POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK: Informatyka (INF)
SPECJALNOŚĆ: Inżynieria systemów informatycznych (INS)

PRACA DYPLOMOWA MAGISTERSKA

Analiza porównawcza frameworków internetowych w języku Ruby w zastosowaniach GISowych

Comparative analysis of Ruby's web frameworks for Geographic Information Systems

AUTOR:
Mikołaj Grygiel

PROWADZĄCY PRACĘ:
dr inż. Roman Ptak

OCENA PRACY:

WROCŁAW 2017

Spis treści

Rozdział 1

Wprowadzenie

1.1 Cel pracy

Język Ruby zajmuję 12 miejsce w rankingu popularności języków programowania *Tiobe*[1]. Dużą popularnością wśród frameworków internetowych cieszy się Ruby on Rails, w rankingu *Hotframeworks*[2] zajmuję 3 miejsce wśród wszystkich frameworków. Ruby on Rails jest bez wątpienia najpopularniejszym frameworkiem w języku Ruby, kolejne dwa frameworki w języku Ruby to Sinatra i Hanami, zajmują w wcześniej przytoczonym rankingu odpowiednio miejsca 25. i 73. Jednak w języku Ruby istnieje kilkanaście frameworków przeznaczonych do budowania aplikacji internetowych.

Celem niniejszej pracy jest poznanie wybranych frameworków w języku Ruby, ich porównanie w konkretnym zastosowaniu jakim są systemy informacji geograficznej oraz odpowiedź na pytanie jaki framework najlepiej wybrać do tworzenia systemu GIS.

1.2 Zakres i koncepcja pracy

"Framework"można zdefiniować jako szkielet służący do budowania aplikacji, czyli zbiór gotowych rozwiązań powtarzających się problemów i wzór do budowania nowych funkcjonalności.[19]

W niniejszej pracy zostaną omówione wybrane frameworki w języku Ruby w świetle ich użyteczności przy budowie systemu informacji geograficznej. Frameworki zostaną porównane na podstawie informacji zawartych w dostępnej dokumentacji narzędzia oraz zaimplementowanej przykładowej aplikacji, za pomocą każdego z wybranych narzędzi, spełniającej wymagania systemu GIS.

[1]Dane z marca 2017 roku dostępne na stronie https://www.tiobe.com/tiobe-index/

[2]Ranking https://hotframeworks.com bierzę pod uwagę liczba repozytoriów kodu na platformie Github i ilość tematów na forum Stackoverflow dotyczących danego frameworku. Dane z dnia 26.03.2017 r.

Rozdział 2

Podstawy teorytyczne

2.1 Charakterystyka Systemów Informacji Geograficznej

System Informacji Geograficznej skrótowo nazywany GIS (ang. Geographic Information System) można zdefiniować na wiele sposobów. Michael Schmandt w swoim opracowaniu[18] podaje następujące definicje:

Definicja 1. GIS jest to system komputerowym składającym się z sprzętu i oprogramowania oraz ludzie, którzy wspomagają zbieranie, zarządzanie, analizowanie i wyświetlanie danych przestrzennych. Stosując tą definicje możemy podzielić system GIS na 4 moduły:

- moduł wprowadzania danych - zawiera narzędzia pozwalające na wprowadzania i przechowywanie danych przestrzennych.
- moduł zarządzania danymi - ta część umożliwia edytowanie oraz przeglądanie zgromadzonych zbiorów danych
- moduł analizowania danych - podsystem, który odpowiada za analizowanie danych geograficznych i wyciągania z nich informacji
- moduł prezentowania danych - pozwala na tworzenie map, modeli, statystyk ilustrujących zgromadzone dane

Definicja 2. System informacji geograficznej to system komputerowy, który pozwala na przechowywanie danych powiązanych ze sobą geograficznie.

Definicja 3. GIS to narzędzie do wyszukiwania wzorców geograficznych(przestrzennych) w zbiorach danych.

Pierwsza definicja jest najbardziej szeroka i zawiera w sobie dwie następne - definicja druga to dwa pierwsze moduły z **Definicja 1.**, a definicja trzecia to moduł analizowania danych.

W podobny sposób do definicji nr 1 GIS jest zdefiniowany w *Principles of Geographic Information Systems*[15] jako zbiór narzędzi pozwalających operować na danych reprezentujących zjawiska geograficzne. Zbiór ten dzieli się na 4 grupy ze względu na funkcje:

