMetadata — dostępne warstwy
- serviceProvider — pozostałe informacje o serwisie, w tym informacje kontaktowe dostawcy usługi.

Warstwy WFS

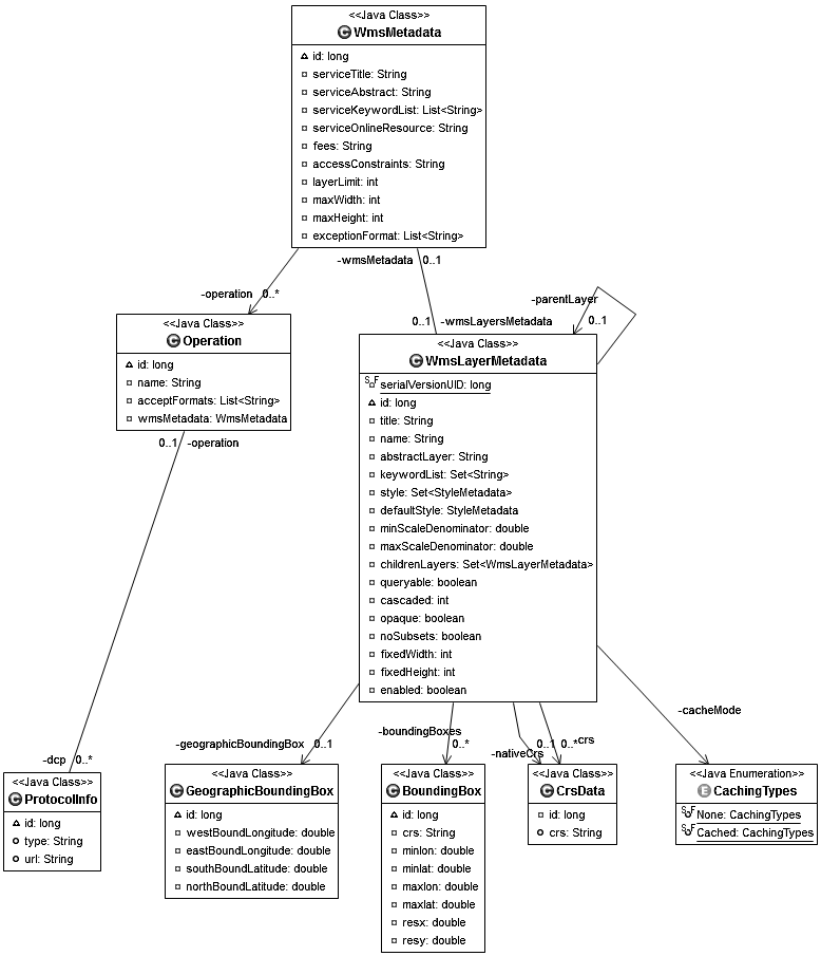


Rysunek 8.4: Diagram asocjacji dla klasy WfsFeatureMetdata

WfsFeatureMetadata — opis warstwy WFS. Zawiera między innymi:

- namespace — przestrzeń nazw danej warstw
- name — nazwa po której możemy się do niej odwoływać
- title — tytuł wyświetlany użytkownikowi
- boundingBox — prostokąty ograniczające
- attributeDefinition — definicje nazwy i typu dla atrybutów obiektów zawartych na danej warstwie
- defaultSRS oraz otherSRS — domyślne oraz wszystkie obsługiwane odwzorowania

Serwis WMS oraz warstwy WMS



Rysunek 8.5: Diagram asocjacji dla danych o serwisie oraz warstwach WMS

WmsMetadata — opis serwisu WMS. Przechowuje warstwy WMS w postaci drzewa, w którym korzeniem jest sztuczny, odgórnie zdefiniowany węzeł. Zawiera między innymi:

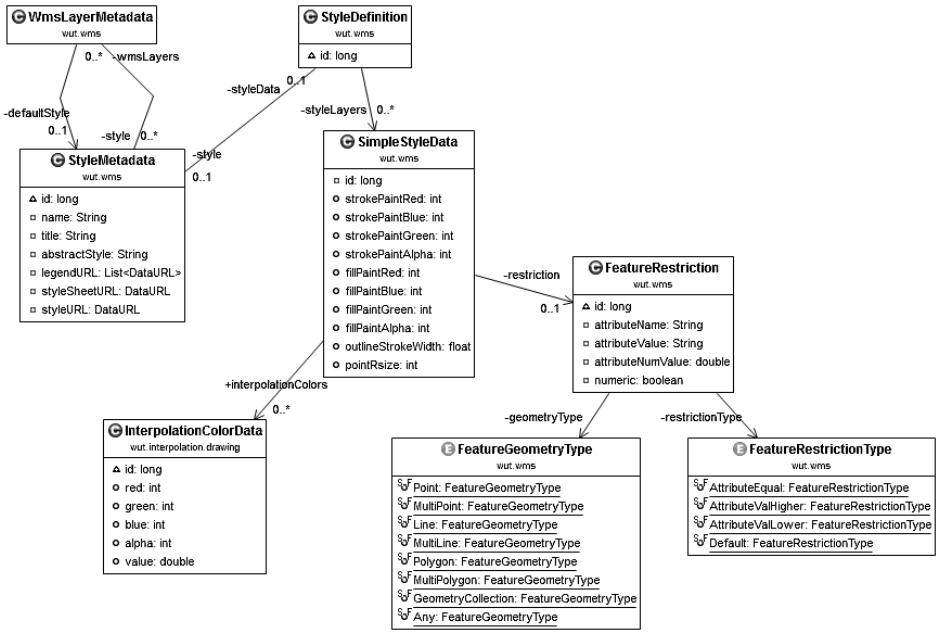
- wmsLayersMetadata — korzeń drzewa zawierającego warstwy
- operation — obsługiwane zapytania

WmsLayerMetadata — opis warstwy WMS. Dodatkowo istnieje jednokierunkowe powiązanie z LayerData poprzez nazwę warstwy WMS oraz pole wmsLayerName w LayerData. Zawiera między innymi:

- name — nazwa po której możemy się do niej odwoływać
- title — tytuł wyświetlany użytkownikowi
- geographicBoundingBox — prostokąt ograniczający we współrzędnych geograficznych
- boundingBoxes — prostokąty ograniczające w pewnych układach odniesień przestrzennych

- nativeCrs — układ odniesień przestrzennych w jakim przechowywane są dane
- crs — układy odniesień przestrzennych w jakich dane mogą być zwrócone, zawiera nativeCrs
- defaultStyle — styl domyślny
- style — style obsługiwane przez tę warstwę, zawiera defaultStyle

Styl warstwy WMS



Rysunek 8.6: Diagram asocjacji dla danych opisujących style warstw WMS

StyleMetadata — opis stylu warstwy WMS. Najważniejsze elementy:

- name — nazwa po której możemy się do niej odwoływać
- title — tytuł wyświetlany użytkownikowi
- styleData — informacje o sposobie wyświetlania

StyleDefinition — kolekcja informacji o sposobie wyświetlania warstw, wymagana do wyrysowania warstwy. Zawiera w sobie listę elementów SimpleStyleData.

SimpleStyleData — informacje o sposobie wyświetlenia obiektów spełniających zadane kryteria. Zawiera:

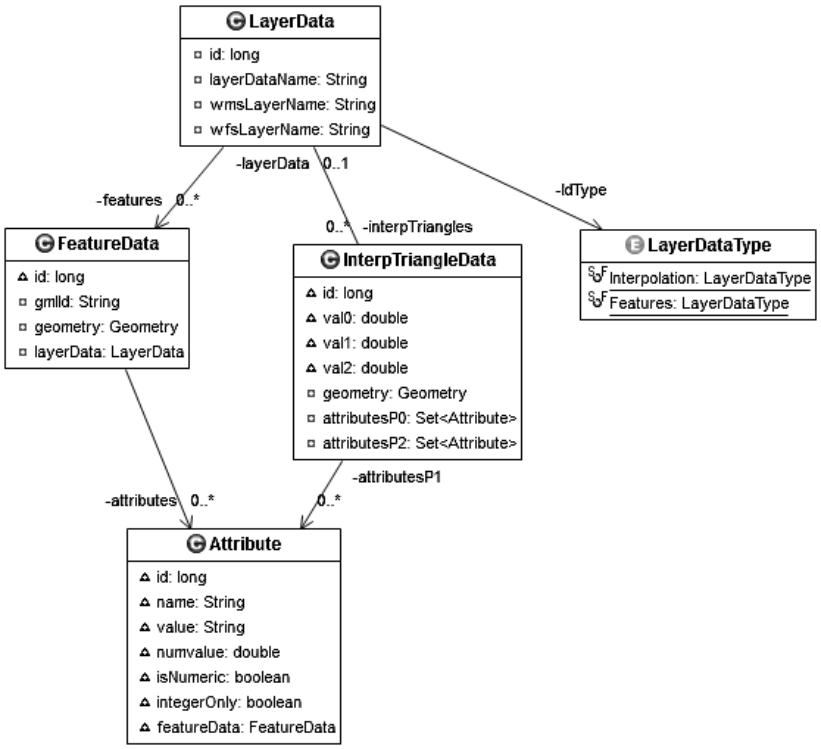
- stroke* — kolor linii (obrysu wielokątów i punktów); zestaw wartości dla kanałów RGB i alpha z zakresu 0–255
- fill* — kolor wypełnienia wielokątów i punktów; zestaw wartości dla kanałów RGB i alpha z zakresu 0–255

- pointRsize — promień koła rysowanego jako symbol obiektów punktowych
- outlineStrokeWidth — szerokość linii (obrysu)
- featureRestriction — kryteria zastosowania stylu; styl zostanie zastosowany do narysowania tylko tych obiektów, które spełniają zadane tu kryteria
- interpolationColors — gradient; używany do wyrysowywania danych interpolacyjnych

FeatureRestriction — kryteria geometryczne i atrybutowe, jakie spełniać musi obiekt, aby zastosować do niego styl. Zawiera:

- geometryType — typ geometrii (np. linia, wielokąt, dowolny)
- restrictionType — typ ograniczenia (np. równy, większy od, dowolny)
- attributeName — nazwa atrybutu
- attributeValue i attributeNumValue — wartość, odpowiednio tekstowa i numeryczna, wykorzystywana przy porównaniach

Obiekt warstwy



Rysunek 8.7: Diagram asocjacji dla danych dotyczących obiektów geometrycznych oraz ich atrybutów

LayerData jest to klasa, która zawiera dane wektorowe, jest powiązana z warstwami WFS (WMS) za pomocą wfsLayerName (wmsLayerName). Zawiera informację o typie danych oraz

zawiera zbiór FeatureData lub InterpTriangleData. FeatureData są to obiekty zawierające dane geometryczne oraz zbiory atrybutów. InterpTriangleData są to obiekty zawierające trójkąty, wartości oraz atrybuty dla każdego z punktów trójkąta. Atrybut to zestaw składający się z nazwy atrybutu oraz jego wartości tekstowej lub numerycznej.

