ZAAWANSOWANE TECHNOLOGIE INTERNETOWE
WYDZIAŁ INFORMATYKI
POLITECHNIKA POZNAŃSKA

PRACA MAGISTERSKA

POMIAR I WIZUALIZACJA ŚCIEŻEK PRZEGLĄDANIA DOKUMENTÓW HIPERTEKSTOWYCH

Jakub OLEK

Promotor:
dr inż. Adam Wojciechowski

Poznań, 2013

Spis treści

1. WSTĘP

Problematyka pomiaru i wizualizacji ścieżek przeglądania dokumentów hipertekstowych została wybrana jako przedmiot badań prowadzonych w ramach pracy magisterskiej ze względu na ciekawość jej dwóch głównych elementów. Chodzi to o trudność implementacji programu, który zbiera oraz przetwarza dane o ruchu użytkowników na danym serwisie internetowym, oraz następnie wykorzystanie tych danych do wyświetlania czytelnych i zrozumiałych dla użytkownika raportów w postaci:

- grafów przejść
- list stron
- ścieżek przejść

W celu weryfikacji czy zaproponowana metodyka badań daje zadowalające rezultaty zostaną przeprowadzone eksperymenty z wykorzystaniem danych z serwerów Wikii. Jest to firma, która pozwala zakładać użytkownikom własne serwisy internetowe w oparciu o oprogramowanie MediaWiki. Posiada ona aktualnie około 100 milionów użytkowników oraz około 1,64 miliarda odsłon miesięcznie. Daje to bardzo dużą grupę badawczą, która może być użyta w celu zbadania poprawności zastosowanych algorytmów.

1.1 Cel i zakres pracy

Celem pracy jest:

- Implementacja programu zbierającego dane z różnego rodzaju plików logów
- Implementacja wizualizacji tych danych
- Analiza pozyskanych danych

Podczas tworzenia programu szczególna uwaga została zwrócona na prywatność użytkowników danej strony internetowej [1]. Budowany program nie będzie zbierał żadnych informacji, które pozwoliłby na identyfikację konkretnego użytkownika danego portalu. Do tworzenia ścieżek wykorzystywać on będzie tylko i wyłącznie stronę z jakiej użytkownik przeszedł oraz na jaką się udał. W żadnym momencie przetwarzania danych nie będzie wymagana wiedza na temat kto dokładnie poruszał się po tych stronach, a jedynie sumaryczna ilość przejść z jednej strony na drugą. Daje to możliwość badania poruszania się użytkowników po portalu jako całej jego społeczności, bez naruszania prywatności jej członków.

Dane potrzebne do pracy tego programu pobierane mogą być z szerokiej gamy logów, które generują serwery www oraz inne programy odpowiedzialne za rejestrowanie ruchu na stronach www. Napisane zostały dwa główne adaptery do zbierania informacji, do logów z serwera Apache [2] oraz logów zapisywanych przez serwer Scribe [3]. Program stworzony w tej pracy pozwala na jego rozszerzenie o nowe adaptery do pobierania danych oraz o nowe widoki. Dzięki temu można dodać obsługę plików logu dowolnego serwera, lub dodać raport, który uznamy za najwygodniejszy czy najczytelniejszy. Podczas tworzenia programu, stworzone zostanie kilka aplikacji wykorzystujących te dane:

- Wizualizacja ścieżek użytkowników
- Podpowiadanie dokąd użytkownik może w następnej kolejności pójść
- Rysowanie grafów przejść poruszania się użytkowników po danym serwisie

Program został napisany w języku Ruby [4], który jest bardzo wydajnym oraz rozbudowanym językiem do pisania skryptów, oraz aplikacji internetowych. Program można podzielić na dwie logiczne części. Część odpowiedzialną za zbieranie danych, uruchamianą z poziomu linii komend, oraz część odpowiedzialną za wyświetlanie danych i uruchamianą w przeglądarce internetowej. Do przechowywania danych wykorzystana została baza danych MySQL, natomiast do wizualizacji danych zastosowane zostały HTML5, CSS3, JavaScript wraz z biblioteką d3.

Eksperymenty prowadzone przed rozpoczęciem realizacji pracy dyplomowej dawały optymistyczne rezultaty, udało nam się pokazać, że z wykorzystaniem prostego algorytmu jesteśmy w stanie uzyskać ścieżki pomiędzy dokumentami hipertekstowymi, które mają semantyczny sens. Dzięki temu kolejnym użytkownikom możemy, podpowiadać strony, które będą miały dla nich (z wysokim prawdopodobieństwem) duże znaczenie.