- zbieranie i przygotowywanie danych

- zarządzanie i przechowywanie danych
- analiza danych
- prezentowanie danych

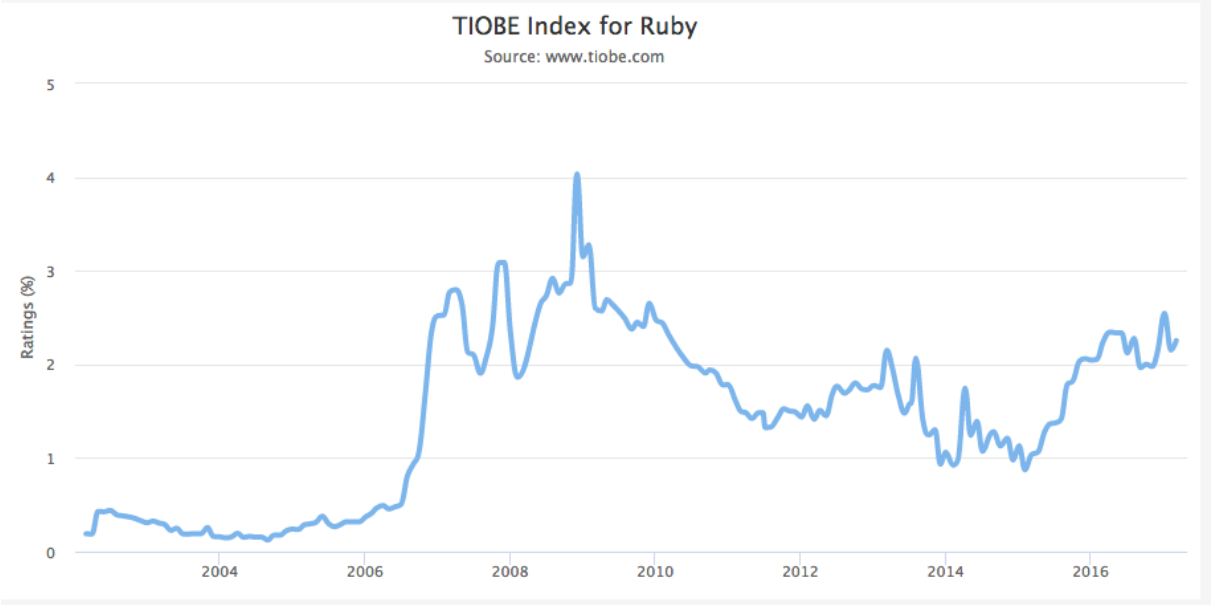
W poniższej pracy przyjmuje się pierwszą definicje Systemu Informacji Geograficznego - GIS to system informatyczny służący do wprowadzania, przechowywania, zarządzania, analizowania i prezentacji danych przestrzennych.

2.2 Charakterystyka języka Ruby

Język Ruby został wydany w 1995 roku. Twórca Rubiego, Yukihiro "Matz" Matsumoto, inspirował się takimi językami programowani jak Perl, Smalltalk, Eiffel, Ada i Lisp by stworzyć jego zdaniem język, który zbalansuje programowanie funkcjonalne z programowanie imperatywnym[6]. Składnia Rubiego ma przypominać język naturalny, autor języka opisuje go jako: *Ruby jest prosty z wyglądu, ale bardzo skomplikowany w środku, tak jak ciało ludzkie.*[1]

Ruby jest językiem ściśle obiektowym, wszystko postrzegane jest jako obiekt. Każda funkcja jest metodą, ponieważ musi być przyłączona do jakiegoś obiektu. Ruby posiada celowo tylko jednokrotne dziedziczenie, ale pozwala na dołączanie wielu modułów, które są zbiorami metod do klasy. Ruby jest bardzo elastycznym językiem, pozwala na usunięcie lub przedefiniowanie dowolnej swojej części. Mimo silnie obiektowej natury, dostępne są również elementy programowania funkcyjnego takie jak funkcje anonimowe lub domknięcia.

Szerszą popularność Ruby zyskał w 2006r., 11 lat po publikacji. Swoją popularność zawdzięcza głównie frameworkowi Ruby on Rails. w rankingu popularności języków programowania Tiobe znajduje się aktualnie na 12 miejscu[2].



Rysunek 2.1 Historia popularności języka Ruby według rankingu Tiobe

[1] wypowiedź w liście ruby-talk 12.05.2000 r., źródło: http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-talk/2773

[2] dane z dnia 08.04.2016 r. https://www.tiobe.com/tiobe-index/

Rozdział 3

Istniejące systemy GIS w języku Ruby

3.1 OpenStreetMap

OpenStreetMap jest internetowym systemem informacji geograficznej z otwartym kodem źródłowym. System zbudowany z wykorzystaniem bazy danych PostgreSQL, frameworku Ruby on Rails oraz biblioteki javascriptowej Leaflet służącej do tworzenia interaktywnych widoków z mapami. Dostęp do danych jest otwarty, dane mogą być edytowane przez dowolnego użytkownika, dlatego mogą być niezgodne z rzeczywistością. OpenStreetMap definiuje 4 typy obiektów przestrzennych[9]:

- Węzeł (ang. node) - pojedynczy punkt geoprzestrzenny reprezentowany przez długość i szerokość geograficzną.
- Linia (ang. way) - jest to uporządkowany zbiór punktów, które mogą reprezentować funkcje liniowe(wektory) lub obszary.
- Relacja (ang. relation) - składa się z uporządkowanej listy węzłów, linii i innych relacji.
- Tag (ang. tag) - to jednostka informacji dołączona do obiektu jednego z powyżej opisanych typów. Tag składa się z klucza oraz wartości.

OpenStreetMap można wykorzystać przez utworzenie komponentu HTML z wybraną mapą, gotowego do zamieszczenia na dowolnej stronie internetowej lub przez pobranie danych z wybranej mapy. Skompresowane aktualne dane dla całej planety z pojedynczego dnia zajmują prawie 40GB. Można pobierać również dane historyczne.