8.2. Działanie aplikacji

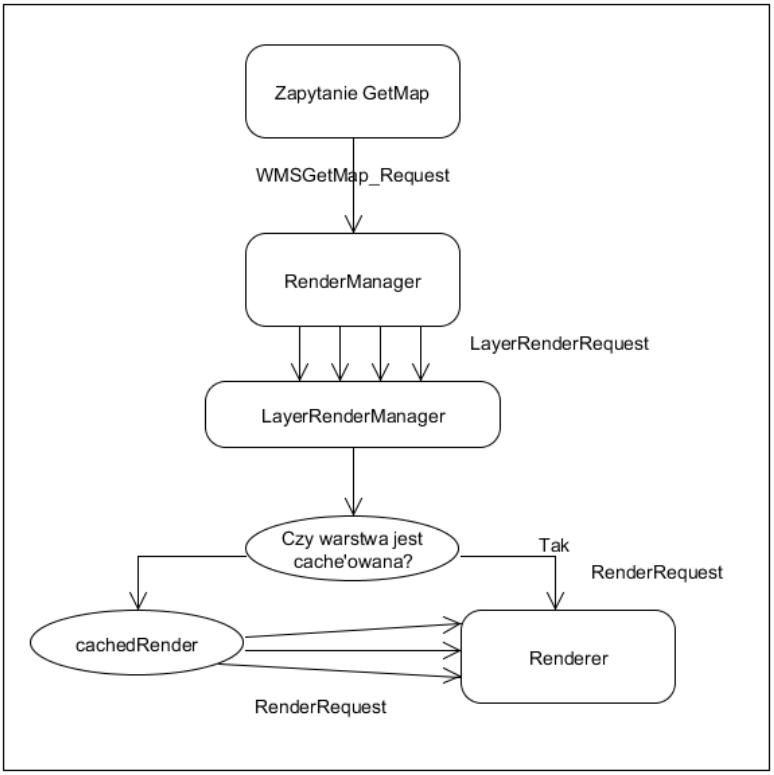
8.2.1. Serwis WMS

Zapytanie GetCapabilities

Do stworzenia odpowiedzi na zapytanie GetCapabilities wykorzystywany jest odpowiednio przygotowany dokument JSP zgodny ze schematem OGC. Wartości, którymi uzupełniane są tagi dokumentu, pobierane są za pomocą mechanizmu Unified Expression Language (EL) z obiektu klasy WmsMetadata. Obiekt ten przechowywany jest na czas zapytania w beanie WMSGetCapabilities_Response, a pobierany z bazy danych w metodzie init tego beana. Pobranie obiektu z bazy odbywa się poprzez wywołanie metody getWmsMetadata wstrzykniętej instancji klasy GISModel.

Zapytanie GetMap

Diagram 8.8 przedstawia w jaki sposób są tworzone odpowiedzi na zapytanie GetMap. Odpowiedź na zapytanie GetMap jest tworzona wieloetapowo. Najpierw serwlet, który jest odpowiedzialny za to, tworzy obiekt WMSGetMap_Request, który zawiera dane samego zapytania. Te dane są przekazywane dalej do obiektu klasy RenderManager, którego zadaniem jest już pobranie części danych dotyczących warstw do wyrysowania, w tym informacji o stylu. W tym momencie tworzone są obiekty LayerRenderRequest, które są przekazywane do obiektu klasy LayerRenderManager. Następnie, w zależności od tego czy warstwa WMS ma włączony cache czy nie wyrysowujemy w odpowiedni sposób grafikę przy użyciu obiektu klasy Renderer. Ta grafika zostaje zakodowana w odpowiednim formacie i w ten sposób zwrócona. LayerRenderManager tworzy obraz na podstwie plików cache (jeśli warstwa ma włączone cachowanie, a pliki nie istnieją, są tworzone) oraz wyrysowując w pamięci mapę. Kafelki to wycinki warstwy otrzymywane poprzez podział prostokąta ograniczającego warstwę. Obrazek o parametrze zoom równym 0 jest kafelkiem pokrywającym całą warstwę. Siatka wyznaczająca kolejne kafelki jest tworzona w wyniku podziału każdego z kafelków na 4, ale o takich samych wymiarach w pikselach co kafelek o mniejszym parametrze zoom.



Rysunek 8.8: Diagram tworzenia odpowiedzi na zapytanie GetMap

Renderer wyrysowuje grafiki przy użyciu Graphics2D, rysując standardowe figury geometryczne wykorzystując podstawowe klasy Javy do rysowania 2D, ustalając odpowiednie parametry do wypełnienia oraz obramowania na bazie wartości w przekazanym stylu. W przypadku danych interpolacyjnych wykorzystywana jest specjalnie napisana klasa TriangleGradientPaint. Klasa ta, przy wyrysowywaniu, najpierw oblicza wartość w danym punkcie trójkąta, na bazie interpolacji liniowej, po czym znajduje odpowiednie wartości graniczne z gradientu, między którymi znaduje się dana wartość, wylicza kolor, wykorzystując po raz kolejny interpolację liniową. Wszystkie dane są pobierane przy użyciu obiektu klasy GISModel, który jest parametrem konstruktora RenderManagera, LayerRenderManagera oraz Renderera.

Zapytanie GetFeatureInfo

Do stworzenia odpowiedzi na zapytanie GetFeatureInfo wykorzystywany jest przygotowany dokument JSP. Generowany XML jest zgodny ze schematem stosowanym przez ESRI ArcGIS. Najpierw jest tworzony obiekt WmsGetFeatureInfo_Request, który pobiera i waliduje parametry zapytania. Następnie za pomocą klasy WmsGetFeatureInfo_Response tworzona

jest odpowiedź. Znajdują się w niej informacje o współrzędnych punktu w zadanym układzie, atrybuty obiektów znajdujących się w tym punkcie oraz — dla danych interpolowanych — interpolowana wartość.

8.2.2. Serwis WFS

Zapytanie GetCapabilities

Do stworzenia odpowiedzi na zapytanie GetCapabilities wykorzystywany jest odpowiednio przygotowany dokument JSP zgodny ze schematem OGC. Wartości, którymi uzupełniane są tagi dokumentu, pobierane są za pomocą EL z obiektu klasy WfsMetadata. Obiekt ten przechowywany jest na czas zapytania w beanie WFSGetCapabilities_Response, a pobierany z bazy danych w metodzie init tego beana. Pobranie obiektu z bazy odbywa się poprzez wywołanie metody getWfsMetadata wstrzykniętej instancji klasy GISModel.

Zapytanie GetFeature

Generowanie dokumentu XML w odpowiedzi na zapytanie GetFeature odbywa się na podobnej zasadzie jak dla zapytania GetCapabilities. Samo zapytanie ma jednak kilka parametrów, na podstawie których bean WFSGetFeature_Response wybiera odpowiednią metodę GisModelu do stworzenia zapytania SQL. GisModel udostępnia do tego celu następujące metody: getFeatureById, getFeatureByTypename, getFeatureWithBBoxTransform, getFeatureWithFilter (Patrz podrozdział: 8.1.2).

Ostatnia z tych metod obsługuje zapytania getFeature z ustalonym parametrem FILTER. Wartość tego parametru —- XML definiujący filtr — zamieniana jest rekurencyjnie przez klasę FilterEncoding na odpowiedni obiekt Hibernate Criteria (z wykorzystaniem rozszerzenia Hibernate Spatial dla filtrów geometrycznych).

Zapytanie DescribeFeatureType

Obsługuje zapytania o schemat XSD dla dynamicznie tworzonych warstw. Jest ono niezbędne w celu prawidłowego odczytania dokumentu zwracanego przez zapytanie GetFeature.

8.2.3. Pozostałe

GetTile

Do stworzenie odpowiedzi na zapytanie GetTile tworzony jest obiekt klasy LayerRenderManager i wywoływana jest metoda wyrysowującą kafelki (runTile) z odpowiednimi parametrami. Obraz żądanego kafelka jest wczytywany z pliku (lub najpierw tworzony, jeśli nie istnieje). Do wyrysowania grafiki wykorzystywana jest klasa Renderer.

CSGetMap

Do stworzenia odpowiedzi na zapytanie CSGetMap wykorzystywany jest serwlet. Tworzy on obiekt klasy CSGetMapRequest przekazywany do RenderManagera, który na podstawie otrzymanych parametrów generuje odpowiednie StyleDefinition. Następnie tworzony jest obiekt klasy LayerRenderRequest (zawierający opis stylu wraz z parametrami zapytania). Na końcu – identycznie jak przy zwykłym zapytaniu GetMap — wykorzystywany jest obiekt klasy LayerRenderManager (różni się tylko wymuszonym pominięciem funkcjonalności cache).

Import plików z danymi interpolacyjnymi

Plik z danymi interpolacyjnymi jest wczytywany i przerwarzany do postaci listy zawierającej punkty z przypisanymi wartościami. Na niej przeprowadzana jest triangulacja Delaunaya, po czym każdy trójkąt jest zapisywany razem z wartościami w jego punktach. Dane są zapisywane jako obiekty klasy InterpTriangleData, zawierające między innymi trójkąty oraz — dla każdego z wierzchołków — dane o wartości oraz atrybuty. Do wczytywania danych csv wykorzystywana jest biblioteka OpenCSV 10. W przypadku danych XML wykorzystujemy standardowe biblioteki udostępniane przez Javę. Do triangulacji używana jest biblioteka JDT.

Kaskadowy import danych z zewnętrznej usługi WFS

Operacja ta została przedstawiona na diagramie aktywności, który znajduje się w załącznikach (14.5). W celu importu danych wysyłane są zapytania getCapabilities i getFeature.

Obsługą odpowiedzi zajmuje się klasa ExternalWFSReplication. Otrzymane z serwera zewnętrznego pliki importujemy wykorzystując klasę GML3Importer (opisana dalej). Do obsługi harmonogramu służy klasa ReplicationScheduler, która po wskazanym przez użytkownika czasie wysyła sygnał do aktualizacji danych. (Patrz również ExternalWFSReplication i ReplicationScheduler 8.1.2)

Import danych z plików KML

Za import plików KML odpowiedzialna jest klasa KMLImporter. Umożliwia ona załadowanie pliku w postaci strumienia lub bezpośrednio z dysku poprzez podanie odpowiedniej ścieżki. Do importu KML wykorzystana została biblioteka Java API for KML. Mapuje ona zawartość pliku na obiekt wewnętrznej klasy KML. Klasa ta udostępnia szereg metod do pobierania danych w postaci obiektów Feature, Document, Folder, Placemark, Geometry, Coordinate, StyleSelectior, Style. Metoda parseAndSave przechodzi po drzewie dokumentu i dodaje nową warstwę WMS i WFS gdy napotka element Folder lub Document. Wszystkim obiektom (Placemark) przypisawane są style. Na koniec wszystkie nowe warstwy zapisywane są do bazy danych.

Import danych z plików GML

Za import plików GML odpowiedzialna jest klasa GML3Importer. Umożliwia ona załadowanie pliku w postaci strumienia lub bezpośrednio z dysku poprzez podanie odpowiedniej ścieżki. Plik wczytywany jest do pamięci w postaci drzewa DOM, a następnie wyszukiwane są wszystkie elementy FeatureMember, które są osobno parsowane przez metodę makeFeature. Każdy taki element zawiera identyfikator, atrybuty i geometrię. Na podstawie tych danych tworzony jest obiekt FeatureData. Geometria parsowana jest za pomocą metody parseGeometryGML klasy FilterEncoding. Na koniec wszystkie nowe warstwy zapisywane są do bazy danych.

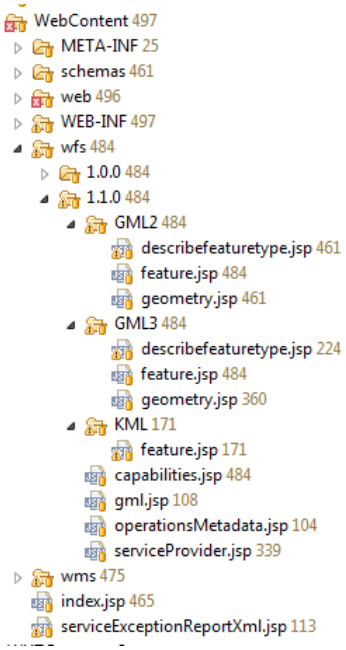
Replikacja

Nasz serwis można skonfigurować tak, aby pracował w architekturze master-slave, gdzie mamy jedno repozytorium centralne oraz dowolną liczbę podłączonych do niego punktów dostępowych. Serwery potomne pozyskują dane z serwera głównego dzięki zastosowanemu mechanizmowi replikacji. Komunikacja między serwerami odbywa się poprzez protokół TCP/IP. W chwili inicjalizacji serwer potomny podłącza się do serwera głównego, przesyła informację

o swoim stanie po czym, jeżeli nie posiada aktualnej, ściąga od niego kopię bazy danych. W momencie w którym slave ma już aktualną bazę danych, serwer główny podłącza się do jego bazy danych za pomocą odpowiednio skonfigurowanego SessionFactory Hibernate'a. W chwili wystąpienia zdarzenia zatwierdzenia transakcji po stronie głównego serwera uruchamia się odpowiednia metoda przygotowanej na potrzeby replikacji klasy przechwytującej zdarzenia Hibernate'a (Interceptor). Klasa ta zbiera informacje o zapisywanych i aktualizowanych danych w bazie danych w przebiegu całej transakcji. Na koniec tworzone są sesje na zdalnych bazach serwerów potomnych i wykonywana jest replikacja zebranych obiektów.