1.2 Struktura pracy

Praca składa się z jedenastu rozdziałów oraz wykazu literatury i załączników. Na początku, po krótkim wprowadzeniu, scharakteryzowano zagadnienia prywatności użytkowników oraz wskazane zostały działania podjęte w celu jej zachowania podczas realizacji niniejszego projektu. Następnie opisane zostały źródła danych, z którymi program może współpracować, oraz konfiguracja programu do pracy z zestawem danych. W kolejnych rozdziałach pracy znalazły się: opis serwera, który odpowiedzialny jest za serwowanie danych zapisanych w bazie danych, strony wizualizacji, opis wykonanych prezentacji tych danych oraz możliwe aplikacje, które mogą wykorzystywać taką wiedzę to temat następnego rozdziału. Szczegóły implementacji i możliwości rozszerzenia programu opisane zostały w kolejnych rozdziałach. Na koniec zawarte zostały wyniki badań oraz uwagi końcowe pracy.

2. PRYWATNOŚĆ UŻYTKOWNIKA

Jednym z głównych założeń programu jest nieingerowanie w prywatność użytkowników odwiedzających dany serwis internetowy. Projekt ten stara się odpowiedzieć na pytanie, w którą stronę użytkownicy poruszają się po witrynie i z jaką częstotliwością, a nie kto się po niej porusza. Szczególną uwagę więc skierowano podczas planowania systemu, których danych unikać, a które przechowywać w bazie danych.

2.1 Dane osobowe

Definicja danych osobowych: Daną osobową będzie taka informacja, która pozwala na ustalenie tożsamości danej osoby, bez nadzwyczajnego wysiłku i nakładów, zwłaszcza przy wykorzystaniu łatwo osiągalnych i powszechnie dostępnych źródeł. Poza zakresem przedmiotowej definicji znajdzie się zatem taka informacja, na podstawie której identyfikacja osoby wymagać będzie nieracjonalnych, nieproporcjonalnie dużych nakładów kosztów, czasu lub działań.

W dzisiejszych czasach użytkownicy coraz bardziej zwracają uwagę na swoją prywatność w sieci, oraz robią oni wszystko by tą prywatność utrzymać. Wyłączają oni w swoich przeglądarkach pliki cookies, blokują wykonywanie się skryptów, całkowicie blokują wtyczki typu Adobe Flash, blokują reklamy, proszą serwisy o nie śledzenie ich za pomocą wysyłania zapytania „Do Not Track”, aby uniemożliwić dostawcom treści na ich śledzenie. Program ten nie śledzi użytkowników, bada tylko częstotliwość przechodzenia z jednej strony na drugą, nigdzie nie zapisując kto to zrobił.

2.2 Pliki Cookie

Plik cookie potocznie zwanym ciasteczkiem – to niewielka informacja tekstowa, wysyłana przez serwer WWW i zapisywana po stronie użytkownika (zazwyczaj na twardym dysku). Wymyślona przez byłego pracownika Netscape Communications – Lou Montulliego. Domyślne parametry ciasteczek pozwalają na odczytanie informacji w nich zawartych jedynie domenie, która je utworzyła. Ciasteczka różnych rodzajów są stosowane najczęściej w przypadku liczników, sond, różnorakich sklepów

internetowych, zapamiętywania stanu zalogowania użytkownika, reklam oraz monitorowania aktywności odwiedzających.

2.3 Do Not Track

„Do Not Track" (DNT) to proponowany nagłówek HTTP – DNT, który prosi o to by aplikacja wyłączyła śledzenie na witrynie oraz między witrynami użytkownika (to nie pełne określenie tego, zostaje nie rozwiązane). Nagłówek ten został zaproponowany w 2009 roku, aktualnie jest standaryzowany przez organizację W3C. Nagłówek ten wprowadzony był pierwszy przez Mozillę w ich przeglądarce Firefox, następnie dołączyły takie przeglądarki jak Safari, Opera oraz Chrome, a pod koniec 2010 roku, Microsoft wprowadził ten mechanizm do swojej przeglądarki internetowej Internet Explorer 9. Nazwa pola w nagłówku to DNT, a akceptuję ona 3 wartości:

- 1 – jeśli użytkownik nie chce być śledzony
- 2 – jeśli użytkownik zgadza się na śledzenie
- null – jeśli użytkownik nie określił swoich preferencji

Domyślnie przeglądarka nie wyśle tego nagłówka dopóki użytkownik, nie podejmie decyzji. System ten jest kompletnie dobrowolny, i nie ciągnie za sobą żadnych prawnych czy technologicznych konsekwencji. Witryny internetowe oraz reklamodawcy mogą uznać wolę użytkownika lub całkowicie ją zignorować.