3.2 MangoMap

MangoMap jest komercyjnym narzędziem do tworzenia map dostępnych przez internet z własnych danych geoprzestrzennych. Ceny za korzystanie z serwisu wynoszą 49-399$ miesięcznie w zależności od liczby map i udostępnianego miejsca na serwerze do przechowywania danych. System zbudowany jest w oparciu o framework Ruby on Rails. Mapy tworzy się przy użyciu interfejsu graficznego. Stworzone mapy mogą być udostępnione na serwerze MangoMap przez unikalny link lub zamieszczone na zewnętrznej stronie WWW przez komponent HTML[7].

Rozdział 4

Frameworki internetowe w języku Ruby

W języku Ruby istnieje kilkanaście wspieranych frameworków internetowych. Wybór wykorzystanych frameworków w niniejszej pracy dokonano w następujący sposób:

1. Podzielono frameworki według daty opublikowania pierwszej wersji na 3 grupy:
   (a) opublikowane w latach 2004 - 2011 - frameworki o ugruntowanej pozycji
   (b) opublikowane w latach 2012 - 2015 - stosunkowo nowe frameworki
   (c) opublikowane w latach 2016 - 2017 - najnowsze frameworki
2. z każdej grupy wybrano framework z największą ilością pobrań.

Ta metoda ma na celu wyłonienie najpopularniejszych frameworków, które powstały w różnych etapach języka Ruby, jednocześnie każdy z nich współpracuje z najnowszą wersją języka. w ten sposób wybrano *Ruby on Rails, Trailblazer i Hanami.*

Tablica 4.1 Frameworki internetowe w języku Ruby [1]

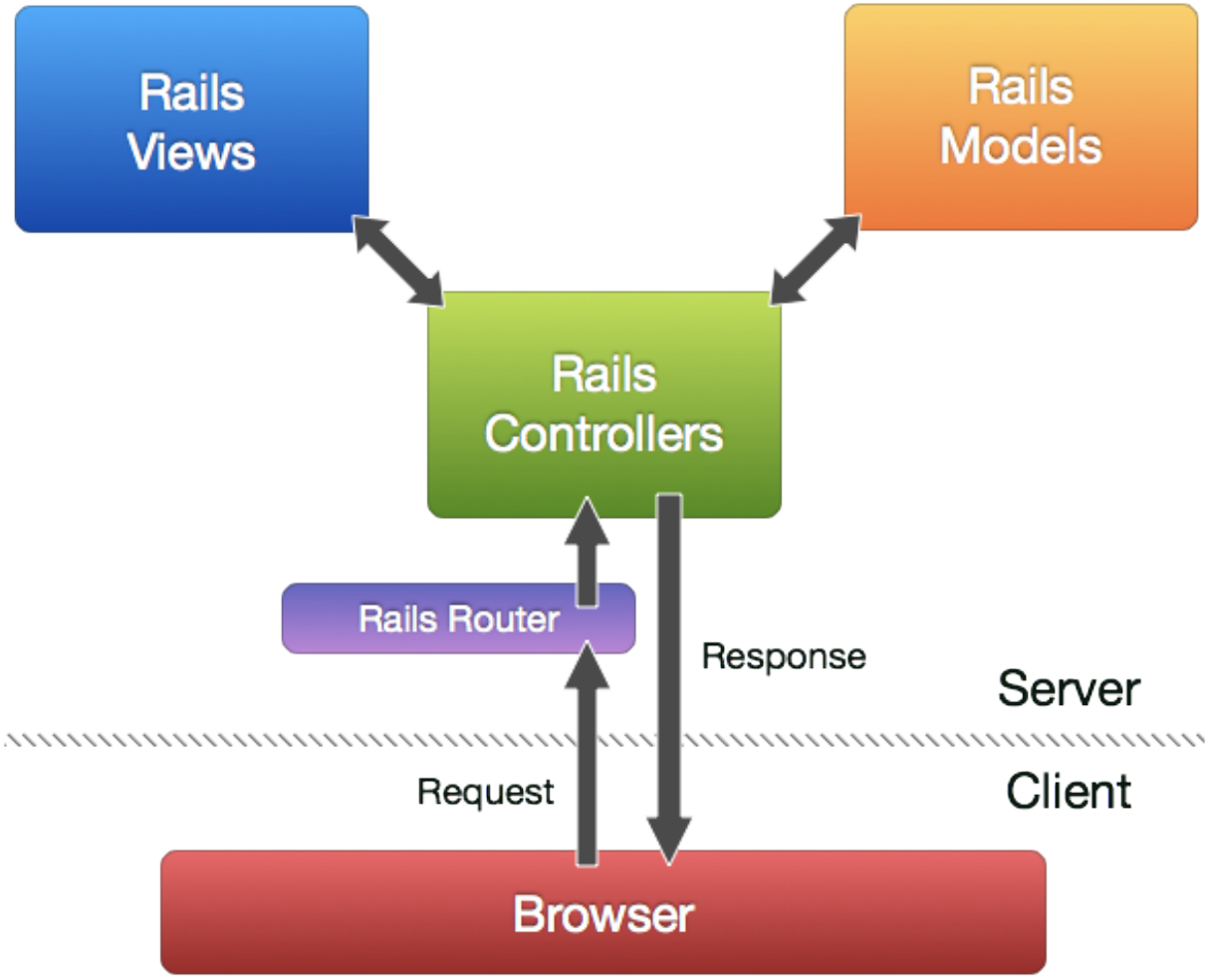
Nazwa	Data opublikowania pierwszej wersji	Data opublikowania najnowszej wersji	Ilość pobrań
Ruby on Rails	25.10.2004 r.	20.03.2017 r.	91 898 706
Hobo	29.04.2007 r.	07.05.2016 r.	211 042
Sinatra	04.10.2007 r.	19.03.2017 r.	45 207 501
Padrino	16.11.2009 r.	23.03.2017 r.	556 481
Cuba	25.04.2010 r.	01.07.2016 r.	124 371
Strelka	24.08.2011 r.	19.01.2017 r.	57 007
Pakyow	20.09.2011 r.	15.07.2016 r.	18 222
Scorched	03.03.2013 r.	12.10.2016 r.	26 929
Trailblazer	26.07.2013 r.	23.01.2017 r.	96 217
Roda	20.07.2014 r.	15.03.2017 r.	105 103
Vanilla	09.05.2015 r.	05.07.2016 r.	80 125
Hanami	20.01.2016 r.	06.04.2017 r.	28 885
Dry-web	21.04.2016 r.	02.02.2017 r.	7 190