Możliwość dalszej rozbudowy WFS

Aktualnie wspieraną wersją standardu WFS przez WUTGeoserver jest wersja 1.1.0. Istnieje jednak możliwość łatwego rozbudowania systemu o kolejne wersje standardu. Dla każdej wersji powstał oddzielny schemat wyjściowych plików XML. Schematy takie można pobrać w postaci plików XSD ze strony OGC[42]). W celu stworzenia implementacji takiego schematu dla poszczególnych zapytań (np. getFeature) należy w katalogu WebContent/wfs/ dodać katalog o nazwie odpowiadającej implentowanej wersji (np. 1.0.0), a w nim podkatalogi odpowiadające formatom wyjściowym, np. GML2, GML3. Wewnątrz nowo utworzonych katalogów dodajemy pliki JSP o nazwach odpowiadającym konkretnym zapytaniom. Przykładowa struktura katalogów i plików znajduje się na Rysunku 8.9.



Rysunek 8.9: Struktura plików i katalogów reprezentujących różne wersje standardu WFS

W plikach JSP ustawiamy format na XML oraz kodowanie znaków (zalecane UTF-8). Treść plików stanowi implementacja odpowiedniego schematu. Wartości dla tagów pobieramy z beanów wsfGCResp (dla GetCapabilities) i wfsGFResp (dla GetFeature) za pomocą EL i tagów JSF (np. wfsGCResp.metadata.fees).

Możliwość dalszej rozbudowy WMS

Aktualnie wspieraną wersją standardu WMS przez WUTGeoserver jest wersja 1.3.0. Żeby uzyskać starsze wersje usługi WMS musimy po pierwsze rozszerzyć WMSGetMap_Request o wczytywanie, w przypadku starszej wersji standardu, z parametru SRS zamiast parametru CRS, a także rozszerzyć BoundingBox o pole zawierające wersję oraz dodać w nim funkcję opakowującą wywołanie funkcji areXYSwapped z klasy Additionals[1].

8.3. Google Maps API — Google WMS

Rozszerzenie do Google Maps API 3 pozwalające wygodnie dodawać serwisy WMS o nazwie gwms. Najważniejszą funkcją jest gwms, przyjmująca jako parametry obowiązkowo: instancję google.maps.Map, adres WMS ze znakiem zapytania na końcu oraz tablicą nazw warstw.

Biblioteka dodaje do mapy serwis WMS jako instancję google.maps.ImageMapType. Klasa ta przyjmuje w konstruktorze zbiór parametrów, z których najważniejszym jest getTileUrl — funkcja, która dla podanego punktu (współrzędnych) i przybliżenia (zoom) zwraca adres URL obrazka mapy — czyli w przypadku serwera WMS adres URL zapytania GetMap. Funkcją tą jest w naszej bibliotece createGetTile.

Do wygenerowania adresu zapytania GetMap niezbędne są m.in. adres serwisu, lista nazw widocznych warstw w postaci ciągu rozdzielonego przecinkami, wielkość w pikselach obrazu wynikowego (domyślnie ustawiona na standardowe dla Google Maps 256 x 256) oraz prostokąt ograniczający (BBOX). Prostokąt jest wyliczany na podstawie parametrów podawanych przez getTileUrl.

Dodatkowo biblioteka umożliwia wygenerowanie legendy (funkcja createLegend z parametrami: id div oraz obiekt gwms) i umieszczenie jej w postaci listy w obiekcie o id podanym w parametrze.

[1] Funkcja rozwiązująca problem z orientacją osi współrzędnych zależnie od wersji WMS. Więcej o problemie opisaliśmy w Rozdziale 4, w sekcji Wersja 1.1.1 a 1.3.0 (4.2.1)

9 Instrukcja użytkownika

Rozdział ten stanowi kompletną instrukcję instalacji, konfiguracji i korzystania z aplikacji WUTGeoserver. Przedstawiamy w nim dodatkowo opis interfejsu użytkownika oraz szczegółowy opis korzystania z usług udostępnianych przez naszą aplikację.

9.1. Hosting środowiska testowego

W celu przeprowadzenia testów, w tym testów akceptacyjnych, udostępniliśmy naszą aplikację na serwerze o adresie URL http://194.29.178.33/8080/WUTGeoserver/web/. W przypadku problemów z dostępem do tego serwera można skontaktować się z naszym promotorem mgr. inż. Michałem Okulewiczem w celu jego ponownego uruchomienia w przygotowanej przez nas konfiguracji.

9.2. Instalacja i konfiguracja aplikacji na systemie Debian

Do aplikacji został przygotowany skrypt instalacyjny setup.sh, który po uruchomieniu pobiera i instaluje kolejne elementy niezbędne do działania aplikacji. Do prawidłowego przebiegu instalacji wymagane jest uruchomienie instalatora z uprawnieniami roota. W poszczególnych etapach użytkownik będzie proszony o potwierdzenie chęci instalacji niezbędnych pakietów.

9.2.1. Etapy instalacji

- Oracle Java 7 JDK (można pominąć, jeżeli w systemie istnieje już zainstalowana Java w wersji 1.7).
- PostgreSQL 9.1 (wraz z bibliotekami Geospatial Data Abstraction Library (GDAL), Geometry Engine — Open Source (GEOS) 3.6 i dodatkiem PostGIS 2.0.2).
- JBoss Application Server 7.1.1 (z możliwością instalacji jako usługa)

Na koniec użytkownik zostanie poproszony o podanie, czy obecny serwer ma pracować w trybie master czy slave, a w drugim przypadku także o podanie adresu IP serwera master.

9.2.2. Konfiguracja

Podstawowa konfiguracja zapisana zostanie w pliku:
/usr/local/jboss-7.1.1/standalone/configuration/wutgeoserver.properties
Według uznania użytkownika można w nim zmodyfikować lokalizację plików tymczasowych aplikacji zmieniając wartości parametrów temp_files_path i tile_files_path. Ustawienie ścieżki na default oznacza domyślną ścieżkę odnoszącą się do miejsca instalacji JBossa. Pola as_url i as_port odnoszą się do wyświetlanych w dokumentach Capabilities adresów URL do operacji. Domyślne ustawienie as_port=8080 i as_url=auto powoduje, że aplikacja automatycznie pobiera adres IP zewnętrznego interfejsu serwera i wyświetla adresy URL w postaci: http://adres_ip:8080/WUTGeoserver/Service?. W przypadku gdy zmodyfikujemy pole as_ip, pole as_port jest ignorowane. Zmiana trybu pracy serwera (master/slave) wymaga dodatkowego uwzględnienia prawidłowej konfiguracji bazy danych PostgreSQL. Na serwerach slave należy się upewnić, że w pliku /etc/postgresql/8.4/main/pg_hba.conf znajduje się linijka:

```
host giswut giswutuser master_ip/32 md5
```

gdzie master_ip jest lokalnym adresem IP serwera master. Ewentualna modyfikacja pliku pg_hba.conf wymaga następnie ponownego uruchomienia usługi PostgreSQL.

Uwaga: W celu prawidłowego działania aplikacji należy się upewnić, że w sieci lokalnej serwerów nie są blokowane porty TCP 3090-3022.

9.3. Praca z WUTGeoserverem

Serwer pozwala na udostępnianie i zarządzanie danymi przestrzennymi zgodnie ze standardami WMS i WFS, udostępnianie w postaci kafelków i cachowanie za pomocą interfejsu HTTP. Również całe zarządzanie serwerem i usługami odbywa się za pośrednictwem strony internetowej /WUTGeoserver/web/.

Serwer jest gotowy do pracy od razu po instalacji. Zawiera przykładowe dane przestrzenne udostępnione przez WFS i WMS.

9.3.1. Informacje kontaktowe

Nazwa zakładki: Contact Information
Adres: /WUTGeoserver/web/contact

Informacje o organizacji udostępniającej dane przestrzenne. Wykorzystywane są w metadanych usług WMS i WFS jako informacje kontaktowe. Wszystkie pola są opcjonalne.

9.3.2. Import

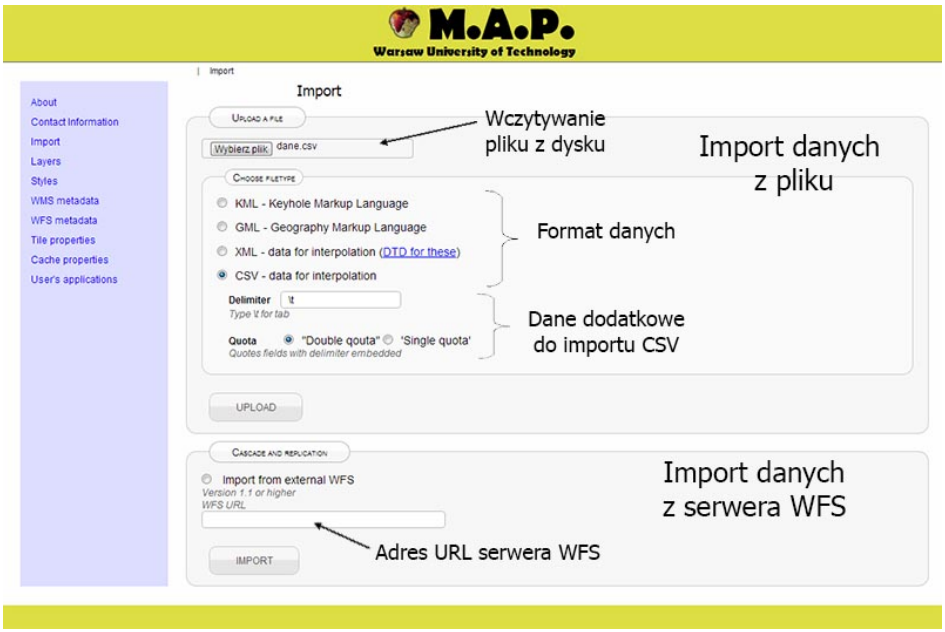
Nazwa zakładki: Import

Adres: /WUTGeoserver/web/import

Import z plików

Serwer pozwala importować wektorowe dane przestrzenne w formacie: GML (3.1.3) i KML z dysku klienta (Rysunek 9.1). Ponadto pozwala pobrać dane punktowe w formacie CSV lub XML (DTD dostępne pod adresem /WUTGeoserver/web/data/XMLInterpDTD.dtd), na podstawie których tworzona jest warstwa WMS z interpolacją danych. Dane interpolacyjne w postaci CSV wymagają dodatkowej konfiguracji, ktora jest możliwa po wczytaniu pliku. Wymagane jest podanie pól, w którcy znajdują się współrzędne geograficzne punktu oraz pola z wartością do interpolacji. Niezbędne jest podanie poprawnego prostokąta ograniczającego (bounding box), aby zapewnić poprawne wczytywanie warstwy przez klientów WMS (QuantumGIS, Geoportal). Jest możliwość ustalenia podstawowego stylu — przypisania kolorów dla minimalnych i maksymalnych wartości. Można również ograniczyć zakres tych wartości (opcjonalne). W przypadku danych w formacie XML informacje te są zawarte w samym pliku, więc nie jest potrzebna dodatkowa konfiguracja.

Wczytane dane automatycznie udostępniane są w usługach WMS i WFS. Dane interpolacyjne są udostępniane jedynie przez WMS.



Rysunek 9.1: Formularz importu danych przestrzennych

Import z zewnętrznych serwisów WFS

Aplikacja umożliwia import danych z zewnętrznych serwisów WFS. W tym celu należy na stronie importu zaznaczyć opcję „Import from external WFS" oraz podać adres URL serwera WFS (bez parametrów). Po naciśnięciu przycisku Import wyświetli się lista warstw dostępnych na wskazanym serwerze. Użytkownik może wybrać dowolną liczbę warstw do zaimportowania, a ponadto określić częstotliwość aktualizacji danych (opcja „never" oznacza, że import dokona się jednorazowo i aktualizacje nie będą przeprowadzane) (Rysunek 9.1).

