|  |
| --- |
| **Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie** |
| Opracowanie aplikacji internetowej umożliwiającej interaktywną naukę języka angielskiego |
|  |
| Praca inżynierska |
| Jakub Jaśkowiec |
| Promotor: dr inż. Mirosław Gajer |

Kraków, 2012

Wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki

KATEDRA AUTOMATYKI

Spis treści

[1. Wstęp 4](#_Toc313826387)

[1.1. Cel pracy 4](#_Toc313826388)

[1.2. Opis aplikacji 4](#_Toc313826389)

[1.3. Zawartość pracy 5](#_Toc313826393)

[2. Część teoretyczna 6](#_Toc313826394)

[2.1. Podstawowe pojęcia używane w pracy 6](#_Toc313826395)

[2.2. Technologie użyte w projekcie 6](#_Toc313826399)

[2.2.1. Język programowania - Ruby 6](#_Toc313826400)

[2.2.2. Ruby on Rails 8](#_Toc313826406)

[2.2.3. Gemy 9](#_Toc313826407)

[2.2.4. Javascript 13](#_Toc313826408)

[2.3. Sposoby przechowywania danych 14](#_Toc313826413)

[2.3.1. Baza danych 14](#_Toc313826414)

[2.3.2. Sesja 14](#_Toc313826415)

[2.4. Wykorzystane API 15](#_Toc313826416)

[2.5. Diagramy UML 16](#_Toc313826417)

[2.5.1. Diagram przypadków użycia 16](#_Toc313826418)

[2.5.2. Diagramy klas 17](#_Toc313826427)

[2.6. Proces nauczania 18](#_Toc313826428)

[3. Część praktyczna 20](#_Toc313826429)

[3.1. Widok startowy 20](#_Toc313826430)

[3.1.1. Punkt 1. 20](#_Toc313826435)

[3.1.2. Punkt 2. 21](#_Toc313826436)

[3.1.3. Punkt 3. 21](#_Toc313826437)

[3.1.4. Pasek nawigacyjny 21](#_Toc313826438)

[3.2. Tworzenie nowego słowa 21](#_Toc313826439)

[3.2. Zarządzanie danymi 27](#_Toc313826444)

[3.2.1. Zarządzanie wyrażeniami 27](#_Toc313826450)

[3.2.2. Zarządzanie kolekcjami 29](#_Toc313826451)

[3.3. Tryb nauki 30](#_Toc313826452)

[4. Podsumowanie 35](#_Toc313826456)

[Bibliografia 36](#_Toc313826457)

# Wstęp

## Cel pracy

Niniejsza praca ma na celu zaprezentowanie aplikacji internetowej umożliwiającej interaktywną naukę języka angielskiego oraz szczegółowe omówienie technologii, w jakiej została ona wykonana.

Głównym założeniem programu jest umożliwienie użytkownikowi uczenia się słów oraz zwrotów z języka angielskiego. Nauka powinna przebiegać w taki sposób, aby wymagała jak najmniejszego wysiłku, jednocześnie umożliwiając jak najlepsze zapamiętywanie prezentowanych treści.

## Opis aplikacji

Głównymi funkcjonalnościami aplikacji są:

1. Umożliwienie użytkownikowi tworzenia własnych słów wraz z ich szczegółowym opisem
2. Podział tworzonych słów na kolekcje tematyczne
3. Prezentacja stworzonych kolekcji w taki sposób, aby uczący mógł zapamiętać jak najwięcej słów

Interfejs użytkownika został stworzony w języku angielskim. Dzięki temu używanie programu jest możliwe dla wszystkich użytkowników znających podstawy języka angielskiego pragnących skorzystać z aplikacji w celu poszerzenia swoich umiejętności językowych.

Pomysł na aplikację narodził się podczas aktywności na jednym z forów języka angielskiego. Umieszczone przeze mnie autorskie prezentacje wywołały pozytywny odzew wśród użytkowników. Aplikacja bazuje na wykorzystanym wtedy pomyśle, automatyzując proces tworzenia treści oraz wprowadzając nowe możliwości.



## Zawartość pracy

Pierwsza część pracy to opis oraz analiza technologii użytej w projekcie. Zostanie dokonana analiza technologii Ruby on Rails, przyczyny jej użycia, zalety oraz wady. Przedstawione zostaną także inne technologie użyte w projekcie, używane gemy (pluginy) oraz zewnętrzne API.

Druga część to prezentacja aplikacji. Przedstawione zostaną funkcjonalności oraz scenariusze użycia programu, udokumentowane obrazami w formie zrzutów ekranu. Pokazana zostanie przydatność aplikacji w procesie uczenia się nowych słów oraz innowacyjność wykorzystanych w niej metod.

# Część teoretyczna

## Podstawowe pojęcia używane w pracy

Aplikacja, program, projekt – odnosi się tutaj do aplikacji internetowej, która została stworzona na potrzeby tej pracy

Słowo, wyrażenie – wyraz lub kilka wyrazów, którymi użytkownik może zarządzać przy pomocy aplikacji



## Technologie użyte w projekcie

### Język programowania - Ruby

**Ruby**– jest zorientowanym obiektowo interpretowanym językiem skryptowym stworzonym w 1995 w Japonii przez Yukihiro "Matza" Matsumoto.

Kod źródłowy Rubiegojest kompilowany przez interpreter w momencie wykonania programu. Wiąże się z tym szereg wad i zalet.

Wadą tego typu rozwiązania jest szybkość – ponieważ kod źródłowy jest interpretowany w momencie wykonania oznacza to, że działa wolniej niż odpowiadający mu kod skompilowanej aplikacji. Inną możliwą wadą jest to, że każdy użytkownik używający aplikacji może zobaczyć jej kod źródłowy. Nie jest to problem dla projektów typu Open Source, ale dla niektórych aplikacji komercyjnych może to być efekt niepożądany.

Zaletą języków interpretowanych jest możliwość przenoszenia ich pomiędzy systemami operacyjnymi oraz różnymi architekturami. Skompilowana aplikacja będzie działać jedynie na systemie operacyjnym oraz architekturze na którą została skompilowana. Do uruchomienia aplikacji napisanej w języku Ruby wystarczy zainstalowany interpreter orazkod źródłowy aplikacji.

Kolejną zaletą jest możliwość pisania oraz wykonywania kodu w czasie rzeczywistym przez interpreter. Jest to ważna funkcjonalność dla wielu programistów.