[1] Zestawienia przygotowano na podstawie danych z https://rubygems.org/ oraz https://www.ruby-toolbox.com. Pominięto frameworki, których ostatnia wersja ukazała się ponad rok temu. Aktualne na dzień 08.04.2017 r.

4.1 Ruby on Rails

Pierwsza stabilna wersja (1.0.0) frameworku Ruby on Rails ukazała się pod koniec 2005 roku, aktualna wersja (5.0.2) została opublikowana 02.03.2017 r. Aplikacja zbudowana z wykorzystanie RoR jest oparta o wzorzec projektowy Model-Widok-Kontroler[17] (ang. Model-View-Controller). Aplikacja oparta na tym wzorcu jest podzielona na 3 części:

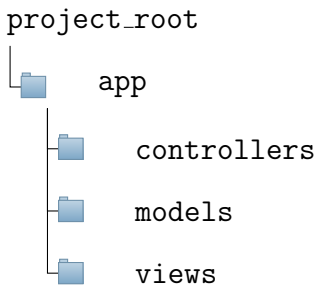
- Modele (ang. model) - reprezentują logikę biznesową. w tej warstwie znajdują się wszelkie obiekty, które służą do wykonywania operacji związanych z implementacją funkcjonalności aplikacji.
- Widoki (ang. view) - służą do prezentowania danych.
- Kontrolery (ang. controller) - obsługują zapytania użytkownika. Nie zawierają w sobie żadnej logiki biznesowej.



Rysunek 4.1 Architektura aplikacji w Ruby on Rails, źródło: http://blog.ifuturz.com/ruby-on-rails/ruby-on-rails-mvc-learn-with-fun.html

Dwie główne zasady frameworku[12]:

- Nie powtarzaj się (ang. Don't Repeat Yourself) - każda informacja powinna mieć pojedynczą, jednoznaczną i autorytatywną reprezentacje w kodzie źródłowym systemu. Ułatwia to utrzymywanie kodu.
- Konwencja ponad konfiguracje (ang. Convention Over Configuration) - RoR posiada ustalone zasady postępowania w różnych przypadkach. Aplikacja domyślnie zachowuje się według tych zasad, nie wymagając dodatkowej konfiguracji. Pozwala to zredukować ilość kodu źródłowego.



Rysunek 4.2 Podstawowa struktura projektu Ruby on Rails

4.2 Roda

Twórcy frameworku Roda jest przy tworzeniu narzędzia podstawili sobie 4 cele[13]:

- Prostota (ang. simplicity) - framework ma być prosty zarówno wewenątrz(w implementacji), jak i na zewnątrz (dla użytkowników).
- Niezawodność (ang. reliability) - Roda wspiera i promuje projektowanie aplikacji z niemutowalnym stanem. Aplikacje zbudowane za pomocą Rody, są zaprojektowane tak. Roda ogranicza zmienne, stałe i metody przypisane do instancji obiektów aby uniknąć konfliktów z kodem zaimplementowanym przez użytkownika.
- Rozszerzalność (ang. extensibility) - framework zbudowany jest całkowicie w oparciu o wtyczki, aby ułatwić dodawanie funkcjonalności do frameworka. Każda część Rody może być zastąpiona własną implementacją przez użytkownika.
- Wydajność (ang. performance) - Roda posiada mały koszt obsługi zapytań, drzewo trasowania i inteligentną obsługę pamięci podręcznej co sprawia, że Roda jest szybsza od innych popularnych frameworków języka Ruby.