9.4. Warstwy

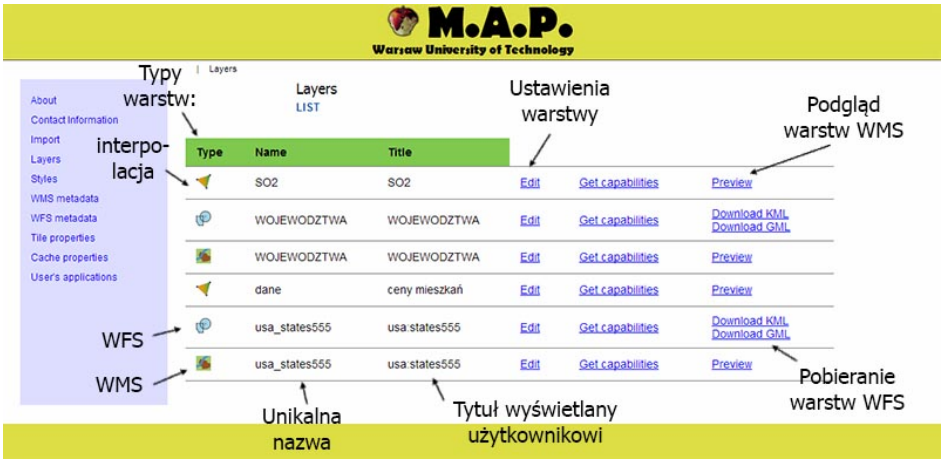
Nazwa zakładki: Layers

Adres: /WUTGeoserver/web/layers

Warstwy są podstawową jednostką udostępniającą dane przestrzenne. Można rozróżnić warstwy udostępniane przez WMS (jako obrazy rastrowe) i WFS (jako dane wektorowe). Dane rastrowe mogą być cachowane lub udostępniane jako kafelki.

W zakładce warstw znajduje się lista wszystkich warstw zapisanych w WUTGeoserverze (Rysunek 9.2). Ikonami oznaczone są typy warstw: WMS, WFS oraz interpolacyjne. Z poziomu tego widoku możliwe jest ściągnięcie danych wektorowych w formacie KML i GML oraz podgląd warstwy WMS.

Edycja metadanych warstwy jest możliwa po kliknięciu linku „Edit". Strony edycji umożliwiają zmianę wyświetlanego użytkownikowi tytułu warstwy, dodanie opisu, słów kluczowych, wybór dostępnych odwzorowań, a w przypadku warstw WMS również styli.



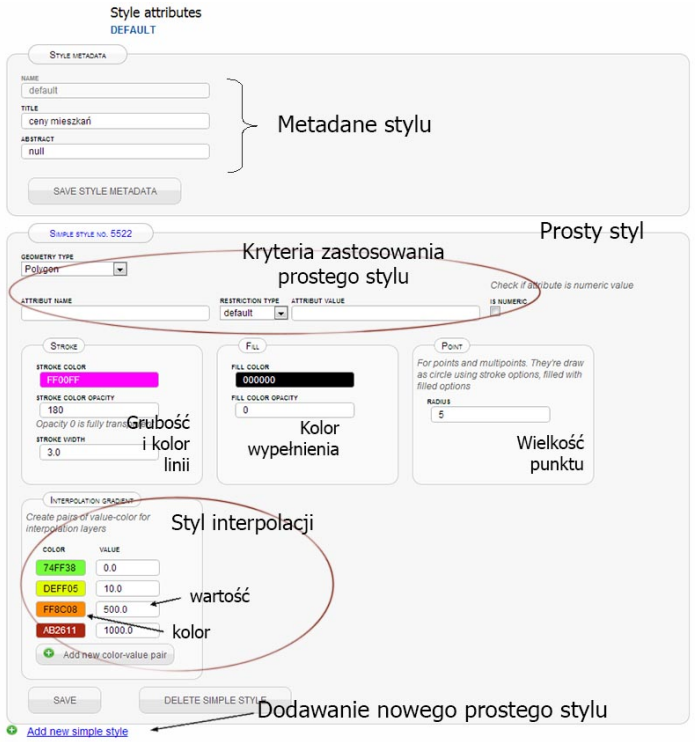
Rysunek 9.2: Lista warstw — Layers

9.5. Style

Nazwa zakładki: Styles

Adres: /WUTGeoserver/web/styles

Wszystkie warstwy rastrowe mają przypisany co najmniej jeden styl rysowania, który określa, w jaki sposób dane wektorowe zostaną przedstawione na obrazie udostępnianym przez WMS. Podczas wczytywania danych i generowania warstwy rastrowej tworzony jest dla niej domyślny styl. Lista styli, z których korzystać może dana warstwa, znajduje się w zakładce z metadanymi warstwy (Layers). Lista styli dostępnych w serwisie znajduje się w zakładce Styles.



Rysunek 9.3: Formularz definicji stylu

Stosujemy podstawowe syle. Zwykłe obiekty geometryczne możemy wypełnić i obrysować kolorem z przezroczystością, a także uzależniać te parametry od typu obiektu geometrycznego oraz od atrybutów, wykorzystując relacje „większe niż”, „mniejsze niż” oraz „równe”. Dane interpolacyjne rysujemy gradientem wyliczanym na bazie wartości w punkcie.

Wybór sposobu rysowania, które zostanie wykorzystany do wyrysowania obiektu polega na sprawdzaniu po kolei, czy obiekt spełnia kryteria związane z danym opisem sposobu rysowania. Obiekt jest rysowany przy pomocy pierwszego stylu, którego warunki spełnia.

Gradient do rysowania danych interpolacyjnych jest formułowany jako posortowana lista par kolor — wartość (punktów gradientu). Jeśli wyinterpolowana wartość piksela znajduje się między punktami gradientu obliczany jest jej kolor na podstawie odległości od nich. Jeśli jest poza zakresem objętym przez punkty, wykorzystywany jest punkt o najbliższej wartości. Formularz edycji stylu znajduje się na rysunku 9.3

Warstwy wektorowe nie mają przypisanych styli — są czystymi danymi bez informacji o metodzie wizualizacji.

9.6. Web Map Service

Adres usługi:/WUTGeoserver/Service

Metadane usługi: /WUTGeoserver/Service?service=WMS&request=GetCapabilities

Usługa WMS umożliwia pobieranie mapy składającej się z udostępnianych warstw rastrowych. Lista warstw, w tym udostępnianych przez WMS, znajduje się w zakładce Layers. Edycja danych warstwy WMS jest możliwa po wybraniu nazwy warstwy z listy. W serwisie udostępniane są tylko te warstwy rastrowe, które nie zostały wyłączone. Opis parametrów podstawowych zapytań (GetCapabilities, GetMap, GetFeatureInfo) znajduje się w rozdziale 4.2.1.

9.6.1. Informacje o WMS

Nazwa zakładki: Web Map Service Metadata

Adres: /WUTGeoserver/web/wmsMetadata

Zawiera podstawowe informacje o usłudze, takie jak tytuł, opis, słowa kluczowe. Są to dane, które otrzymuje klient usługi w odpowiedzi na zapytanie GetCapabilities. Służą do wyszukiwania danych — wpisane tu informacje mogą być wykorzystane przez serwisy wyszukiwarek usług przestrzennych.

Zapytania CSGetMap

Adres usługi:/WUTGeoserver/CSGetMap

Zapytanie CSGetMap obsługuje te same parametry co zapytanie GetMap, ale z jedną różnicą — jako STYLES jest podana definicja parametrów w postaci XML (w samym zapytaniu już bez tabulatur). Obsługiwane są jedynie zapytania POST:

```
<Styles>
  <StyleDefinition>
```

```
    <SimpleStyle>
      <Restriction geometryType='Point' restrictionType='Default'
      attributeName='' attributeVal=''/>
      <FillColor hexValue='\#00FF00'/>
      <StrokeColor hexValue='\#0000FF' alpha='100'/>
    </SimpleStyle>
...
  </StyleDefinition>
  <ExistingStyle name='nazwaStylu'/>
  <StyleDefinition>
    <InterpolationColorData>
      <InterpolationColorVal hexColor='\#00FF00' value='1'/>
      <InterpolationColorVal hexColor='\#000000' value='34'/>
    </InterpolationColorData>
  </StyleDefinition>
</Styles>
```

Tag StyleDefinition definiuje styl dla warstwy z obiektami geometrycznymi, ExisitingStyle pobiera istniejący styl, a StyleDefinition określa wygląd warstwy z danymi interpolacyjnymi. Możliwe wartości geometryType: Point, MultiPoint, Line, MultiLine, Polygon, MultiPolygon, GeometryCollection, Any. Możliwe wartości restrictionType: AttributeEqual, AttributeValHigher, AttributeValLower, Default.

9.6.2. Serwis z kafelkami

Adres usługi:/WUTGeoserver/GetTile

Nazwa zakładki: Tiles

Adres: /WUTGeoserver/web/tiles

Aby wykorzystać serwis z kafelkami trzeba najpierw zdefiniować warstwę kafelkową w zakładce Tiles (Rysunek 9.4. Następnie można już odpytywać o kafelki. W zapytaniu podajemy nazwę warstwy oraz współrzędne x, y oraz zoom, np: `GetTile?layer=layername&x=2&y=2&zoom=3`

Przykłady zapytań znajdują się w Załączniku

9.7. Web Feature Service

Adres usługi (dla aplikacji klienckich):/WUTGeoserver/Service

Metadane usługi: /WUTGeoserver/Service?service=WFS&request=GetCapabilities

Serwer udostępnia usługę WFS w wersji 1.1.0. Umożliwia pobieranie i filtrowanie danych wektorowych warstw udostępnianych przez serwer. Dostępne są dwa formaty wyjściowe: GML w wersji 3 i KML

Rysunek 9.4: Tworzenie nowej warstwy kafelkowej

9.7.1. Informacje o WFS

Nazwa zakładki: Web Feature Service Metadata

Adres: /WUTGeoserver/web/wfsMetadata

Zawiera podstawowe informacje o usłudze, takie jak tytuł, opis, słowa kluczowe. Są to dane, które otrzymuje klient usługi w odpowiedzi na zapytanie GetCapabilities. Służą do opisu i wyszukiwania danych przestrzennych. Lista warstw WFS znajduje się w zakładce Layers. W serwisie udostępniane są tylko te warstwy, które nie zostały wyłączone.