2.4 Śledzenie użytkowników

W żadnym momencie działania programu (zbieranie danych, wizualizacja danych) nie wykorzystywane są jakiekolwiek dane użytkownika. Wiele serwisów do generowania tego typu danych wykorzystuje IP, login, *beacon* (unikatowy identyfikator nadawany użytkownikowi) itp. Wiążą one więc ściśle dane o ruchu z konkretnymi użytkownikami. Proponowana implementacja tego zagadnienia, pozostawia prywatność nienaruszoną poprzez zupełne ignorowanie powiązania zarejestrowanego przejścia ze strony na stronę, a danym użytkownikiem.

Web beacon – wykorzystywane są w połączeniu z plikami *cookies* by pomóc w zrozumieniu zachowań użytkowników odwiedzających witrynę internetową. Jest to przeważnie przezroczysty plik graficzny (najczęściej 1x1 piksel), który dodawany jest do strony bądź poczty e-mail. Użycie web *beacon'a* pozwala stronie na zbieranie danych o akcjach jakie wykonują użytkownicy. Jest on częścią strony tak samo jak tekst czy zdjęcia, jednak jest on tak mały, że jest nie widoczny. Obrazek ten jest ściągany z innego serwera który działa jako miejsce zbierania tych danych. Moment zapytania o ten obraz to moment w którym serwer ten może zapisać, że jakieś zdarzenie miało miejsce. Na przykład firma posiadająca sieć stron może używać tej technologii w celu zliczania oraz rozpoznawania użytkowników poruszających się po tej sieci. Zamiast gromadzić statystyki oraz zarządzanie plikami cookie na wszystkich ich serwerach osobny, mogą oni użyć web *beacon'ów* aby trzymać te dane razem. Posiadanie takiej możliwości, pozwala na lepsze spersonalizowanie wizyty oraz stworzenia jej bardziej przyjaznej.

Wyniki badań poprzez nie wykorzystanie tych technologii nadal są poprawne, gdyż nie jest nam potrzebna wiedza o konkretnym użytkowniku, który odwiedza witrynę, a jedynie ilość przejść ze jednej strony na drugą. Daje nam to balans pomiędzy utrzymaniem prywatności użytkowników, a ilością pytań na które możemy odpowiedzieć mając dane statystyki.

3. ŹRÓDŁA DANYCH

Program ten napisany został napisany z myślą o jego rozszerzalności, prostocie konfiguracji oraz możliwości podpięcia wszelkiego rodzaju źródeł danych. Wprowadza on więc koncepcję adapterów, które odpowiedzialne są za czytanie i dostosowywanie plików z danymi w ten sposób, aby główna cześć programu mogła je łatwo przekształcić, zliczyć oraz zapisać w bazie danych.

3.1 Rozszerzalność

Kolejnym z głównych założeń programu jest możliwość jego rozszerzania o nowe źródła danych. Poprzez wprowadzenie koncepcji adaptera program można rozszerzyć o dowolne typy logów czy ich wariacji.

Adapter jest to osobna klasa która definiuje jedną metodę *load* której zadaniem jest zwrócić dane w odpowiedniej formie tak aby główny moduł programu mógł je dalej zrozumieć i zapisać w bazie danych. Metoda ta głównie składać się będzie z paru części

- czytania pliku
- filtrowania wpisów
- transformacji ich do wymaganej formy

Dokładny format opisać należy w pliku konfiguracyjnym, to jest dokładnie jakie dane mają zostać zapisane oraz znak podziału. Domyślnie są to pola – nazwa serwisu, strona startowa oraz strona docelowa.

Adapter taki zapisany powinien być w folderze *bin/adapters* z nazwą odpowiadająca nazwie klasy np.: adapter dla Apache powinien być zapisany w *bin/adapters/ApacheAdapter.rb* a klasa powinna nazywać się *ApacheAdapter*.