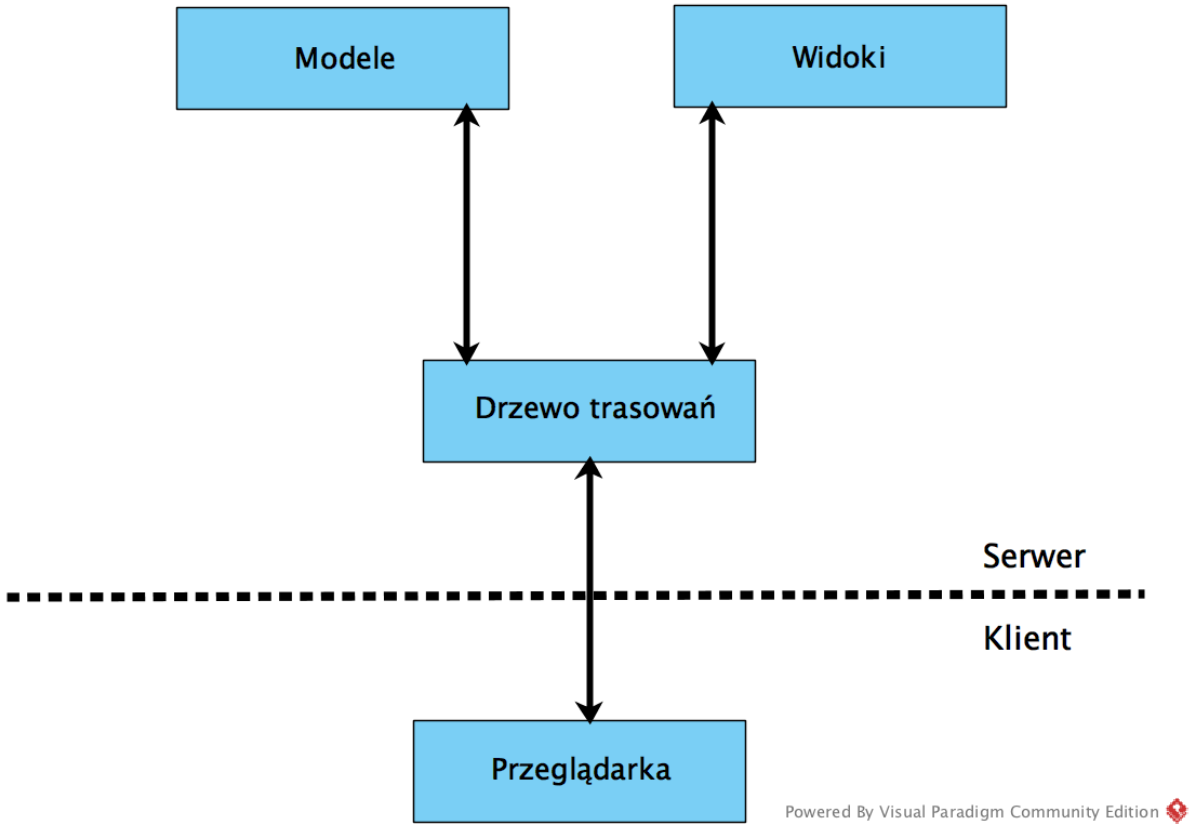
Roda opiera się o drzewo trasowania, punkty dostępu aplikacji zdefiniowane są w strukturze drzewa. Przykład drzewa trasowań znajduje się we fragmencie kodu 4.1.

Fragment kodu 4.1 Proste drzewo trasowań

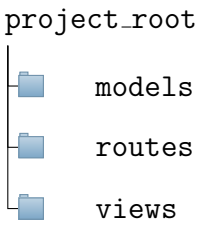
```
r.on "a" do             # /a gałęź
  r.on "b" do           # /a/b gałęź
    r.is "c" do         # /a/b/c zapytanie
      r.get do end      # GET  /a/b/c zapytanie
      r.post do end     # POST /a/b/c zapytanie
```

```
    end
    r.get "d" do end  # GET  /a/b/d zapytanie
    r.post "e" do end # POST /a/b/e zapytanie
  end
end
```

W przeciwieństwie do Ruby on Rails, Roda nie posiada warstwy kontrolerów i osobnego modułu obsługującego trasowanie, w Rodzie przy definicji danego punktu końcowego znajduje się od razu kod obsługujący otrzymane zapytanie.