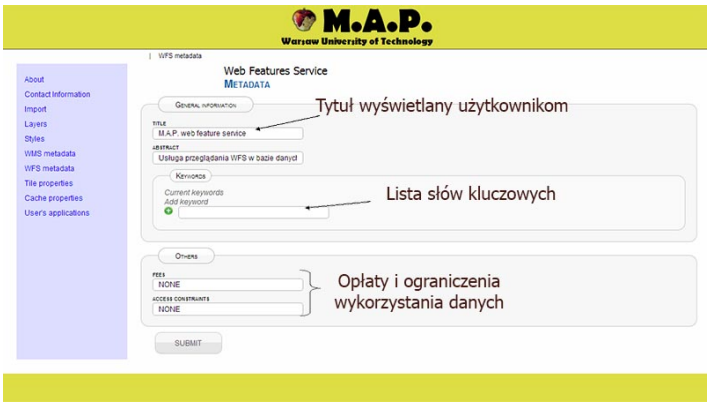
9.7.2. Korzystanie z usługi WFS

Zapytanie GetFeature

Zapytanie GetFeature umożliwia pobranie dokładnych danych wektorowych obiektów i atrybutów udostępnianych warstw. Oprócz podstawowych parametrów zapytania WFS (VERSION, REQUEST, SERVICE) obsługuje parametry:

- SRSNAME — oznaczenie układu odniesienia w formacie EPSG:xxxx (opcjonalny, w przypadku pominięcia zostanie wybrany domyślny układ EPSG:4326),
- BBOX — prostokąt ograniczający wyszukiwane obiekty, we współrzędnych odpowiadających układowi odniesienia podanej projekcji (SRSNAME), w formacie: minx, miny, maxx, maxy (parametr opcjonalny, nie można stosowaś jednocześnie z FILTER i FEATUREID, wymaga podania parametru TYPENAME)
- TYPENAME — lista nazw warstw, które będą uwzględniane w wyszukiwaniu obiektów (wymagane, jeżeli nie podano parametru FEATUREID),

- FILTER — filtr wyszukiwania zgodny ze specyfikacją OGC i obsługiwanymi rodzajami wymienionymi w capabilities (opcjonalny, nie można stosować jednocześnie z BBOX i FEATUREID, wymaga podania parametru TYPENAME),
- OUTPUTFORMAT — format w jakim zwracane są dane, możliwe: GML3, KML (parametr opcjonalny, domyślnie: GML3),
- RESULTTYPE — rodzaj zwracania wyników, możliwe wartości: hits (liczba wystąpień), result (lista obiektów),
- MAXFEATURES — ograniczenie górne na liczbę zwracanych wyników



Rysunek 9.5: Metadane usługi WFS

Filtry dla zapytania GetFeature

Filtry podajemy w formacie drzewa XML + GML. Dostępne są następujące filtry do zapytań:

- Porównujące wartości atrybutów: LessThan, GreaterThan, LessThanEqualTo, GreaterThanEqualTo, EqualTo, NotEqualTo, Like, Between, NullCheck
- Relacje geometryczne BBOX, Equals, Disjoint, Intersects, Touches, Crosses, Within, Contains, Overlaps, Beyond
- Operatory logiczne: AND, OR, NOT

Składnia tworzenia filtrów jest następująca:

```
<Filter>
  <PropertyName>
    <!|- nazwa filtrowanego atrybutu lub SHAPE dla filtrów geometrycznych-->
  </PropertyName>
  <!|- Filtr lub ciąg filtrów połączonych operatorami logicznymi-->
</Filter>
```

Pojedyncze wartości (liczbowe bądź tekstowe) podajemy pomiędzy tagami `<Literal>`

Przykładowe zapytania zamieściliśmy w rozdziale Załączniki.

Zapytania DescribeFeatureType

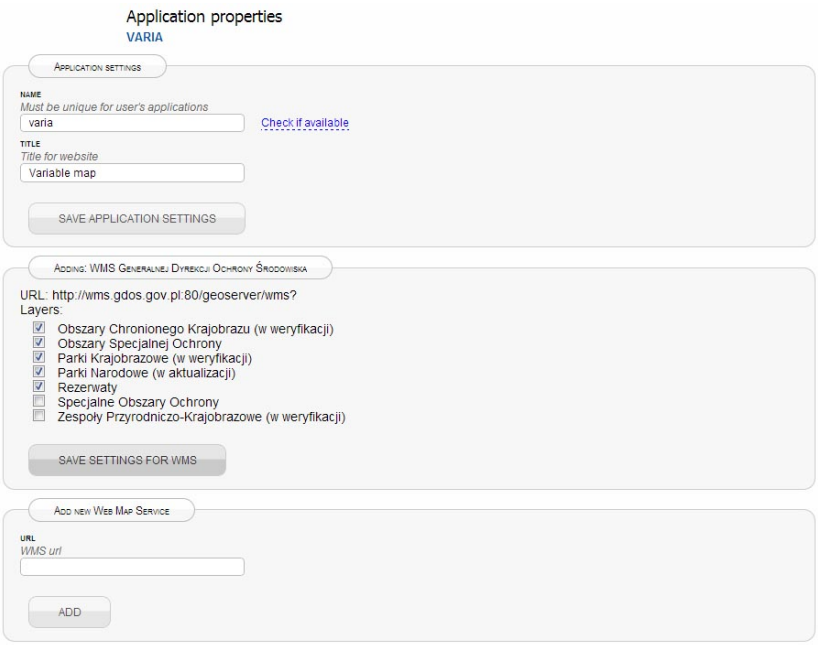
Zapytania DescribeFeatureType pozwalają na uzyskanie schematu XSD dla warstw wyświetlanych w formacie GML. Dla każdej warstwy tworzona jest implementacja typu GML:AbstractFeature uwzględniająca właściwe dla niej atrybuty. Jedynym dodatkowym parametrem dla zapytania DescribeFeatureType jest opcjonalny parametr TYPENAME, który umożliwia podanie listy warstw dla których mają zostać wyświetlone schematy. Niepodanie tego parametru jest równoznaczne z wyświetleniem wszystkich warstw.

9.8. Aplikacje użytkownika

Nazwa zakładki: User's application

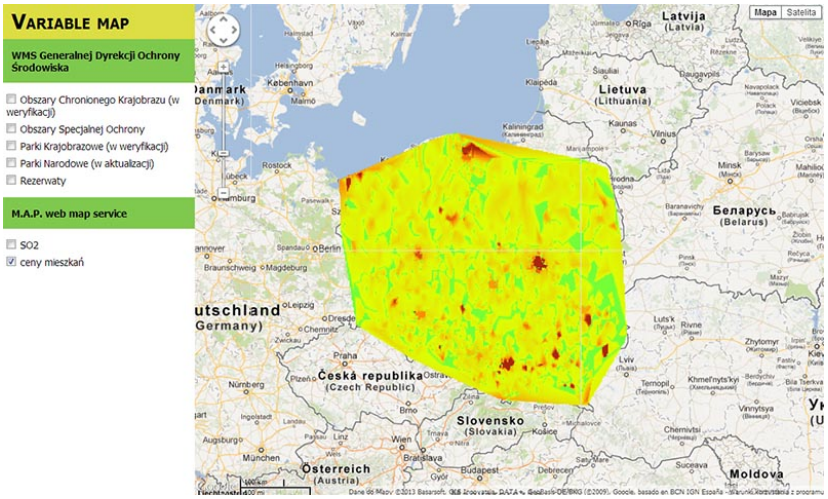
Adres: /WUTGeoserver/web/apps

Aplikacje użytkownika to szybki kreator prostych kompozycji mapowych. Umożliwia stworzenie mapy o unikalnej nazwie, która dostępna będzie pod adresem /WUTGeoserver/app/[nazwa]. W zakładce aplikacji można stworzyć nową mapę oraz edytować istniejące. Interfejs pozwala dodać do aplikacji lokalne dane oraz zewnętrzne serwery WMS i wybrać warstwy, które chcemy wyświetlić na mapie.



Rysunek 9.6: Formularz tworzenia aplikacji użytkownika

Aplikacje użytkownika — mapy Widok przykładowej mapy użytkownika znajduje się na rysunku 9.7. Cała aplikacja wykorzystuje Google Maps API do wyświetlania map oraz autorską bibliotekę gwms do nałożenia warstw z WMS oraz wygenerowania legendy znajdującej się z lewej strony. Lista widocznych warstw umożliwia włączenie i wyłączenie widoczności danych.



Rysunek 9.7: Aplikacje użytkownika — mapa z nałożonymi danymi interpolacyjnymi

10 Wykorzystane biblioteki

Wykaz wykorzystanych bibliotek open source wraz z licencjami

- Apache Commons http://commons.apache.org [licencja Apache v.2.0]
- Apache HttpComponents Client (HttpClient) http://hc.apache.org/ [licencja Apache v.2.0]
- JAK (Java Api for KML) http://labs.micromata.de/display/jak/Home [licencja New BSD (BSD 3-Clause)]
- JDT Java Delaunay Triangulation http://code.google.com/p/jdt/ [licencja Apache v.2.0]
- jQuery http://jquery.com [licencja GPL]
- OpenCSV http://opencsv.sourceforge.net/ [licencja Apache v.2.0]

Licencje New BSD, GPL oraz Apache v.2.0 nie nakładają wymagań, jeżeli chodzi o sposób licencjnowania, na aplikacje wykorzystujące oprogramowania na tych licencjach jako biblioteki. Odnośniki do treści licencji znajdziecie w bibliografii

11 Testy wydajności

Jednym z elementów, na które zwracaliśmy szczególną uwagę przy pisaniu WUTGeoserver a była wydajność oraz skalowalność. Z tego względu ważne było przetestowanie serwera pod względem wydajności. Wykorzystaliśmy do tego celu aplikację Apache JMeter. Pliki testów oraz raporty zostały umieszczone na płycie w załączonych materiałach w folderze test. Testowaliśmy podstawowe usługi serwisu WMS oraz WFS, przy czym skupiliśmy się na zapytaniach GetFeature oraz GetMap. Pomiary czasu bazowaliśmy na czasie oczekiwania, nie na czasie do pobrania zawartości zapytania, ponieważ w przypadku plików graficznych prędkość łącza odgrywała zbyt dużą rolę.

Testy przeprowadziliśmy na komputerach pracujących pod kontrolą systemu linux, dystrybucji Debian. Na obu był zainstalowany JBoss w wersji 7.1.1, PostgreSQL w wersji 9.1 oraz Apache2 w wersji 2.2.16. Apache był wykorzystywany jak proxy w przypadku testów wydajności, a w przypadku testów skalowalności także jako load balancer.

Niestety z powodu wymagań licencyjnych nie możemy udostępnić pliku z danymi, na których przeprowadzaliśmy testy.

Plan wykonywanych testów był zapisany w jednym pliku, WUT.jmx, zmieniane były jedynie niektóre ustawienia.

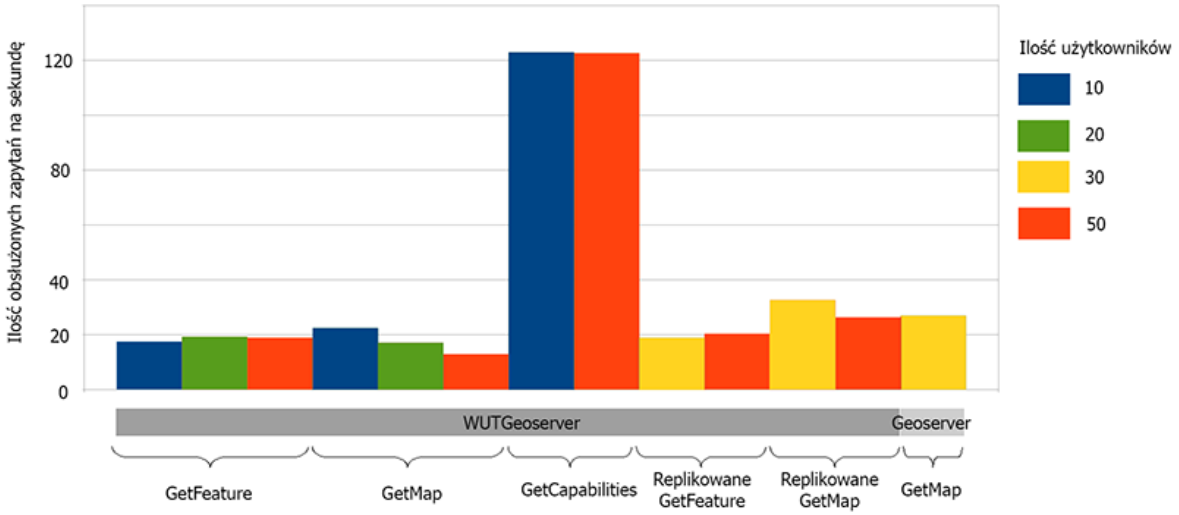
Najpierw sprawdziliśmy wydajność wykonywania pojedynczych zapytań. Ustawiliśmy jednego użytkownika (jeden wątek). Czasy odpowiedzi przedstawiono na rysunku 11.1

W kolejnym kroku chcieliśmy przetestować serwer pod obciążeniem. W tym celu uruchomiliśmy testy dla różnej liczby użytkowników działających równolegle. Na wykresach zaznaczono uzyskane wyniki. Rysunek 11.2 przedstawia zależność ilości obsłużonych zapytań na sekundę od liczby użytkowników i rodzaju zapytania. Rysunek 11.3 natomiast ilustruje

Zapytanie	Czas odpowiedzi [ms]		
	średni	maksymalny	minimalny
WFS GetCapabilities	18.4766	335	9
WFS GetFeature	114.5729	835	35
WMS GetCapabilities	21.4782	787	9
WMS GetMap	58.8222	790	17

Rysunek 11.1: Wydajność wykonywania pojedynczych zapytań

średni czas odpowiedzi w zależności od tych samych parametrów. Dla porównania przeprowadziliśmy również testy na aplikacji Geoserver [26]; należy jednak podkreślić, że Geoserver został uruchomiony na wolniejszej maszynie. Określenia „Replikowane" zapytanie oznaczają, że serwery skonfigurowano w architekturze master-slave. Ponadto testy WUTGeoservera zostały ustawione tak, aby w ciągu dwóch minut powoli zwiększać listę użytkowników od zera do danej.