Inne istotne cechy języka:

Ruby jest:

- zorientowany obiektowo

wszystko jest obiektem (włączając typy prymitywne, takie jak liczby całkowite, nil-wartości)

- dynamicznie typowany

sprawdzanie zgodności typów odbywa się w czasie wykonywania programu

Ruby posiada:

- ducktyping

rozpoznawanie obiektu odbywa się nie na podstawie deklaracji typu, lecz poprzez badanie metod udostępnionych przez obiekt

- garbagecollector (odśmiecacz pamięci)

zarządzanie dynamiczną pamięcią wykonywane jest automatycznie, programista nie musi zwalniać przydzielonej pamięci

- domknięcia (closures)

elementy kodu mogą być traktowane jako obiekty

- obsługę wyjątków

obsługa sytuacji wyjątkowych możliwa jest poprzez mechanizm obsługi wyjątków

- system pluginów

dzięki systemowi zarządzania pakietami RubyGems, ułatwione jest zarządzanie zewnętrznymi pluginami

- Interactive Ruby Shell

powłoka, dzięki której możliwe jest programowanie w linii poleceń

- implementację na wszystkich popularnych platformach

Ruby działa m.in. na systemach Windows, Linux oraz Mac.

Aplikacja jest stworzona w języku Ruby w wersji 1.9.2. Jest kompatybilna z wersją Ruby 1.8.7.



### Ruby on Rails

**Ruby on Rails**– jestframeworkiem do tworzenia aplikacji internetowych w języku Ruby. Został stworzony w 2004 roku przez Davida Heinemeier Hanssona. Udostępniony jest na zasadzie licencji Open Source.

Ruby on Rails korzysta z wzorca projektowego Model-View-Controller. Dzięki temu struktura aplikacji jest ściśle zdefiniowana, pozwala to na oddzielenie logiki aplikacji od interfejsu użytkownika. Za poszczególne moduły odpowiadają:

Model – ActiveRecord

Moduł odpowiedzialny za logikę aplikacji oraz komunikację z bazą danych.

Komunikacja z bazą danych jest standardowo realizowana przy pomocy systemu ORM, którym jest ActiveRecord. Framework umożliwia użycie innych systemów ORM.

Kod odpowiedzialny za logikę aplikacji umieszczony jest w plikach reprezentujących obiekty bazodanowe.

Kontroller – ActionController

Kontrolery odpowiedzialne są za sterowanie aplikacją i przepływ danych pomiędzy modelami a widokami.

Przykładowy kod akcji ‘create’, która odpowiada za stworzenie wyrażenia na podstawie danych przesłanych w formularzu:

def create

@expression = Expression.new(params[:expression])

set\_image\_from\_params

if @expression.save

redirect\_to @expression, :notice => 'Expression was successfully created.'

else

render :action => "new"

end

end

Widok – ActionView

Widoki tworzą interfejs użytkownika generowany przy użyciu danych z kontrolera oraz kodu HTML zapisanego w plikach widoków. Pliki widoków zostały stworzone w języku HAML (opis w dalszej części pracy).

Framework dostarcza narzędzia, które ułatwiają programiście podstawowe czynności związane z tworzeniem aplikacji webowych. Jednym z takich narzędzi jest scaffolding, który pozwala na automatyczne generowanie koduumożliwiającego uruchomienie podstawowej aplikacji.

Ruby on Rails posiada zintegrowane środowisko do testowania aplikacji. Domyślnym modułem testującym jest Test/Unit, jednak ze względu na rosnącą popularność narzędzia RSpec (używanego m.in. do Test Driven Development), został on użyty w tej aplikacji.

Do uruchomienia aplikacji wymagany jest odpowiedni serwer. W środowiskach deweloperskich dominuje serwer WEBrick oraz Mongrel,natomiast w środowisku produkcyjnym najpopularniejszym rozwiązaniem jest użycie modułu PhusionPassenger dla serwera Apache lub nginx.

Technologia Ruby on Rails została wybrana z następujących powodów:

- jest nowoczesna - daje to możliwość pracy przy użyciu nowatorskich technologii

- dobrze nadaje się do tego typu aplikacji - wRuby on Rails powstało wiele interaktywnych aplikacji webowych, takich jak Twitter, Groupon czy Github

- duże wsparcie ze strony społeczności - istnieje wiele bibliotek oraz tutoriali ułatwiających pracę programiście

Aplikacja jest stworzona w środowisku Ruby on Rails w wersji 3.0.3.

### Gemy

**Gem** – to spakowana aplikacja lub biblioteka w języku Ruby. Posiada nazwę (np. rake) oraz wersję (np. 0.4.16).

Zarządzanie gemami na komputerze odbywa się przy pomocy polecenia *gem*. Umożliwia ona między innymi instalowanie, usuwanie oraz wyszukiwanie dostępnych pakietów.

RubyGems to system zarządzania gemami.

CechyRubyGems:

- łatwa instalacja oraz usuwanie pakietów i ich zależności

- zarządzanie i kontrola lokalnymi pakietami

- zarządzanie zależnościami

- wyszukiwanie oraz przeglądanie lokalnych i zdalnych pakietów

- możliwość instalacji wielu wersji tych samych pakietów

- interfejs webowy umożliwiający przeglądanie dokumentacji dotyczącej zainstalowanych pakietów

W projekcie wykorzystano szereg gemów, dzięki którym możliwe było rozszerzenie funkcjonalności aplikacji oraz ułatwienie pracy programistycznej:

- Rspec

Jest popularnym narzędziem używanym do Test Driven Development w języku Ruby. Umożliwia tworzenie m.in. testów jednostkowych, funkcjonalnych oraz wydajnościowych.

W niniejszej pracy został użyty zamiast standardowego narzędzia testującego wbudowanego w Ruby on Rails (Test Unit). Zostały w nim stworzone głównie testy funkcjonalne.

Fragment testu sprawdzającego poprawność akcji odpowiedzialnej za wyświetlenie listy słów:

describe "GET index" do

it "assigns all expressions as @records" do

expression = Expression.create! valid\_attributes

get :index

assigns(:records).should eq([expression])

end

end

- Haml

Haml (HTML AbstractionMarkup Language) jest językiem znaczników używanym do prostego i przejrzystego opisywania HTML. Został stworzony, aby obejść wiele niedoskonałości tradycyjnych systemów szablonów używając zarazem eleganckiej składni. Haml umożliwia zastąpienie standardowego systemu szablonów używanych w Ruby on Rails (RHTML).