Rysunek 4.3 Architektura aplikacji w frameworku Roda



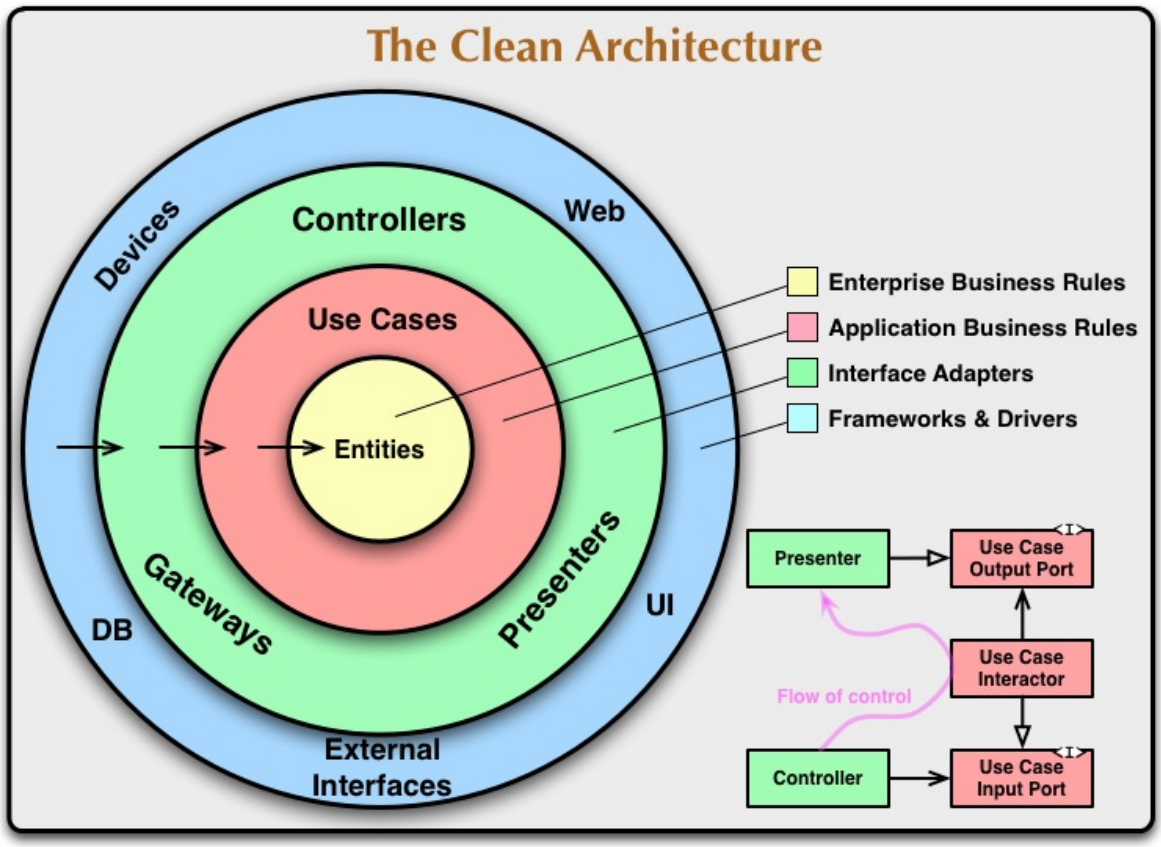
Rysunek 4.4 Struktura projektu Roda

4.3 Hanami

Hanami jest lekkim frameworkiem internetowym opartym o architekturę MVC, zbudowanym z wielu mikro-bibliotek. w przeciwieństwie do Ruby on Rails, Hanami nie stawia konwencji ponad konfiguracje, wszystkie informacje powinny być zawarte w napisanym

przez programistę kodzie, którego zrozumienie nie wymaga znajomości konwencji frameworka. Architekura projektu jest głównie inspirowana przez te dwa podejścia:

- Czysta architektura (ang. clean architecture) - schemat tej architektury składa się z kolejnych zawierających się w sobie kół. Każde wewnętrzne koło nie ma żadnych zależności w zewnętrznym kole, czyli w obiektach należących do wewnętrznego koła, nie ma odwołań do obiektów z zewnętrznych kół[16]. To podstawowa zasada tego podejścia.



Rysunek 4.5 Schemat czystej architektury[16]

Na schemacie 4.5 można wyróżnić 4 byty:

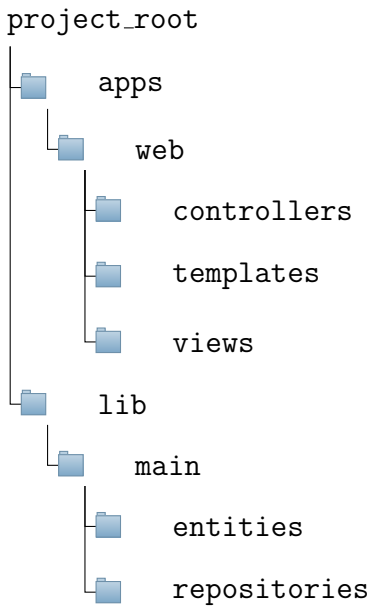
  - Encje (ang. entities) - zawierają ogólne reguły biznesowe systemu. Mogą być reprezentowane przez obiekty z metodami, struktury danych lub pojedyncze funkcje.
  - Przypadki użycia (ang. use cases) - ta warstwa skupia w sobie wszystkie możliwe przypadki użycia systemu przez użytkownika. Znajduje się tu cała logika biznesowa zarządzania encjami. Zmiana w tej warstwie nie powinna wpływać na encje, interfejs użytkownika lub bazę danych.
  - Adaptery interfejsów (ang. interface adapters) - na tym poziomie dane są konwertowane z formatu najbardziej dogodnego dla przypadków użycia i encji do formatu przyjmowanego przez zewnętrzne narzędzia takie jak baza danych lub przeglądarka internetowa.

  - Frameworki oraz narzędzia (ang. frameworks and drivers) - w tej warstwie znajdują się zewnętrzne narzędzia takie jak inne frameworki, baza danych lub przeglądarka internetowa.

- Najpierw monolit (ang. monolith first) - według tej zasady budowę systemu zaczyna się od monolitycznej aplikacji. Jednak budowa aplikacji od samego początku powinna być jak najbardziej modularna aby można było ją później rozbić na wiele mniejszych aplikacji.