3.2 Konfiguracja

W celu poinstruowania programu do używania danego adaptera należy podać jego nazwę w pliku konfiguracji. Daje to możliwość szybkiej zmiany źródła danych.

3.3 Gotowe adaptery

Razem z programem napisane zostały dwa adaptery do logów

- Apache
- serwera Scribe oraz formatu wykorzystywanego w firmie Wikia

Apache został wybrany, gdyż daje bardzo duże możliwości jeśli chodzi o wykorzystanie tego projektu w wielu serwisach. Serwer Apache jest to otwarto-źródłowy serwer HTTP dostępny dla wielu różnych platform. Jest to najszerzej stosowany serwer HTTP w Internecie. Pod koniec 2011 roku jego udział wśród serwerów wynosił 65%. W połączeniu z interpreterem języka skryptowego PHP i bazą danych MySQL, Apache stanowi jedno z najczęściej spotykanych środowisk w firmach oferujących miejsce na serwerach sieciowych.

Logi serwera Scribe zostały wybrane, gdyż takimi posługuje się Wikia – serwis umożliwiający bezpłatne uruchamianie projektów wiki opartych na oprogramowaniu MediaWiki. Aktualnie serwis ten posiada ponad 100 milionów użytkowników oraz 1,64 miliarda odwiedzin miesięcznie. Daje nam to bardzo duża grupę badawczą, by móc stwierdzić czy zaproponowane rozwiązanie daje wymierne wyniki.

3.4 Logi Apache

Aby móc wykorzystać logi które zapisywane są przez serwer Apache należy go odpowiednio skonfigurować. Modułem Apache odpowiedzialnym za konfigurację logowania zdarzeń to *mod_log_config*. Moduł ten daje możliwość łatwego przystosowania logowania zapytań klientów. Logi te mogą być zapisane do pliku w konfigurowalnym formacie lub mogą być wysyłane do zewnętrznego programu. Zdarzenia mogą być logowane również z uwzględnieniem reguł tak, że konkretne

zapytania mogą być zawierane lub pomijane z logów na podstawie charakterystyk zapytania. Moduł ten udostępnia trzy dyrektywy:

- *TransferLog* – do tworzenia plików logu
- *LogFormat* – do konfiguracji formatu logowania
- *CustomLog* – do zdefiniowania pliku logów oraz jego formatu w jednym kroku

Dyrektywy *TransferLog* oraz *CustomLog* mogą zostać użyte wielokrotnie na każdym z serwerów w celu logowania każdego zapytania do wielu plików. Format dyrektywy LogFormat i CustomLog to ciąg znaków. Ten ciąg znaków wykorzystywany jest do logowania każdego zapytania. Zawierać może on znaki które zostaną skopiowane do pliku logu oraz znaki kontrolne w stylu C „\n” oraz „\t” do reprezentacji znaków nowych linii oraz tabulacji. Znaki cudzysłowu oraz backslasha powinny zostać poprzedzone znakiem „\”. Charakterystyki zapytania logowane są za pomocą dyrektyw „%” w ciągu znaków formatu. Zamienione one zostają na odpowiednie wartości w pliku logów. Moduł ten może zapisać wiele informacji na temat każdego zapytania, natomiast w celu poprawnego działania programu potrzebne natomiast są dwie informacje:

- ***%U*** – zapisuje on ścieżkę URL zapytania, bez znaków po znaku ?
- ***%{Referer}i*** – zapisuje wartość linii Referer: nagłówka w zapytaniu wysłanym do serwera.

Przykładowa linia w pliku httpd (pliku konfiguracyjnym serwera Apache), która pozwoli na działanie programu:

```
CustomLog "%{Referer}i -> %U"
```

3.5 Logi Scribe

Scribe jest to serwer do agregowania logów ze strumienia danych. Zaprojektowany został by mógł być skalowany do bardzo dużej ilości maszyn oraz być odpornym na awarie sieci oraz maszyn. Scribe to serwer, który uruchomiony jest na każdej maszynie w systemie, skonfigurowany do agregowania wiadomości oraz wysyłania ich do centralnego serwera lub centralnych serwerów Scribe w większych grupach. Jeśli jednostka centralna nie jest dostępna, lokalna instancja serwera zapisuje wiadomości do pliku na lokalnym dysku i wysyła je w momencie odzyskania przez serwer centralny sprawności. Główne serwery Scribe mogą zapisać te wiadomości do plików, które są ich docelowym miejscem, lub wysłać je do kolejnej warstwy serwerów.