Główne założenia języka:

- wcięcie reprezentuje zagnieżdżenie (brak tagów zamykających)

- tagi HTML są reprezentowane jako ‘%’ oraz nazwa tagu (np. %div )

- atrybuty HTML zapisywane są w postaci hasha z języka Ruby (np. {:class => ‘klasa’} )

Porównanie składni standardowego systemu szablonów RHTML z HAML:

|  |  |
| --- | --- |
| RHTML | HAML |
| <div id="profile">  <div class="left column">  <div id="date">  <%= print\_date %>  </div>  <div id="address">  <%= current\_user.address %>  </div>  </div>  <div class="right column">  <div id="email">  <%= current\_user.email %></div>  <div id="bio">  <%= current\_user.bio %></div>  </div>  </div> | #profile  .left.column  #date= print\_date  #address  = current\_user.address  .right.column  #email= current\_user.email  #bio= current\_user.bio |

- Active Scaffold

Ułatwia budowę interaktywnego panelu administracyjnego, który służy do zarządzania danymi. Panel w dużej mierze korzysta z technologii AJAX (AsynchronousJavaScript and XML), co czyni go wygodniejszym w użyciu.

Przy jego użyciu została zbudowana większość funkcjonalności opisanych w punktach 3.2.

- Wordnik

Jest to oficjalny gem portalu wordnik.com. Umożliwia dostęp do Wordnik API, które jest wykorzystywane do pobierania danych do opisu słowa. Strona internetowa projektu: <https://github.com/wordnik/wordnik-ruby>

-Paperclip

Wspiera obsługę plików graficznych. Umożliwia zapisywanie na serwerze, skalowanie oraz odczytywanie plików graficznych wybranych przez użytkowników.

- Devise

Umożliwia autentykację użytkowników. Korzysta z modelu User.

- Coffee-rails

Umożliwia używanie języka CoffeScript zamiast Javascriptu. CoffeScript charakteryzuje się poprawioną w stosunku do Javascriptu składnią. Jest to język który kompiluje się do Javascriptu i w takiej formie wykonywany jest po stronie klienta.

- Sass

Jest rozszerzeniem kaskadowych arkuszy stylów CSS. Wprowadza nowe możliwości w definiowaniu stylów, takie jak zmienne, dziedziczenie selektorów czy zagnieżdżone reguły.

Przykładowy kod:

.next{

min-width:110px;

vertical-align:top;

overflow:hidden;

padding-top:100px;

a{

padding: 10px 25px;

padding-bottom:1000px;

padding-top:100px;

&:hover{

background-color: lightgray;

}

background-color: #EEE;

min-width:90px;

color: black;

text-decoration: none;

}

- Mysql

Umożliwia używanie bazy danych MySql w Ruby on Rails. Opis bazy danych został przedstawiony w dalszej części pracy.

### Javascript

Javascript jest językiem skryptowym, często używanym w technologiach webowych w celu zwiększenia interaktywności aplikacji.

Domyślną biblioteką Javascript dla Ruby on Rails jest jQuery. Oferuje ona wiele funkcji, które ułatwiają pracę w języku Javascript.

Dodatkowo, w projekcie został użyty język CoffeeScript – język, który kompiluje się do Javascriptu. CoffeeScript posiada elegancką składnię, która usprawnia pracę z językiem Javascript.

Przykłady użycia:

$('#definitions .definition').live 'click', () ->

textarea = $('textarea#expression\_definition')

textarea.val( textarea.val() + this.innerHTML.replace("\n","") + "\n" )

this.className = this.className + ' inactive'

Powyższy kod korzysta z funkcji ‘live’ dostępnej w bibliotece jQuery. Funkcja umożliwia przypisanie zdarzenia do określonego elementu. Ten fragment kodu odpowiada za zmianę treści pola tekstowego oraz zmianę klasy elementu po kliknięciu w definicję.



## Sposoby przechowywania danych

### Baza danych

Baza danych użyta w projekcie to MySql. Komunikacja z bazą odbywa się przy pomocy systemu ORM (Object-Ralational Mapping) wbudowanego w Ruby on Rails - Active Record.

W bazie danych są przechowywane:

- dane rejestracyjne użytkowników,

- dane o wyrażeniach,

- dane o kolekcjach.

### Sesja

Drugim sposobem przechowywania danych jest mechanizm sesji, wbudowany w Ruby on Rails.

W sesji przechowywane są:

- odpowiedzi użytkownika,

- atrybuty wyświetlane podczas trybu nauki.

## Wykorzystane API

Aplikacja wykorzystuje następujące zewnętrzne API:

- Google Image Search API

Wykorzystywane jest w celu pobrania obrazów reprezentujących słowo.

Strona główna projektu: <http://code.google.com/apis/imagesearch/>

- Wordnik API

API serwisu internetowego Wordnik.com umożliwia pobieranie definicji, przykładów użycia oraz synonimów dla danego słowa.

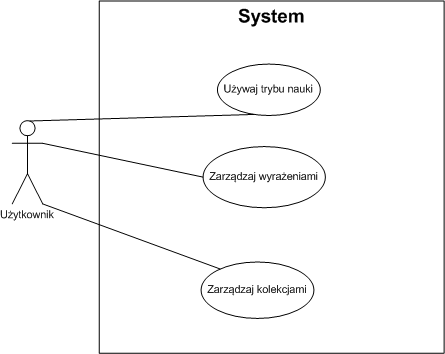
Używanie API umożliwia gem „wordnik”.

Przykładowe użycie (pobieranie definicji):

@definitions = Wordnik.word.get\_definitions(@query, :useCanonical => 'true').map { |d| d["text"]}