  Zgodnie z zasadą *czystej architektury* w utworzonym projekcie systemu z użyciem Hanami oddzielona jest warstwa logiki biznesowej znajdująca się w folderze *lib* projektu od mechanizmu komunikacji z innymi serwisami zawartego w folderze *apps* .

  Hanami posiada kontener aplikacji, który pozwala nam utworzyć wiele aplikacji w ramach jednego projektu, które korzystają z tego samego zbioru encji i przypadków użycia, a następnie uruchomić je w jednym procesie Rubiego.



Rysunek 4.6 Podstawowa struktura projektu Hanami

Rozdział 5

Narzędzia dostępne do przetwarzania danych geograficznych w języku Ruby

5.1 Przechowywanie danych

Omawiane frameworki domyślnie wykorzystują relacyjną bazę danych do przechowywania informacji. Najpopularniejsze bazy danych to PostgreSQL, MySQL, SQLite.

5.1.1 PostgreSQL

PostgreSQL posiada kilka wbudowanych typów przestrzennych[10]:

- Point - reprezentuje punkt na mapie, to podstawowy typ, który służy do budowania pozostałych typów przestrzennych. Dane zapisane są jako para współrzędnych w postaci tekstowej *(x, y)*, gdzie x i y to liczby zmienno przecinkowe.
- Line - prosta w znaczeniu geometrycznym, mogą być reprezentowane przez równanie liniowe $Ax + By + C = 0$, wtedy w bazie danych zapisane są współczynniki *A, B, C*. Innym sposobem reprezentacji tego typu są dwa punkty, przez które przechodzi prosta *[(x1 , y1) , (x2 , y2)]* .
- Line Segment - w znaczeniu geometrycznym to odcinek, reprezentowany przez dwa punkty, początek i koniec odcinka *[(x1 , y1) , (x2 , y2)]*.
- Box - to czworokąty zapisane jako dwa przeciwległe wierzchołki *((x1 , y1) , (x2 , y2))*.
- Path - lista połączonych ze sobą punktów, ścieżka. Ścieżka może być otwarta, jeśli pierwszy i ostatni punkt nie są ze sobą połączone, lub zamknięta, jeśli pierwszy i ostatni punkt są ze sobą połączone. Nawiasy kwadratowe reprezentują otwartę ścieżkę *[(x1 , y1) , ... , (xn , yn)]*, zamnkięta ścieżka jest przedstawiona za pomocą okrągłych nawiasów *((x1 , y1) , ... , (xn , yn))*.
- Polygon - to wielokąt zapisany za pomocą listy punktów, reprezentujących kolejne wierzchołki wielokątu *((x1 , y1) , ... , (xn , yn))*.
- Circle - okrąg zapisany przy pomocy środka i promienia ¡ *(x , y) , r* ¿.

PostgreSQL dostarcza niewiele funkcji do wyszukiwania danych. Sprawdzenie zależności między dwoma punktami takich jak np. przecięcie dwóch obiektów lub zawieranie się jednego punktu w drugim nie jest dostępne dla typów *Path* i *Polygon*.

5.1.2 PostGIS

PostGIS jest biblioteką rozszerzającą możliwości PostgreSQL w zakresie przetwarzania danych przestrzennych. Biblioteka jest wspierana przez fundacje *Open Source Geospatial*[11]. Dane są zapisane w formie binarnej jako współrzędne geograficzne lub geometryczne w zależności od wyboru. Postgis zapewnia wsparcie typów zdefiniowanych przez *Open Geospatial Consortium*:

- POINT - pojedynczy punkt na mapie.
- LINESTRING - zbiór połączonych ze sobą punktów reprezentujący ścieżkę.
- POLYGON - zbiór połączonych ze sobą punktów reprezentujący wierzchołki wielokątu.

Open Geospatial Consortium definiuje również kolekcje obiektów powyższych typów:

- MULTIPOINT - kolekcja punktów.
- MULTILINESTRING - kolekcja linii.
- MULTIPOLYGON - kolekcja wielokątów.
- GEOMETRYCOLLECTION - kolekcja dowolnych obiektów geometrycznych.

Dla wszystkich typów dostępny jest taki sam zestaw funkcji, które np. sprawdzają zależności między dwoma obiektami, co jest pomocne przy wyszukiwaniu obiektów.

5.1.3 MySQL Spatial

MySQL Spatial jest rozszerzeniem dla bazy MySQL, które dostarcza wsparcie dla danych przestrzennych. Rozszerzenie dostarcza typy danych zdefiniowane przez grupę *Open Geospatial Consortium*[8]. Zbiór funkcjonalności pokrywa się z funkcjonalnościami rozszerzenia Postgis.

5.1.4 SpatiaLite

SpatiaLite jest rozszerzeniem dla bazy danych SQLite, które zawiera wsparcie dla typów przestrzennych zdefiniowanych przez grupę *Open Geospatial Consortium*[14]: *Point, Line, Multiline, Polygon, Multipolygon*. Cała baza danych zapisana jest w pojedynczym pliku. Biblioteka udostępniona jest na zasadzie otwartego źródła.