Scribe jest wyjątkowy w tym, że log klienta składa się z dwóch ciągów znaków, kategorii oraz wiadomości. Kategoria to wysoko poziomowy opis miejsca przeznaczenia wiadomości oraz może posiadać specyficzną konfiguracje dla serwera Scribe, co pozwala na przeniesienie miejsca przechowywania danych poprzez zmianę konfiguracji zamiast kodu klienta. Serwer pozwala również na konfigurację na podstawie przedrostka kategorii oraz ustawienia domyślne, które mogą dodawać nazwę kategorii do nazwy pliku. Łatwość dostosowywania oraz rozszerzalność jest zapewniona poprzez abstrakcję „składu”. Składy te ładowane są dynamicznie na podstawie konfiguracji i mogą być zmieniane podczas działania programu bez zatrzymywania serwera.

Scribe zaimplementowany został jako usługa thrift z wykorzystaniem nieblokującego serwera C++. Używany jest na tysiącach serwerów Facebook's i niezawodnie dostarcza dziesiątki miliardów wiadomości dziennie. Thrift to protokół komunikacji binarnej, napisany do obsługi wielu języków. Opracowany został przy tworzeniu portalu internetowego Facebook do rozwoju skalowalnych usług dla wielu języków.

4. KONFIGURACJA

Konfiguracja programu odbywa się za pomocą pliku konfiguracyjnego zapisanego w języku YAML [4]. Program wymaga dostępu do pliku konfiguracyjnego z którego odczytać może takie dane jak:

- dane bazy danych
- informacje potrzebne do uruchomienia serwera
- nazwa adaptera, który ma być użyty

4.1 Położenie

Położenie pliku konfiguracyjnego jest dowolne jak długo program oraz serwer mogą go odczytać. Aby wczytać plik konfiguracyjny do programu należy uruchomić program z parametrem '-c' np:

```
./pathfinder.rb –m etl –c config.yaml
./pathfinder.rb –m server –c path/config.yaml
```

4.2 Format

Do zapisu konfiguracji wybrany został język YAML [5]. Jest to uniwersalny język formalny przeznaczony do reprezentacji różnych danych w ustrukturalizowany sposób. Jego nazwa to akronim rekursywny od słów '*YAML Ain't Markup Language' (ang. YAML, to nie kolejny język znaczników)*. Początkowo skrót ten oznaczać miał *'Yet Another Markup Language'* (ang. kolejny język znaczników), jednak zmieniono tę koncepcję, w celu położenia nacisku na cel, któremu ten język ma służyć, to jest opisowi zbioru danych.

Poszczególne elementy struktury danych są oddzielone znakami nowej linii, a ich hierarchia ustalana jest na podstawie wcięcia linii. Język wprowadza trzy podstawowe struktury danych, które można wykorzystać w dokumencie: słowniki, listy

oraz skalary. Została w nim wprowadzona obsługa referencji przez co wyeliminowana została konieczność redundancji danych.

Wszystkie te cechy sprawiają, że dokumenty napisane w języku YAML są czytelne dla człowieka, zwięzłe oraz dają się w łatwy sposób przetwarzać prostym narzędziom tekstowym, takim jak grep, awk czy sed. Ruby [6] posiada wbudowaną bibliotekę do czytania danych zapisanych w tym języku.

4.3 Opcje

Poprzez plik konfiguracyjny mamy pełną kontrolę nad tym jak program działa. Poniżej wypisane są wszystkie dostępne opcje wraz z wymaganą hierarchią. Najważniejszymi częściami pliku konfiguracji to część *database, adapter* oraz *server* gdyż opisują one kluczowe części programu.