## Diagramy UML

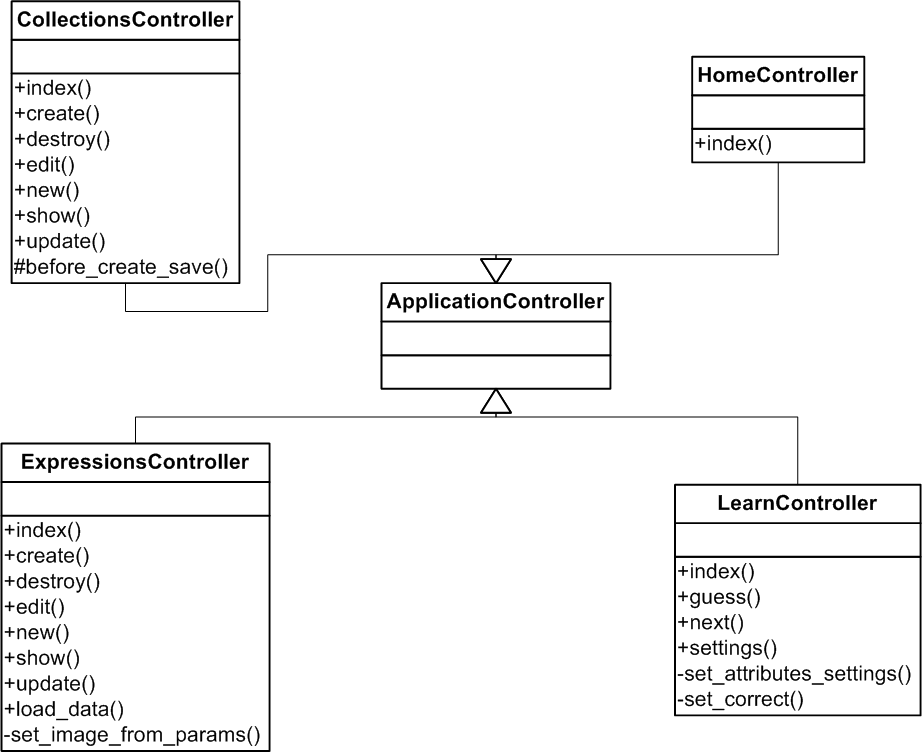
### Diagram przypadków użycia



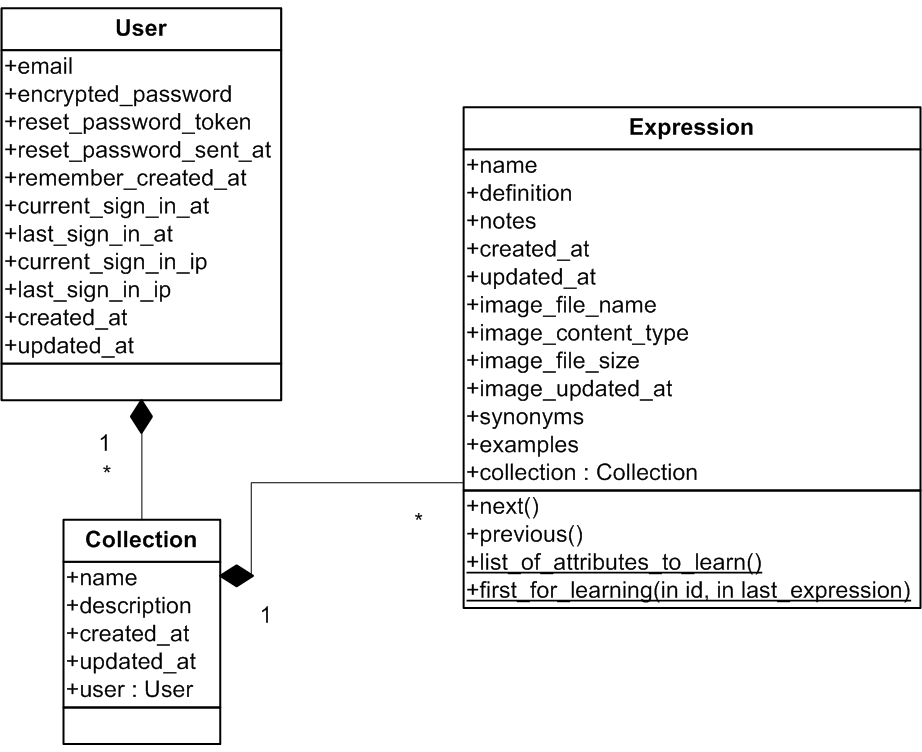
Rysunek - Diagram przypadków użycia



### Diagramy klas



Rysunek 2 - diagram klas - kontrolery



Rysunek 3 - diagram klas - modele

## Proces nauczania

Na innowacyjność procesu nauki składają się następujące czynniki:

1. użytkownik sam generuje treści, których będzie się uczył – pobudza to kreatywność i umożliwia lepsze zapamiętywanie
2. aplikacja korzysta z różnych technik zapamiętywania:

- zapamiętywanie wzrokowe poprzez powiązany z wyrażeniem obraz

- budowanie sieci skojarzeń poprzez synonimy

- zapamiętywanie kontekstu w jakim słowo zostało użyte poprzez przykłady użycia

- dokładny opis wyrażenia w postaci definicji

1. powtarzanie zgromadzonej wiedzy w trybie nauki - umożliwia usystematyzowanie wiedzy i lepsze zapamiętanie wszystkich słów

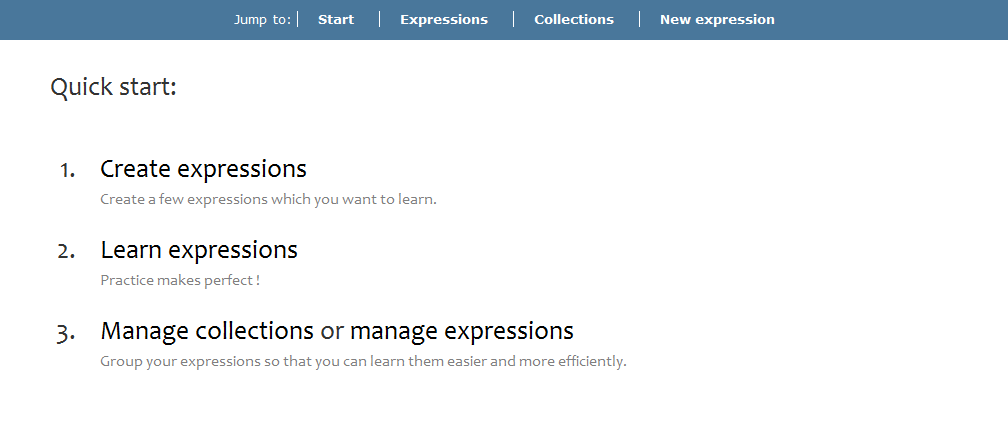
# Część praktyczna

Uwaga: obrazy reprezentujące wyrażenia zostały pobrane przy użyciu Google Image Search API (punkt 2.4).

## Widok startowy

Strona główna (rys. 1) zawiera krótki opis możliwości aplikacji. Składa się on z listy kroków oraz krótkiego opisu.





Rysunek 4 – Strona główna

### Punkt 1.

Umożliwia stworzenie wyrażenia.