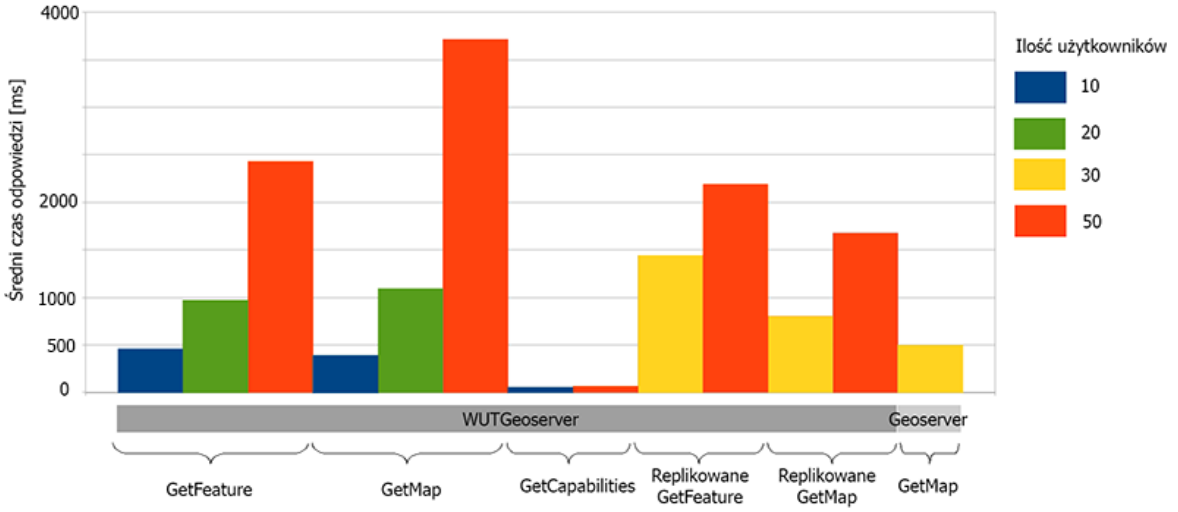


Rysunek 11.2: Ilość zapytań, które obsługiwaliśmy w ciągu sekundy

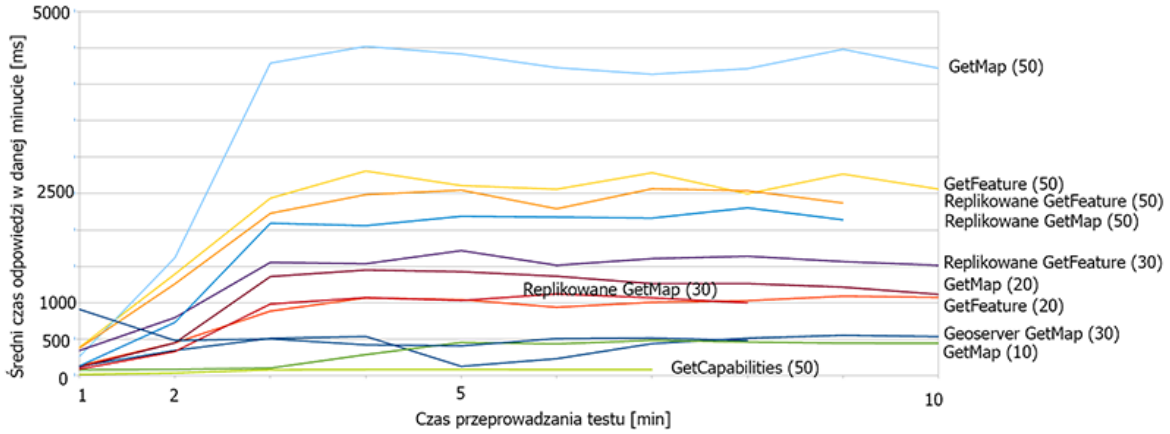
Pierwszym nasuwającym się wnioskiem jest fakt, że replikacja znacząco poprawia wyniki testów. Niestety wyniki Geoservera są bardzo zbliżone, co oznacza, że w przypadku maszyn o zbliżonych parametrach nasza aplikacja przegrywa pod względem wydajności. Prawdopodobnie ma to związek z tym, że Geoserver przetrzymuje bardzo dużą część danych w pamięci aplikacji, my natomiast trzymamy wszystkie dane w bazie danych i za każdym razem musimy je z niej pobierać.

Kolejną kwestią jest to, że mimo podobnej ilości zapytań obsługiwanych w ciągu sekundy, czas oczekiwania pojedynczego klienta na odpowiedź na zapytanie GetFeature oraz GetMap wydłużał się znacznie wraz ze wzrostem liczby klientów. Z tego możemy wywnioskować, że serwis mimo, tego, że wydaje się, że powinien mieć dobrą ilość zapytań jakie może przetworzyć to nie będzie nadawał się do wykorzystania, kiedy będzie pracować na nim ponad 50 osób równolegle, w przypadku jednego serwera. Na szczęście dodanie dodatkowego serwera poprawia w znacznym stopniu wydajność

Wykres na rysunku 11.4 przedstawia średni czas odpowiedzi w funkcji czasu trwania testu. Tylko Geoserver startował z pełną liczbą użytkowników, stąd stosunkowo długi czas



Rysunek 11.3: Czas średni dla zapytań



Rysunek 11.4: Zmiana w czasie odpowiedzi w zależności od długości trwania testu. W nawiasie podano liczbę użytkowników

jego odpowiedzi na początku. Średni czas trwania zapytania ustalał się na stabilnym poziomie już po trzeciej minucie. Po raz kolejny widać, że wydajność Geoservera jest lepsza niż naszej aplikacji.

Podsumowując, nasza aplikacja w momencie kiedy jest tylko na jednym serwerze, a zaczyna korzystać z niej większa liczba użytkowników równolegle zaczyna pracować zbyt długo nad każdym zapytaniem, żeby można to było wykorzystywać. Na szczęście dodanie kolejnych serwerów pozwala w znacznym stopniu poprawić te czasy.

12 Podział pracy

Podział pracy był następujący:

Paweł Krawczyk W swojej części pracy inżynierskiej zająłem się, w pierwszym etapie, stworzeniem architektury aplikacji, tzn. zintegrowanego ze szkieletem JSF modelu zgodnego ze wzorcem MVC, a także programowym obsłużeniem architektury serwerowej typu master-slave. Stworzyłem moduł importu danych z plików KML i GML i serwisów zewnętrznych oraz moduł replikacji. W ramach oprogramowania usług zająłem się usługą WFS, w tym operacjami GetCapabilities, GetFeature i DescribeFeatureType oraz eksportem danych do plików w formacie KML.

Aneta Rosłan

W aplikacji zajmowałam się implementacją zapytań GetCapabilities i GetFeatureInfo usługi WMS. Stworzyłam interfejs web oraz aplikacje użytkownika (służące do tworzenia kompozycji mapowych). Napisałam bibliotekę JavaScript rozszerzającą Google API v.3 o wygodne dodawanie serwisów WMS. Byłam odpowiedzialna za interpretację i wyjaśnianie zagadnień geodezyjno-kartograficznych oraz związanych z nimi przepisów prawnych. Kontrolowałam poprawność używanej terminologii.

Maciej Wierzchowski

Moja część tej pracy polegała głównie na implementacji usługi GetMap oraz rzeczy z nią związanych. W to się wlicza obsługa danych interpolacyjnych, standardowe zapytania GetMap, GetMap ze stylem zdefiniowanym w zapytaniu, GetTile, a także obsługa cache'a. Oprócz tego stworzyłem moduł importu danych interpolacyjnych w formatach CSV oraz XML.

13 Podsumowanie

Celem tej pracy było stworzenie systemu, który będzie udostępniał dane przestrzenne zgodnie ze standardami — poprzez usługi WMS oraz WFS. Dodatkowo uwzględniliśmy implementacje kilka usług dodatkowych, w celu rozszerzenia funkcjonalności.

Początkowo zamierzaliśmy rozszerzyć istniejącą aplikację o otwarym kodzie — GeoServer. Niestety trudności związane m.in. z brakami w dokumentacji zmusiły nas do zmiany koncepcji. Dalsze rozważania doprowadziły nas do biblioteki GeoTools, aczkolwiek po przeszukaniu możliwych źródeł natrafiliśmy na te same problemy co w przypadku aplikacji Geoserver. Podjęliśmy więc decyzję, że aplikacja w całości zostanie napisana przez nas. Do tego celu wybraliśmy technologię Java, m.in. ze względu na prostotę pisania aplikacji w tym języku, dobrze rozwiniętą technologię serwerową oraz wieloplatformowość.

Jako bazę danych wybraliśmy PostgreSQL z rozszerzeniem PostGIS ponieważ udostępniał bardzo ważne dla nas elementy takie jak typ geometry, pozwalający na zapis geometrii różnego typu oraz szeroka gama funkcji pozwalająca nam na nim operować, takich jak przecięcie, wyliczenie prostokąta ograniczającego oraz transformację pomiędzy projekcjami geodezyjnymi.

W trakcie implementacji usług pierwsze problemy na jakie się natknęliśmy były związane z wyborem układu osi współrzędnych. Niektóre układy współrzędnych pierwszą oś mają skierowaną na wschód, a inne na północ (p. też. 4.2.1). Natomiast, jak się okazało, baza danych PostGIS ma definicje transformacji między różnymi układami współrzędnych, które zakładają zawsze, że oś x jest w osi wschód-zachód, a y północ południe. Dodatkowo okazało się, że serwis Geoportal w wersji pierwszej nie trzyma się standardu, jeżeli chodzi o zwracane dane wektorowe, to samo tyczy się też wyświetlania danych zaimportowanych do niego. Oznacza to, że w jego wypadku, osie układu są ułożone w sposób odwrotny do zdefiniowanego. Mimo, ze to niezgodne ze standardem, to, ponieważ zakładaliśmy współpracę z Geoportalem, importujemy, eksportujemy oraz wyrysowujemy dane w w EPSG:2180 przyjmując takie same osie co Geoportal.

Dodatkowo różnego rodzaju problemy pojawiały się w związku z mapowaniem obiektowo relacyjnym przy użyciu biblioteki Hibernate. Na przykład zaistniał problem z mapowaniem uporządkowanych list, czyli takich, w których chcieliśmy żeby lista zachowywała kolejność

wrzucanych elementów. W jednym ze sprawdzanych rozwiązań, przy usuwaniu elementów było naruszane ograniczenie unikalności kolumny odwołującej się do id obiektów zawartych w liście. Oznaczało to, że najprawdopodobniej lista przy przesortowywaniu była albo od nowa wrzucana do tabeli, jeszcze przed usunięciem starej, albo była zmieniana kolejność w inny sposób niż poprzez zmiane wartości w kolumnie zawierającej indeks elementu na liście. Rozwiązaliśmy to poprzez zaimplementowanie indeksów w tabeli ręcznie, a sama lista jest pobierana w odpowiedniej z kolejności z powodu dodania anotacji OrderBy.