- *database* – dane dotyczące połączenia z bazą danych. Pola te są wymagane zarówno przez część zapisująca dane jak i serwer.
    - *host* – host na którym znajduje się baza danych. Domyślnie jest to *localhost*.
    - *user* – użytkownik który ma być użyty do połączenia. Domyślnie jest to *root*
    - *password* – hasło użytkownika. Domyślnie jest puste
    - *DB* – nazwy bazy, w której zapisane są dane. Domyślnie jest to *PathFinder*
- *adapter* – dane dotyczące adaptera który ma być użyty
    - *name* – nazwa adaptera np.: Apache
    - *config* – dane które mają być przekazane do adaptera. Każdy adapter, może wymagać innych danych do poprawnego działania
- server – dane dotyczące serwera API

- port – port na jakim serwer ma nasłuchiwać zapytań
- CORS – czy serwer ma zezwalać na połączenie z dowolnego hosta. Domyślnie dostęp taki jest zabroniony

Dane które można przekazać do adaptera Scribe oraz Apache:

- *tail* – ile ostatnich linii pliku ma być przeczytane
- *head* – ile pierwszych linii pliku ma być przeczytane
- *grep* – wyrażenie regularne, które ma być użyte na każdej linii logu w celu dodatkowego filtrowania danych
- *file* – wzór jaki ma być użyty do szukania plików. Przyjmuję on również gwiazdkę jako wilczą kartę np.: logs/* lub logs/logs-2013-07-04-*
- *threads* – ile wątków ma zostać użyte do czytania plików. Czytanie oraz parsowanie dużej ilości danych można przyspieszyć za pomocą wykonania tej czynności na wielu wątkach. Wartość tę należy ustawić odpowiednio do sprzętu na jakim program ten jest wykonywany.

Tail, *head* oraz *grep* są to przydatne opcje podczas testów naszego systemu, znacząco mogą one przyspieszyć czas trwania parsowania pliku przez co możemy zobaczyć, czy cały system działa stabilnie.

Przykładowy plik konfiguracji:

```
#Database
database:
  host: localhost
  user: root
  password:
  DB: PathFinder

#Adapter
adapter:
  name: Scribe
  config:
    file: 07/*
    threads: 4

#Server
port: 2000
CORS: true
```

5. SERWER

Program można uruchomić również w trybie serwera, który nasłuchuje połączeń i zwraca dane w formie dokumentu JSON. Daje on możliwość wykorzystania danych zbieranych przez program w innych programach. Implementuje on również prosty serwer HTTP, który może być użyty do serwowania części programu odpowiedzialnej za wizualizacje.

5.1 Konfiguracja

Serwer w całości konfigurowany jest przez plik konfiguracji. Dane które wymaga on do pracy to dane na temat połączenia z bazą danych, port na którym nasłuchiwać ma połączeń oraz czy zezwalać na połączenia CORS. Przykład części konfiguracji odpowiedzialnej za serwer:

```
port: 5005
CORS: false
```

5.2 Uruchamianie

W celu uruchomienia serwera należy uruchomić program z odpowiednimi parametrami. Należy również pamiętać o tym by dane, których wymaga do pracy znajdowały się w pliku konfiguracji. Polecenie:

```
./pathfinder.rb –m server –c config.yaml
```

utworzy instancje programu, która połączy się z bazą danych oraz rozpocznie nasłuchiwanie połączeń na zadanym porcie.

5.3 Zaimplementowane metody

Serwer składa się z dwóch logicznych części:

- serwująca część programu odpowiedzialnej za wizualizację

- serwująca dane na temat ścieżek użytkowników

Pierwsza jego część umożliwia uruchomienie programu w przeglądarce internetowej poprzez połączenie sieciowe. Co daje możliwość prostego uruchomienia programu na dowolnym komputerze oraz dostępu do niego poprzez nawigację przeglądarki pod adres danego komputera.

Druga część odpowiedzialna jest za serwowanie danych i jest to proste *REST'owe* API aplikacji. Umożliwia ona dostęp do danych zapisanych w bazie danych, w celu dalszego ich wykorzystania w innych aplikacjach. Jest to cześć łącząca część programu odpowiedzialnej za zbieranie danych oraz część, która te dane wizualizuje.

API (ang. *Application Programming Interface*) opisuje jak komponenty oprogramowania powinny się ze sobą komunikować. W praktyce większość API to biblioteka która zawiera specyfikacje algorytmów, struktur danych, klas, obiektów i zmiennych. Specyfikacja API może mieć wiele form takich jak międzynarodowy standard POSIX, zamknięte specyfikacje takie jak Microsoft Windows API, biblioteki języków programowania. API w kontekście aplikacji internetowych, zwykle zdefiniowane jest jest jako zestaw wiadomości żądań HTTP oraz struktury wiadomości odpowiedzi, która najczęściej przyjmuje formę dokumentu XML lub JSON.