Stworzone wyrażenia zostaną wykorzystane w procesie nauki, dlatego jest to element konieczny do dalszej pracy.

Szczegółowy opis w punkcie 3.2.

### Punkt 2.

Umożliwia wejście do trybu nauki wyrażeń.

Szczegółowy opis w punkcie 3.4.

### Punkt 3.

Zarządzanie kolekcjami oraz wyrażeniami.

Dzięki grupowaniu wyrażeń w kolekcje, użytkownik może podzielić swoją naukę na różne grupy tematyczne lub stworzyć kolekcje o różnych poziomach trudności.

Szczegółowy opis w punkcie 3.3.

### Pasek nawigacyjny

Na wszystkich stronach widoczny jest pasek nawigacyjny. Umiejscowiony jest na górze strony, na niebieskim tle. Umożliwia on szybkie przejście do interesującej sekcji .

## Tworzenie nowego słowa

Proces tworzenia nowego słowa został zaprojektowany tak, aby użytkownik na podstawie sugerowanych danych mógł samodzielnie opisać interesujące go słowo. Umożliwia to lepsze zapamiętywanie nowego słownictwa, gdyż pobudza kreatywność oraz pozwala stworzyć własne skojarzenia.

Na opis słowa składają się następujące elementy:

- nazwa,

- obraz,

- definicja,

- przykłady użycia,

- synonimy,

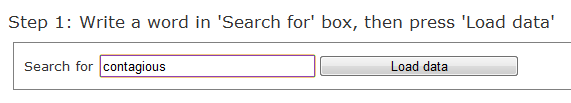
- notatki.

Dodatkowo, każde słowo można umieścić w kolekcji.

Poniżej zostanie omówiony proces tworzenia nowego wyrażenia, na przykładzie słowa ‘contagious’.

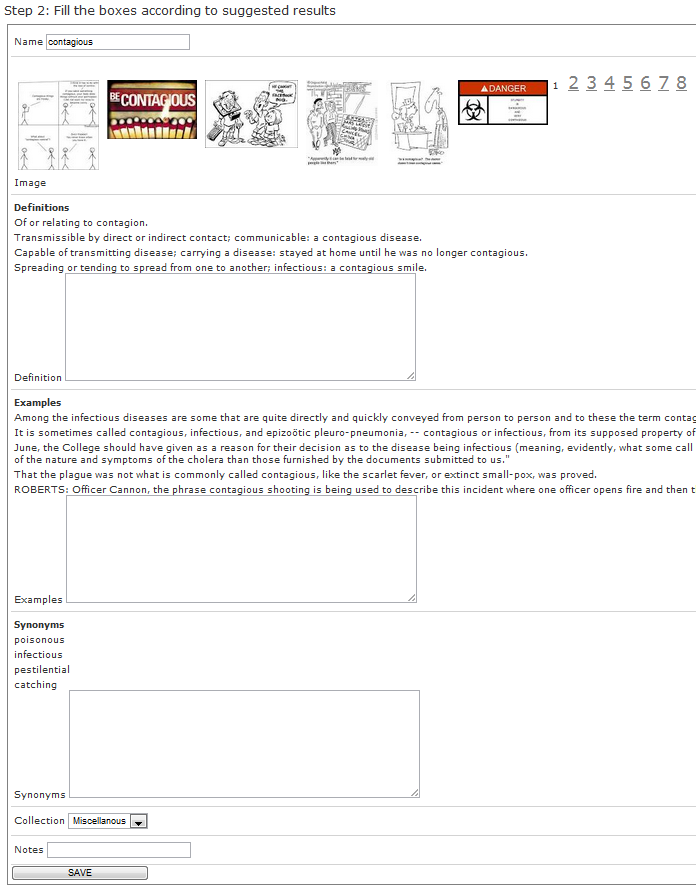
a) w polu ‘Search for’ wpisujemy słowo / frazę

b) klikamy ‘Load data’



Rysunek 5 – wczytywanie danych

Wyniki wyszukiwania zostają załadowane do ramki poniżej. Poszczególne atrybuty zostały oddzielone poziomą linią.



Rysunek 6 - widok załadowanych wyników

Po załadowaniu sugerowanych wyników, zadaniem użytkownika jest wykorzystanie dostępnych danych w celu stworzenia opisu wyrażenia. Atrybuty, które należy wypełnić są następujące:

- Nazwa (Name)

Nazwa, pod jaką słowo zostanie zapisane w systemie. Domyślnie wypełniane jest wyrażeniem wpisanym do pola „Search for”. Po wielokrotnym skorzystaniu z akcji „Load data” wyświetlana jest historia wyszukiwania.



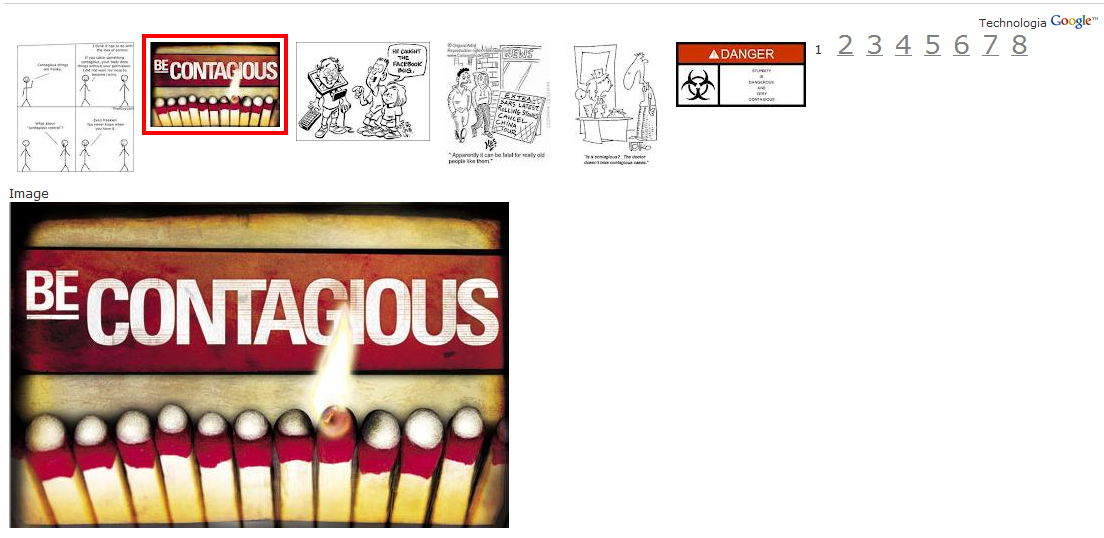
Rysunek 7 - pole z nazwą

- Obraz (Image)

Obraz, który jest dołączany do słowa.