Mimo wielu różnych problemów z bibliotekami oraz trudnościami związanymi z luźnym podejściem do standardów twórców aplikacji pracujących na danych przestrzennych udało nam się stworzyć sprawny system implementujący usługi WMS, WFS oraz dodatkowe, takie jak usługę pozwalającą na wykonywanie zapytań GetMap ze zdefiniowanym stylem w zapytaniu, a także usługę pozwalającą na zwracanie kafelków. Oprócz tego system ten może replikować dane w architekturze master-slaves, posiada interfejs www służący do zarządzania, zmian w metadanych oraz importowania w różnych formatach: KML, GML oraz CSV i XML dla danych interpolacyjnych, możliwy jest także import danych z zewnętrznych serwisów WFS.

Napisaliśmy również bibliotekę rozszerzającą Google API o funkcję wczytującą dane z WMS. Wykorzystujemy ją w Aplikacjach użytkownika — części WUTGeoservera pozwalającej na tworzenie własnych kompozycji mapowych. W Aplikacjach nie zaimplementowaliśmy sporej części bardziej zaawansowanych założonych funkcjonalności. Praktycznie poza możliwościami natywnymi Google Maps API (przesuwanie, powiększanie mapy) dodaliśmy jedynie wspomnianą obsługę WMS z generowaniem do niej interaktywnej legendy.

Nasz system został przetestowany pod względem wydajności i po wprowadzeniu pewnych modyfikacji w programie udało się otrzymać dla pojedynczych użytkowników bardzo dobre wyniki, niestety wyniki przy większym obciążeniu są już dość dość niekorzystne, w szczególności gdy zestawimy je z wynikami dla aplikacji Geoserver.

W przypadku dalszego rozwoju tej aplikacji trzeba zacząć od poprawy wydajności pod obciążeniem, można to robić poprzez przechowywanie ostatnio używanych danych wektorowych oraz optymalizacje związane z Hibernate i indeksami w bazie danych. Zachęcamy, by zająć się implementacją nowszych lub starszych standardów usług WMS/WFS, a także rozbudować system o udostępnianie więcej niż jednej usługi każdego rodzaju. Można również przerobić usługę GetTile na w pełni działający serwis WMTS, zgodny ze standardem. Warto rozważyć dodanie obsługi innych formatów danych przestrzennych.

14 Słownik

- Dane przestrzenne — dane odnoszące się bezpośrednio lub pośrednio do określonego położenia lub obszaru geograficznego[10]
- Długość geograficzna (ang. *longitude*) ozn. λ — jedna ze współrzędnych geograficznych; kąt dwuścienny między płaszczyzną południka przechodzącego przez dany punkt a płaczczyzną przyjętego południka zerowego, o wartościach od $0°$ do $360°$ lub od $-180°$ do $180°$.
- Elipsoida (obrotowa) — bryła powstała przez obrót elipsy wokół własnej osi symetrii; stosowana w obliczeniach jako przybliżony kształt powierzchni Ziemi
- Geodezja — nauka zajmująca się określaniem kształtu i wielkości globu ziemskiego oraz wyznaczaniem położenia i wysokości punktów na powierzchni Ziemi[1].
- Geoida — teoretyczna powierzchnia pokrywająca się z powierzchnią mórz i oceanów Ziemi, przedłużona w sposób umowny pod powierzchnią lądów; jest powierzchnią stałego potencjału pola siły ciężkości[4]
- GML (ang. *Geography Markup Language*) — oparty na XML format danych wektorowych; zawiera w sobie informacje o geometrii i atrybutach obiektów
- GRS 80 — elipsoida obrotowa będąca powierzchnią odniesienia obowiązującą w Polsce [8][9]
- KML (ang. *Keyhole Markup Language*) — oparty na XML format danych wektorowych; zawiera w sobie informacje o geometrii, atrybutach i sposobie wyświetlania obiektu
- Mapa — graficzny obraz powierzchni Ziemi na płaszczyźnie wykonany w zmniejszeniu w sposób określony matematycznie, uogólniony i umowny. Przedstawia za pomocą znaków umownych przedmioty terenowe lub na ich tle zjawiska przyrody i życia gospodarczo-społecznego, dobrane i scharakteryzowane zgodnie z przeznaczeniem mapy[4]
- Odwzorowanie Gaussa-Krügera — odwzorowanie równokątne walcowe poprzeczne, wykorzystywane w polskich układach współrzędnych „1992” i „2000”
- Odwzorowanie kartograficzne (ang. *map projection*) — określony matematycznie sposób odzwierciedlenia powierzchni odniesienia (elipsoidy, kuli) na płaszczyźnie.

- Odwzorowanie Mercatora — odwzorowanie równokątne walcowe normalne, wykorzystywane przez serwisy mapowe Bing Maps, Google Maps, OpenStreetMap
- Powierzchnia odniesienia — bryła przybliżająca powierzchnię Ziemi, zazwyczaj kula lub elipsoida
- Stopień geograficzny — jednostka pomiaru kąta. Jeden stopień (°) to 60 minut (′), a jedna minuta to 60 sekund (″).
- Szerokość geograficzna (ang. *latitude*) ozn. ϕ — jedna ze współrzędnych geograficznych; kąt między prostą łączącą dany punkt ze środkiem kuli a płaszczyzną równika, o wartościach 0° na równiku do 90° na biegunach
- Układ „1992” — układ współrzędnych płaskich prostokątnych, należy do państwowego systemu odniesień przestrzennych, stosowany dla map o małej skali.
- Układ „2000” — układ współrzędnych płaskich prostokątnych, należy do państwowego systemu odniesień przestrzennych, stosowany dla mapy zasadniczej.
- Web Feature Service (WFS) — usługa udostępniania danych wektorowych za pomocą interfejsu HTTP
- Web Map Service (WMS) — usługa udostępniania danych wektorowych w formacie rastrowym za pomocą interfejsu HTTP
- WGS-84 — elipsoida obrotowa, popularny system odniesienia wykorzystywany m.in. w nawigacji satelitarnej i przez Google Maps
- XML (ang. *Extensible Markup Language*) — tzw. rozszerzalny język znaczników; metajęzyk pozwalający na tworzenie własnych zestawów znaczników w formacie czytelnym zarówno przez ludzi jak i maszyny

Literatura

[1] W. Kosiński, „Geodezja", PWN Warszawa 2010

[2] S. Przewłocki, „Geomatyka", PWN Warszawa 2008

[3] J. Gaździcki (red.) „Internetowy Leksykon Geomatyczny", Polskie Towarzystwo Informacji Przestrzennej [PTIP]

[4] J. Jaroszewicz, Wykłady z przedmiotu Mapoznawstwo, 2007

[5] J. Chmiel, Wykłady z przedmiotu Analizy przestrzenne i modelowanie, 2010

[6] J. Beaujardiere. „OpenGIS Web Map Service (WMS) Implementation Specification" for version 1.3.0 [online], ref. 06-042, Open Geospatial Consortium Inc., 2006

[7] P. Vretanos. „OpenGIS Web Feature Service (WFS) Implementation Specification" for version 1.1.0 [online], ref. 04-094, Open Geospatial Consortium Inc., 2005

[8] Rozporządzenie Rady Ministrów z dnia 8 sierpnia 2000 r. w sprawie państwowego systemu odniesień przestrzennych (Dz. U. 2000 nr 70 poz. 821)

[9] Rozporządzenia Rady Ministrów z dnia 15 października 2012 r. w sprawie państwowego systemu odniesień przestrzennych (Dz.U. 2012 poz. 1247)

[10] Ustawa z dnia 4 marca 2010 r. o infrastrukturze informacji przestrzennej (Dz. U. 2010 nr 76 poz. 489)

[11] Dyrektywa 2007/2/WE Parlamentu Europejskiego i Rady z dnia 14 marca 2007 r. ustanawiająca infrastrukturę informacji przestrzennej we Wspólnocie Europejskiej (INSPIRE) (Dziennik Urzędowy Unii Europejskiej 2007 L 108)

[12] Open Geospatial Consortium (OGC) http://www.opengeospatial.org/ (21.01.2013)

[13] Geoportal.gov.pl http://geoportal.gov.pl/ (21.01.2013)

[14] Google Maps http://maps.google.pl/ (21.01.2013)

[15] Google Earth http://www.google.pl/intl/pl/earth/ (21.01.2013)

[16] OpenStreetMap http://www.openstreetmap.org/ (21.01.2013)

[17] Zumi http://www.zumi.pl/ (21.01.2013)

[18] Bing http://www.bing.com/maps/ (21.01.2013)

[19] Targeo http://www.targeo.pl/ (21.01.2013)

[20] EkoMapa http://mapa.ekoportal.pl/ (21.01.2013)

[21] IKAR Państwowego Instytutu Geologicznego http://ikar2.pgi.gov.pl/mvs_viewer/ (21.01.2013)

[22] Wikipedia http://www.wikipedia.org/ (24.01.2013)

[23] Wolfram MathWorld http://mathworld.wolfram.com/ (30.01.2013)

[24] XKCD http://xkcd.com/ (21.01.2013

[25] MSDN http://msdn.microsoft.com (30.01.2013)

[26] Geoserver http://http://geoserver.org/ (21.01.2013)

[27] GeoTools http://www.geotools.org/ (21.01.2013)

[28] Open Source Geospatial Foundation http://www.osgeo.org/ (21.01.2013)

[29] Quantum GIS http://www.qgis.org/ (21.01.2013)

[30] EPSG Geodetic Parameter Registry http://www.epsg-registry.org/ (31.01.2013)

[31] Spatial Reference http://spatialreference.org/ (31.01.2013)

[32] Java SE http://www.oracle.com/technetwork/java/javase/ (21.01.2013)

[33] Java EE http://www.oracle.com/technetwork/java/javaee/ (21.01.2013)

[34] Java Servlet http://www.oracle.com/technetwork/java/index-jsp-135475.html (21.01.2013)

[35] JavaServer Pages (JSP) http://www.oracle.com/technetwork/java/javaee/jsp/index.html (21.01.2013)

[36] JavaServer Faces (JSF) http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html (21.01.2013)

[37] Hibernate http://www.hibernate.org/ (21.01.2013)

[38] Hibernate Spatial http://www.hibernatespatial.org/ (21.01.2013)

[39] PostGIS http://trac.osgeo.org/postgis/ (21.01.2013)

[40] PostgreSQL http://www.postgresql.org/ (21.01.2013)

[41] JBoss http://www.jboss.org/jbossas (21.01.2013)

[42] Schematy XSD dla standardów OGC http://schemas.opengis.net (21.01.2013)

[43] Geospatial Data Abstraction Library (GDAL) http://www.gdal.org/ (21.01.2013)

[44] Geometry Engine — Open Source (GEOS) http://trac.osgeo.org/geos/ (21.01.2013)

[45] Licencja Apache v.2.0 http://www.apache.org/licenses/LICENSE-2.0 (21.01.2013)

[46] Licencja New BSD dla JAK http://labs.micromata.de/display/jak/Licenses (21.01.2013)

Załączniki

Przykłady zapytań

Na przykładzie warstwy: wut:states

GetMap

/WUTGeoserver/Service?service=WMS&request=GetMap&version=1.3.0
&layers=states&styles=&crs=EPSG:4326&bbox=24.955967,-124.73142200000001,
49.371735,-66.969849&format=image/png&width=375&height=255

GetTile

- /WUTGeoserver/GetTile?layer=statesTiles&zoom=0&x=0&y=0
- /WUTGeoserver/GetTile?layer=statesTiles&zoom=1&x=1&y=1