5.2 Obiektowe przetwarzanie danych

5.2.1 RGeo

RGeo wspiera typy danych zdefiniowane przez *Open Geospatial Consortium*, takie jak punkt, linia i wielokąt. Dane są reprezentowane w formie obiektowej. Biblioteka pozwala

na podstawowe operacje analizowania danych przestrzennych, takie jak szukanie punktów przecięcia i obliczenia pola powierzchni. Biblioteka operuje na reprezentacji geograficznej i gemoetrycznej danych i pozwala konwertować dane pomiędzy dostępnymi reprezentacjami. RGeo udostępnia udostępnia jedynie interfejs w języku Ruby dla programistów, a do obsługi danych geograficznej korzysta z bibliotek GEOS i proj.4, te biblioteki są napisane w języku C++.

5.2.2 GeoRuby

GeoRuby jest biblioteką całkowicie napisaną za pomocą języka Ruby do przetwarzania danych przestrzennych. Biblioteka ściśle podąża za modelem danych definiowanych przez *Open Geospatial Consortium*, dlatego dobrze łączy się z takimi bazami danych jak Postgis, Mysql Spatial, SpatiaLite i innymi, które również wspierają typu opracowane przez OGC[1].

5.3 Prezentowanie danych

5.3.1 Leaflet

Leaflet jest biblioteką napisaną w języku Javascript, która pozwala dodać interaktywną mapę do widoku aplikacji internetowej. Biblioteka nie dostarcza graficznej reprezentacji mapy, ale zapewnia obsługę mapy z zewnętrznego serwisu np. OpenStreetMap. Do mapy można dodawać interaktywne obiekty. Biblioteka dostępna jest na zasadzie otwartego źródła.[3]

5.3.2 GoogleMaps JavaScript API

GoogleMaps JavaScript API pozwala dołączyć mapę z serwisu Google Maps. Mapa jest interaktywna i pozwala wyświetlać obiekty dodane przez programistę. W darmowejwersji GoogleMaps API pozwala wczytać mapę 25 000 razy w ciągu dnia.[2]

Bibliografia

[1] *Dokumentacja biblioteki GeoRuby*, dostępna pod adresem: `https://github.com/nofxx/georuby`, aktualne na dzień 15.06.2017r.

[2] *Dokumentacja biblioteki Google Maps JavaScript API*, dostępna pod adresem: `https://developers.google.com/maps/documentation/javascript/`, aktualne na dzień 15.06.2017r.

[3] *Dokumentacja biblioteki Leaflet*, dostępna pod adresem: `http://leafletjs.com/reference-1.0.3.html`, aktualne na dzień 15.06.2017r.

[4] *Dokumentacja biblioteki Rgeo*, dostępna pod adresem: `https://github.com/rgeo/rgeo`, aktualne na dzień 15.06.2017r.

[5] *Dokumentacja Hanami*, dostępna pod adresem: `http://hanamirb.org/guides/`, aktualne na dzień 08.03.2017r.

[6] *Dokumentacja języka Ruby*, dostępna pod adresem: `https://www.ruby-lang.org/pl/documentation/`, aktualne na dzień 08.03.2017r.

[7] *Dokumentacja MangoMap*, dostępna pod adresem: `http://help.mangomap.com/`, aktualne na dzień 22.04.2017r.

[8] *Dokumentacja MySQL*, dostępna pod adresem: `https://dev.mysql.com/doc/refman/5.7/en/`, aktualne na dzień 15.06.2017r.

[9] *Dokumentacja OpenStreetMap*, dostępna pod adresem: `http://wiki.openstreetmap.org/`, aktualne na dzień 08.03.2017r.

[10] *Dokumentacja PostgreSQL*, dostępna pod adresem: `https://www.postgresql.org/docs/9.6/static/index.html`, aktualne na dzień 09.06.2017r.

[11] *Dokumentacja PostGIS*, dostępna pod adresem: `http://postgis.net/documentation/`, aktualne na dzień 08.03.2017r.

[12] *Dokumentacja Ruby on Rails*, dostępna pod adresem: `http://guides.rubyonrails.org/`, aktualne na dzień 08.03.2017r.

[13] *Dokumentacja Roda*, dostępna pod adresem: `http://roda.jeremyevans.net/documentation.html`, aktualne na dzień 01.06.2017r.

[14] *Dokumentacja SpatiaLite*, dostępna pod adresem: `https://www.gaia-gis.it/fossil/libspatialite/index`, aktualne na dzień 09.06.2017r.

[15] Huisman Otto, By (de) Rolf A., *Principles of Geographic Information Systems*, ITC, 2009

[16] Martin Robert, *The Clean Architecture*, dostępna pod adresem: `https://8thlight.com/blog/uncle-bob/2012/08/13/the-clean-architecture.html`, aktualne na dzień 20.04.2017r.

[17] Ruby Sam, Thomas Dave, Hansson Heinemeier David, *Agile Web Development with Rails 5*, Pragmatic Programmers, 2016

[18] Schmandt Michael, *GIS Commons: An Introductory Textbook on Geographic Information Systems*, dostępne pod adresem: `http://giscommons.org/`, aktualne na dzień 07.04.2017r.

[19] Smyrdek Przemysław, *Czym jest framework i po co go używać*, dostępne pod adresem: `http://poznajprogramowanie.pl/czym-jest-framework-i-po-co-go-uzywac/`, aktualne na dzień 20.04.2017r.