Obrazy pobierane są z wykorzystaniem technologii Google, przy użyciu Google Images API.

Po załadowaniu danych ukazują się miniatury. Po kliknięciu w miniaturę, zostaje ona otoczona czerwoną ramką, a poniżej zostaje załadowany obraz w rozmiarach, w jakich może być zapisany w systemie. Po prawej widzimy szare cyfry, dzięki którym możemy załadować więcej miniatur.

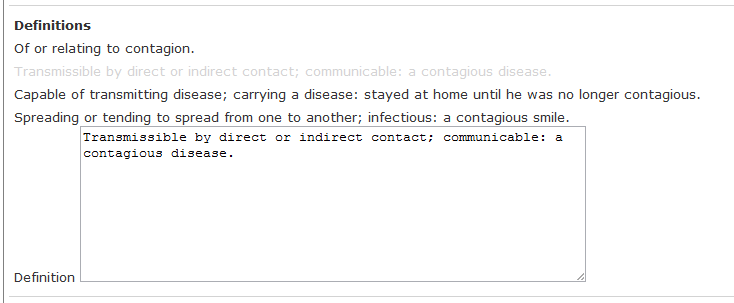


Rysunek 8 –wybór obrazu reprezentującego wyrażenie

- Definicje (Definitions)

Dostępna jest lista definicji do wyboru. Po kliknięciu w definicję, zostaje ona wpisana do pola tekstowego poniżej i oznaczona kolorem szarym.

W systemie jako definicja zostanie zapisana treść pola tekstowego. Użytkownik może dowolnie wybierać definicje z listy: może wybrać jedną lub wiele definicji, może także wpisać swoje własne, jeśli żadna z proponowanych nie jest satysfakcjonująca.



Rysunek 9 –tworzenie definicji

- Przykłady użycia (Examples) / Synonimy (Synonyms)

Ich uzycie jest analogiczne do pola „definicja”

- Kolekcja (Collection)

Przy pomocy listy wyboru możliwe jest dodanie słowa do wybranej kolekcji.

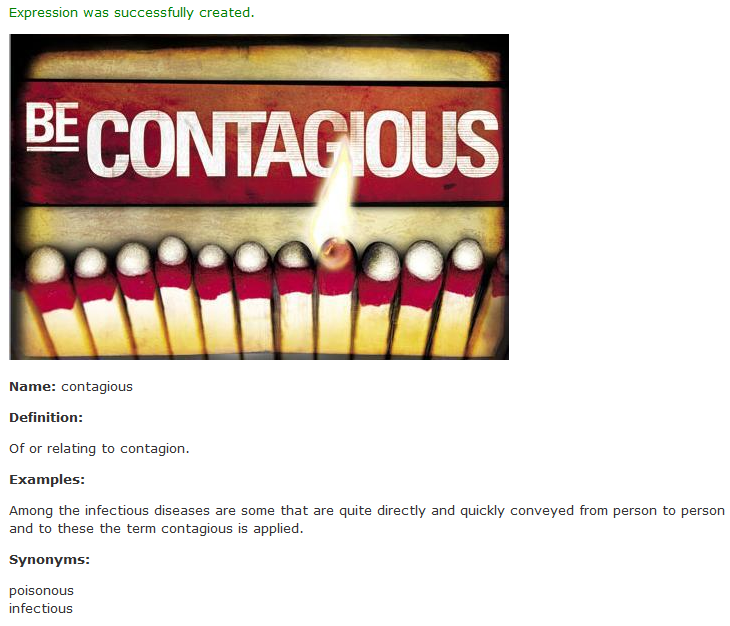


Rysunek 10 - wybór kolekcji

- Notatki (Notes)

Umożliwia dodanie własnych notatek dla słowa.

Po zapisaniu danych, użytkownik zostaje przeniesiony na stronę prezentującą stworzone słowo. Widnieją na niej wszystkie dane, które zostały wybrane podczas procesu tworzenia słowa.



Rysunek 11 – widok końcowy stworzonego wyrażenia



## Zarządzanie danymi

Użytkownicy mogą zarządzać danymi w następujący sposób:



### Zarządzanie wyrażeniami

Z poziomu widoku pokazanego na rysunku 6. możliwe jest przeglądanie, edycja oraz usuwanie stworzonych wyrażeń. Każde wyrażenie zajmuje jeden wiersz.



Rysunek 12 – lista wyrażeń

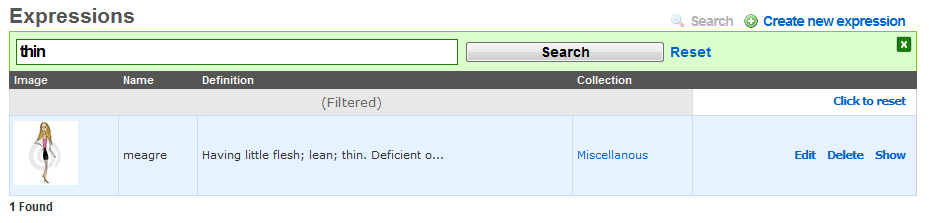
Linki ‘Edit’ oraz ‘Createnewexpression’ prowadzą odpowiednio do edycji oraz tworzenia nowego wyrażenia (opisanego w punkcie 3.1).