GetFeature

Pobieranie całej warstwy

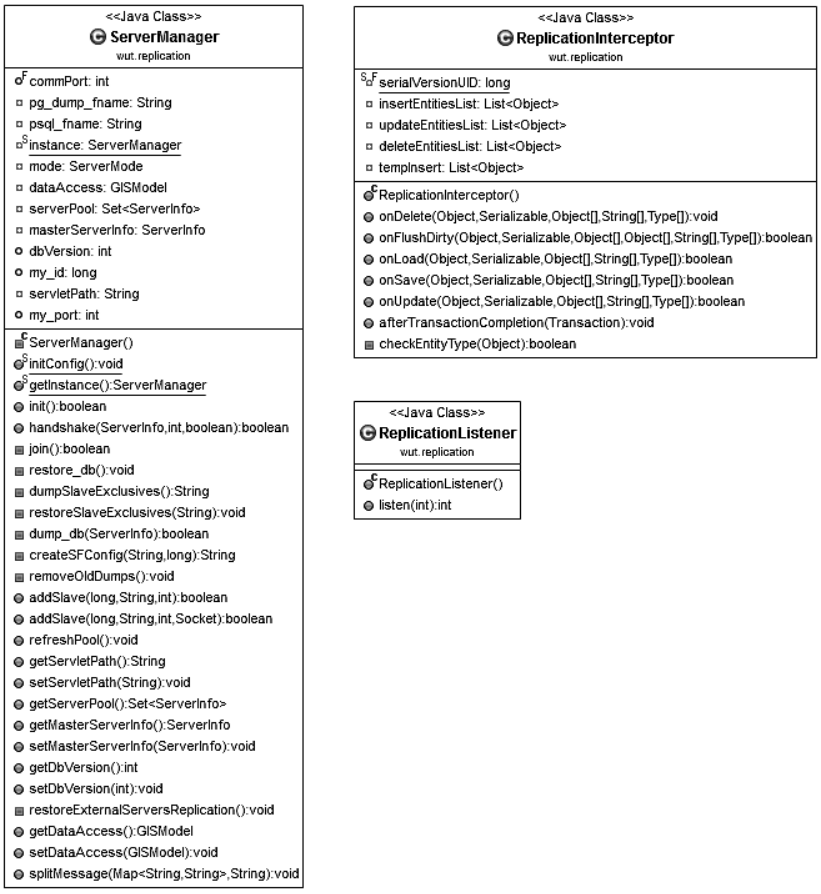
/WUTGeoserver/Service?SERVICE=WFS&REQUEST=GetFeature&TYPENAME=wut:states
&version=1.1.0

Zapytania filtrujące:

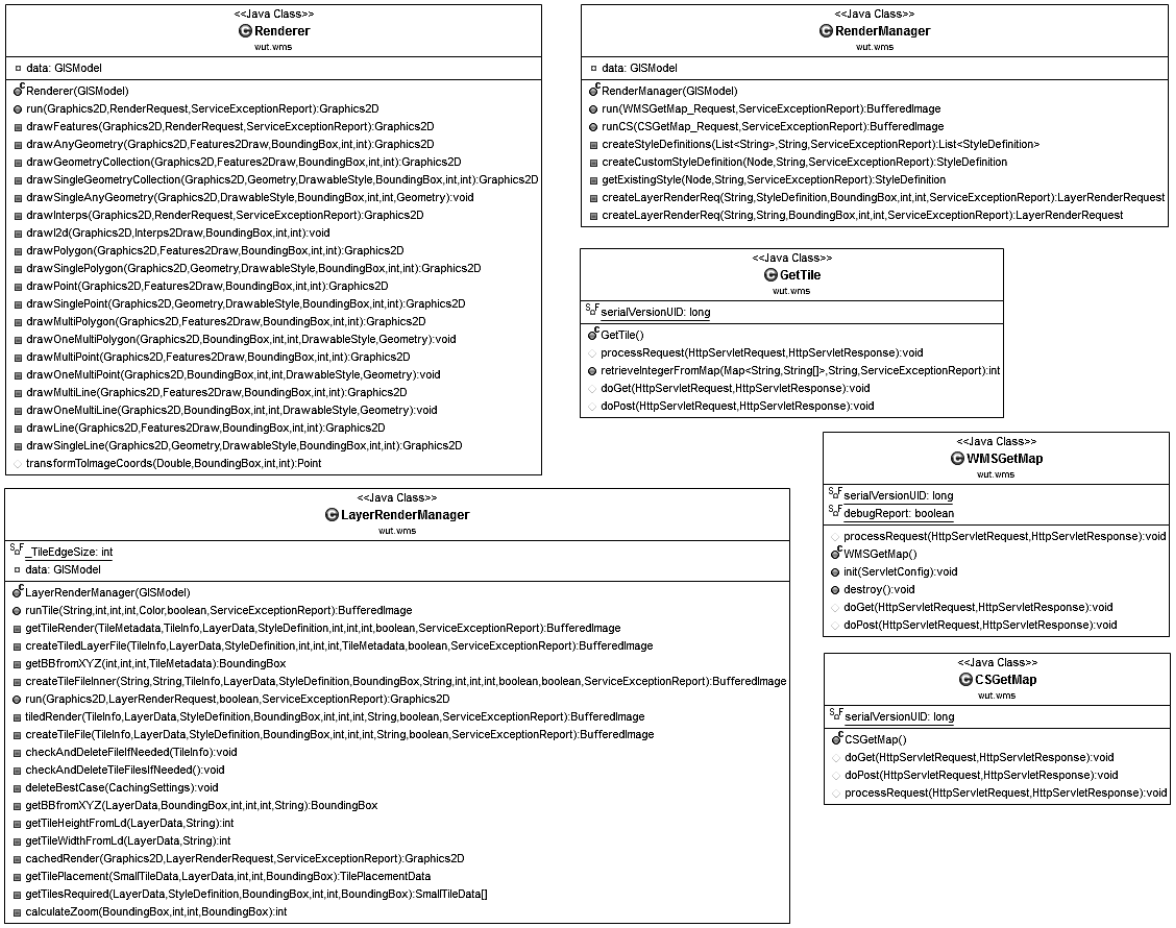
- /WUTGeoserver/Service?SERVICE=wfs&REQUEST=getfeature
  &TYPENAME=wut:states&VERSION=1.1.0&FILTER=<Filter
  xmlns="http://www.opengis.net/ogc"><Like wildcard='*' singleChar='.'
  escape='!'><PropertyName>STATE_NAME</PropertyName>
  <Literal>*lum*</Literal></Like></Filter>
- /WUTGeoserver/Service?SERVICE=wfs&REQUEST=getfeature&VERSION=1.1.0
  &TYPENAME=wut:states&FILTER=<Filter xmlns="http://www.opengis.net/ogc">
  <GreaterThan><PropertyName>FAMILIES</PropertyName><Literal>300000
  </Literal></GreaterThan></Filter>

Diagramy

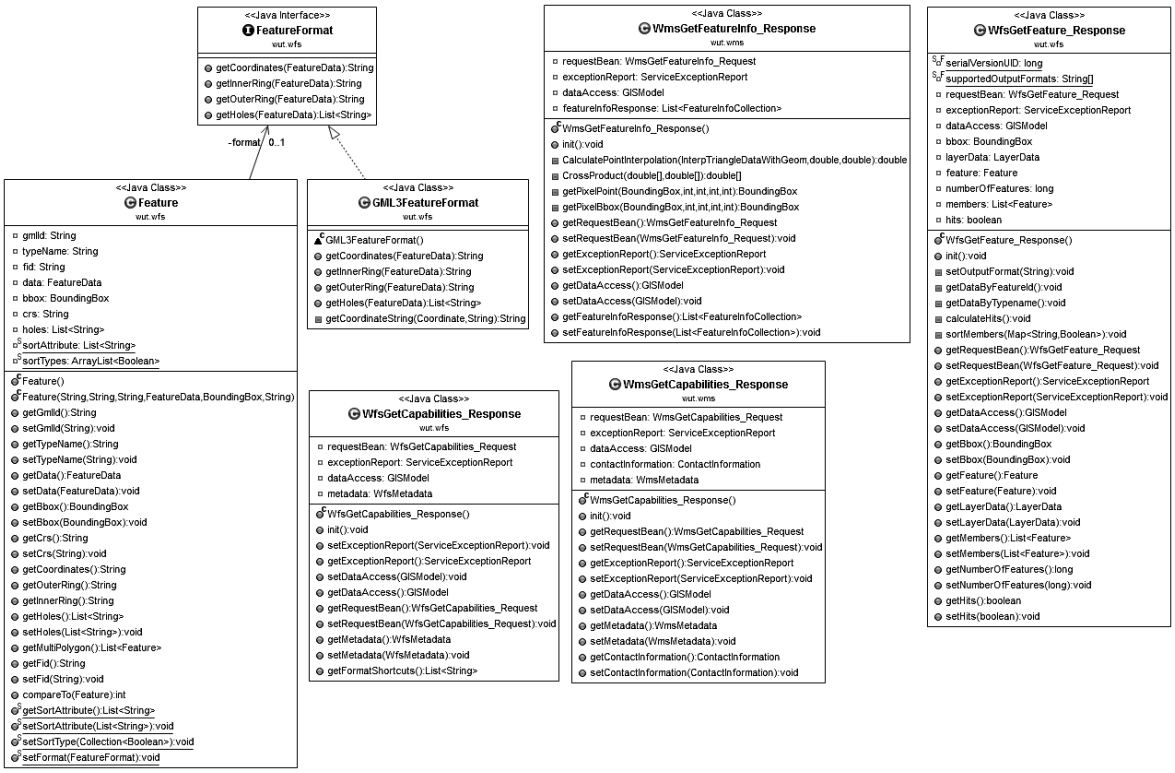
Rysunek 14.1: Diagram klasy GisModel



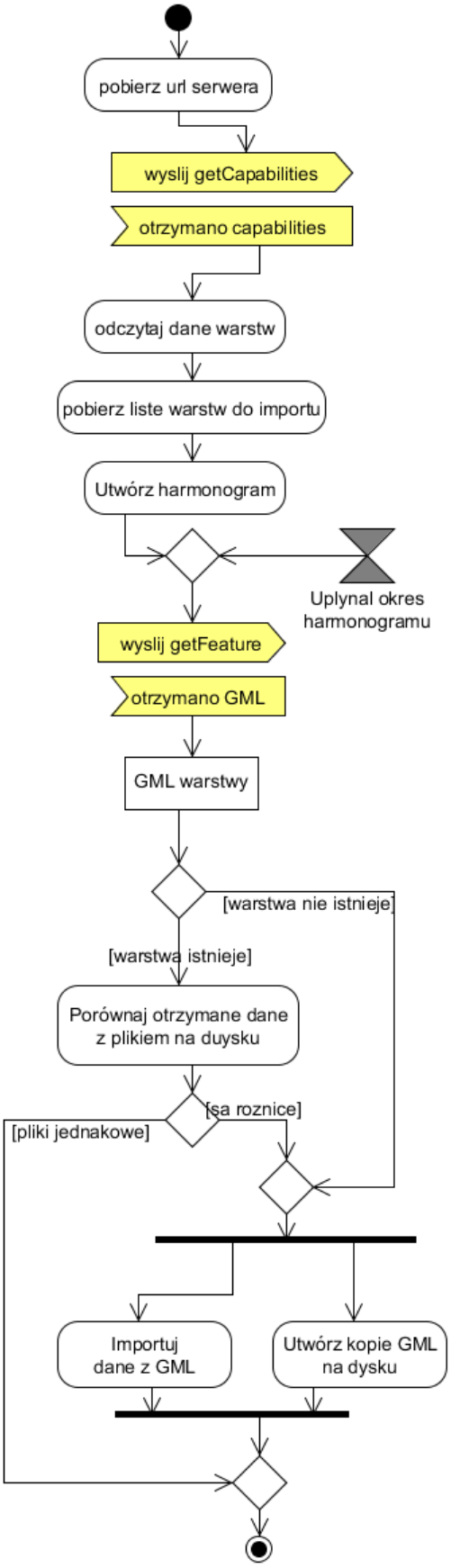
Rysunek 14.2: Diagram klas ServerManager, ReplicationListener i ReplicationInterceptor



Rysunek 14.3: Diagramy klas odpowiedzialnych za renderowanie grafiki



Rysunek 14.4: Diagramy przykładowych Beanów oraz klas Feature i FeatureFormat



Rysunek 14.5: Diagram aktywności dla procesu cyklicznego importu danych z zewnętrznego serwisu WFS

Aneta Rosłan
Nr albumu 211073

Warszawa, 31 stycznia 2013

Oświadczenie

Oświadczam, że moją część pracy inżynierskiej pod tytułem „Geoportal", której promotorem jest mgr inż. Michał Okulewicz wykonałam samodzielnie, co poświadczam własnoręcznym podpisem.

.....

Aneta Rosłan