Wyrażenia można przeszukiwać – po kliknięciu w ikonę ‘Search’, pojawia się następujący pasek:



Rysunek 13 - wyszukiwanie

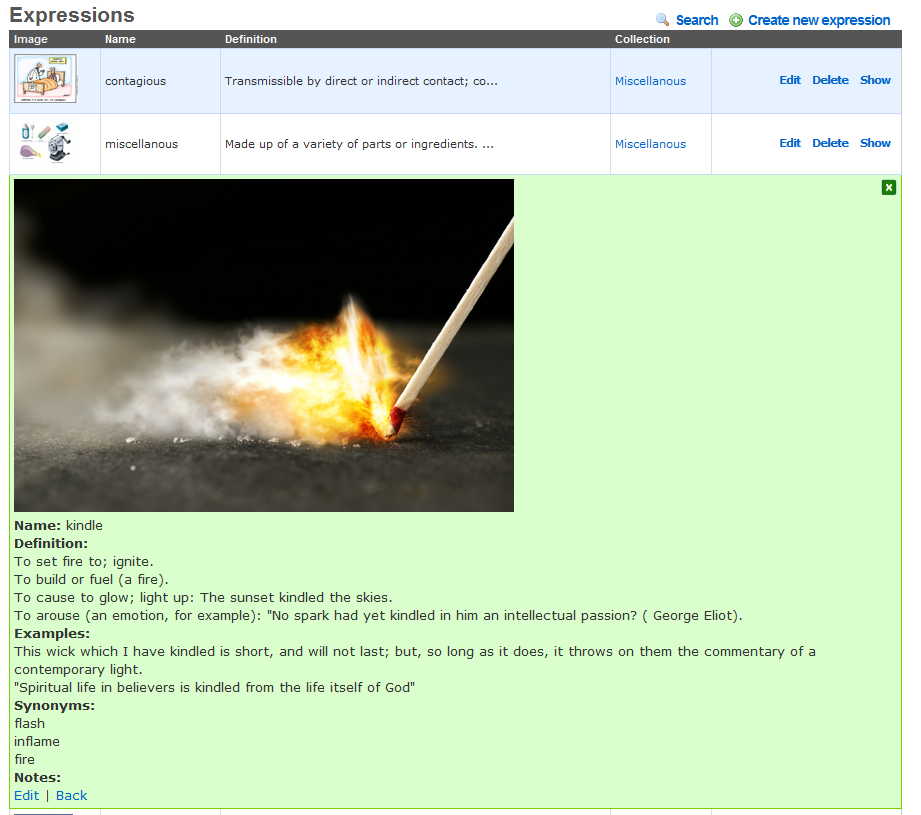
Rezultaty wyszukiwania prezentowane są w następujący sposób:



Rysunek 14 – rezultaty wyszukiwania

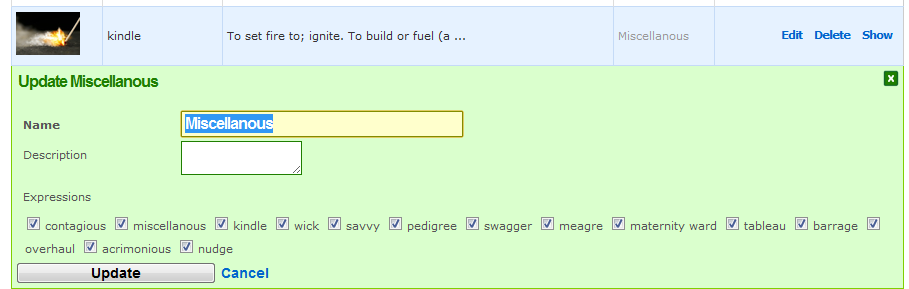
Wyszukiwanie odbywa się nie tylko po nazwie, lecz także po innych elementach tekstowych opisujących słowo.

Po kliknięciu w ‘Show’ zostają wyświetlone wszystkie dane opisujące wyrażenie:



Rysunek 15 – podgląd wyrażenia

Możliwe jest także edytowanie kolekcji. W tym celu należy kliknąć w nazwę kolekcji, do której zostało przyporządkowane wyrażenie.

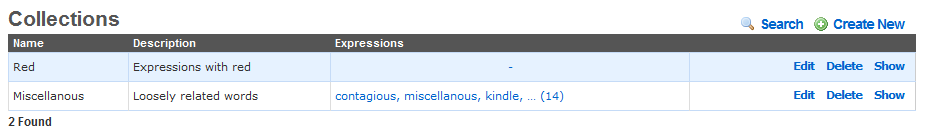


Rysunek 16 – edycja kolekcji

### Zarządzanie kolekcjami

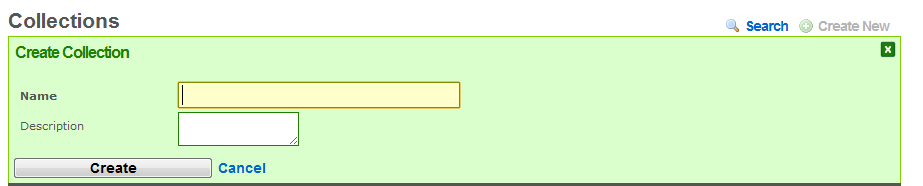
Zarządzanie kolekcjami odbywa się analogicznie do zarządzania wyrażeniami, dlatego pominięte zostaną niektóre elementy wspólne.

Głównym widokiem jest lista kolekcji:



Rysunek 17 – lista kolekcji

Nową kolekcję można stworzyć klikając link ‘Create New’:



Rysunek 18 – tworzenie nowej kolekcji

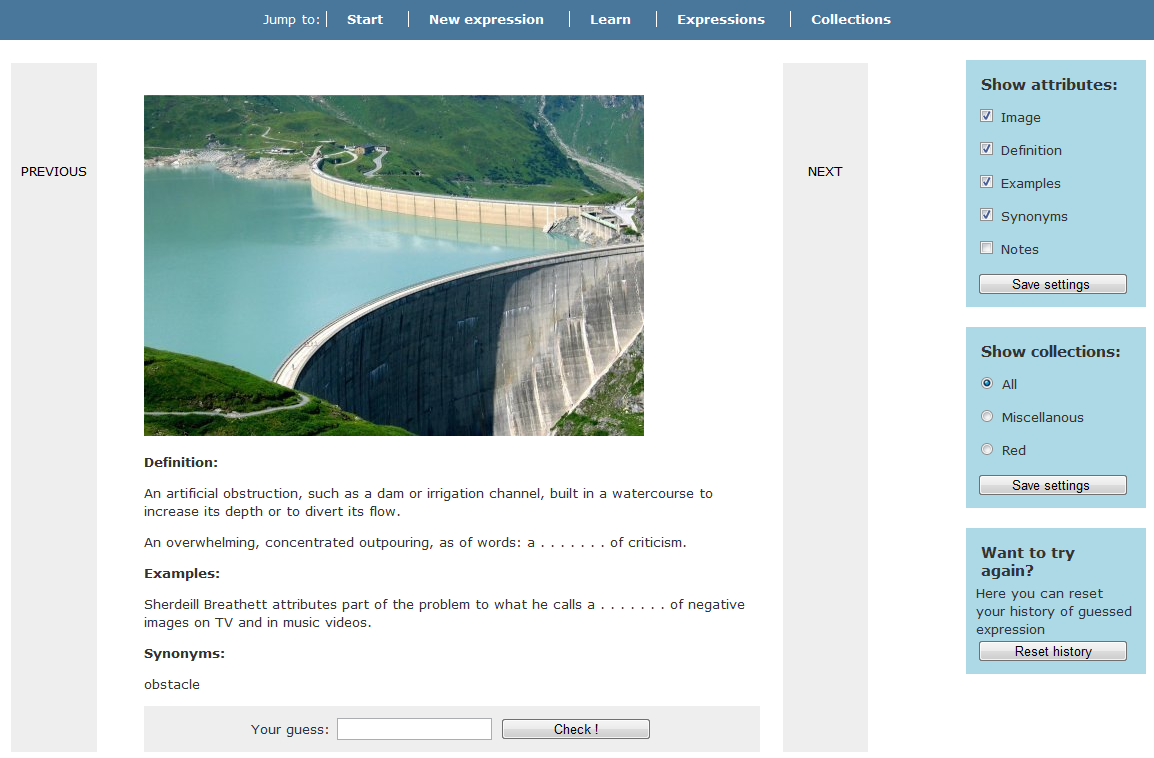
Po kliknięciu w spis wyrażeń powiązanych z kolekcją wyświetlona zostaje lista wyrażeń:



Rysunek 19 – podgląd kolekcji

## Tryb nauki

Po wejściu w tryb nauki prezentowany jest następujący ekran:



Rysunek 20 - Tryb nauki

Widok zawiera opis wyrażenia, bez jego nazwy. Wszelkie jego wystąpienia są filtrowane poprzez użycie kropek.

Wokół wyrażenia widoczne są przyciski 'PREVIOUS' oraz 'NEXT'. Służą one do nawigacji, pozwalają przechodzić do odpowiednio poprzedniego i następnego wyrażenia.

Zadaniem użytkownika jest odgadnięcie nazwy wyrażenia na podstawie prezentowanych danych. Odgadywanie odbywa się przy pomocy następującego elementu:



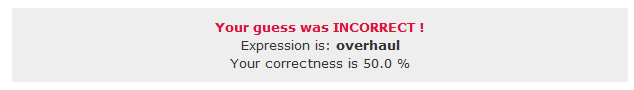
Rysunek 21 - Odgadywanie nazwy wyrażenia

Po odganięciu nazwy, prezentowany jest widok z uzupełnionym wyrażeniem, rezultatem odgadnięcia oraz procentową wartośią poprawnych odpowiedzi udzielonych przez użytkownika.



Rysunek 22 - Poprawne odgadnięcie

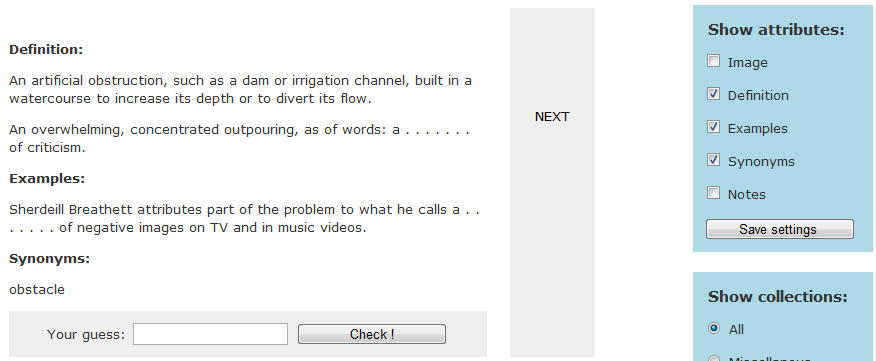
Jeśli odgadnięta nazwa jest niepoprawna, komunikat jest następujący:



Rysunek 23 - niepoprawne odgadnięcie

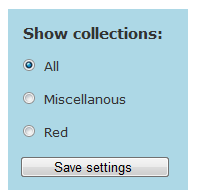
W celu zwiększenia poziomu trudności, można ograniczyć ilość wyświetlanych attrybutów. W tym celu należy odznaczyć odpowiednie wartości w niebieskim prostokącie zatytułowanym 'Show attributes'.

Poniższy obraz ilustruje ograniczenie wyświetlanych atrybutów do definicji, przykładów użycia oraz synonimów.



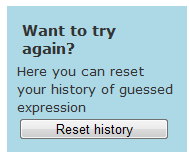
Rysunek 24 - regulacja ilości wyświetlanych atrybutów

W przypadku dużej ilości danych, konieczny staje się podział materiału. Można to osiągnąć poprzez wybranie kolekcji, która będzie używana podczas nauki.



Rysunek 25 - Wybór kolekcji do nauki

Jeśli użytkownik chce skasować historię swoich odpowiedzi, może to uczynić przy pomocy ostatniego niebieskiego prostokątu. Po kliknięciu w przycisk 'Reset history' skasowana zostanie historia odpowiedzi wraz ze statystykami, możliwe będzie ponowne odgadywanie nazw wyrażeń.



Rysunek - Resetowanie historii odpowiedzi



# Podsumowanie

Praca została zrealizowana zgodnie z planem. Aplikacja udostępnia użytkownikowi tryby zarządzania danymi oraz tryb nauki.

Program może służyć jako ciekawsza alternatywa dla nielubianego przez wielu uczenia słówek na pamięć. Dzięki rozbudowanym opcjom tworzenia wyrażeń oraz interaktywności, nauka jest ciekawsza i efektywniejsza.

Projekt można rozwijać w następujących kierunkach:

- dodanie nowych trybów nauki,

- pełniejsza analiza postępów użytkownika, np. automatycznie dostosowywanie poziomu trudności,

- eksportowanie oraz importowanie wyrażeń, w celu wymiany danych pomiędzy użytkownikami lub innymi programami.

# Bibliografia

1. Dave Thomas. Programming Ruby 1.9. The Pragmatic Bookshelf, 2009.
2. <http://en.wikipedia.org/wiki/Ruby_(programming_language)>
3. <http://www.ruby-lang.org/pl/about/>
4. Sam Ruby. Agile Web Development with Rails 4th edition. The Pragmatic Bookshelf, 2011.
5. <http://guides.rubyonrails.org/getting_started.html>
6. <http://docs.rubygems.org/read/chapter/1>
7. <http://en.wikipedia.org/wiki/Haml>
8. <http://sass-lang.com/>
9. <http://en.wikipedia.org/wiki/CoffeeScript>
10. <http://rspec.info/>