REST (ang. *Representational state transfer*) to styl architektury rozdystrybuowanych systemów takich jak WWW. REST wyrósł na przewodni model projektowania sieciowego API. Styl architektury REST został stworzony przez W3C Technical Architecture Group (TAG) na równi z HTTP/1.1 i bazuje na już istniejącym projekcie HTTP/1.0. WWW jest to największa implementacja systemu który zgodny jest ze stylem architektonicznym REST. Architektura REST poprzez konwencje składa się z klientów oraz serwerów. Urządzenia klienckie wykonują zapytania do serwerów, a serwery przetwarzają żądania oraz zwracają odpowiednie dane. Żądania i odpowiedzi budowane są w około przesyłu reprezentacji danych zasobów. Klient rozpoczyna wysyłanie żądania w momencie kiedy jest gotowy na zmianę do nowego stanu.

Zaimplementowany serwer poprzez swoją naturę obsługuje tylko zapytania GET. Podstawowy format URL do zapytania po dane to:

host/api/metoda[/serwis[/strona]]

gdzie

- metoda – jakie zapytanie wykonać do programu
- serwis – z którego serwisu pobierać dane
- strona – dla jakiej strony pobrać dane

Dostępne metody to:

- get_sites – zwraca dostępne serwisy w bazie danych, użyte do zaimplementowania podpowiadania serwisów w części prezentacyjnej
- get_pages – zwraca dostępne strony dla danego serwisu odpowiadające stronie, użyte do zaimplementowania podpowiadania stron dla danego serwisu w części prezentacyjnej
- get_path – zwraca pełną ścieżkę dla zadanej strony oraz serwisu
- get_most_visited – zwraca wszystkie strony na jakie udali się użytkownicy z zadanej strony posortowane malejąco względem ilości przejść
- get_all_paths – jest to połączenie dwóch powyższych metod zwraca ścieżki dla wszystkich stron na jakie udali się użytkownicy z danej strony

5.4 Format odpowiedzi

Serwer zwraca dane w formacie JSON (*JavaScript Object Notation*). Jest to oparty o tekst otwarty standard zaprojektowany do wymiany danych czytelny dla człowieka. Pochodzi on z języka skryptowego JavaScript do reprezentacji prostych struktur danych oraz tablic asocjacyjnych nazywanych obiektami. Nie jest on ograniczony do pracy z JavaScript, dostępne są jego parsery dla wielu języków programowania. Format ten został opracowany przez Douglasa Crockforda i opisany

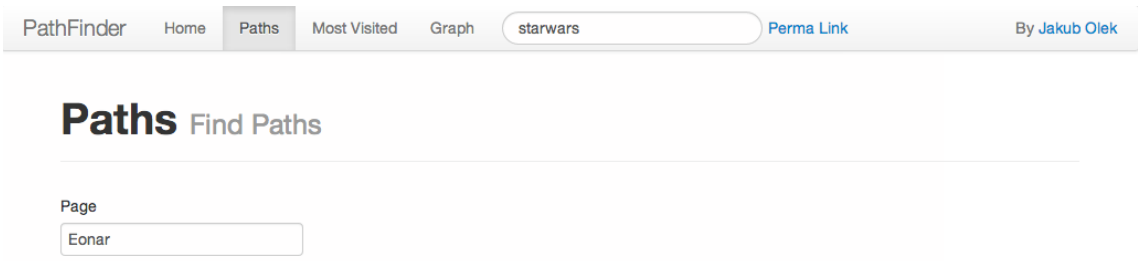
jest w dokumencie RFC 4627. Oficjalnym typem dla dokumentu JSON to *application/json* a rozszerzenie pliku to *.json*.

Format ten najczęściej używany jest do serializowania oraz transferu strukturyzowanych danych poprzez połączenie sieciowe. Głównie używany do przesyłania danych pomiędzy aplikacją internetową oraz serwerem, jako alternatywa dla XML. Typy które przechowywane mogą być w dokumencie JSON to: liczba, ciąg znaków, tablica, obiekt, null oraz logiczny. Lekka waga dokumentu zapisanego w JSON oraz dostępność parsera w Ruby oraz JavaScript, spowodowało, że jest to format idealny do zastosowań pisanego programu.

6. WIZUALIZACJA

Kolejnym elementem systemu jest wizualizacja danych które zostały zebrane. Daje ona możliwość przyjrzeniu się danym w czytelny dla człowieka sposób. Dzięki temu można zobaczyć, jak użytkownicy poruszają się po danej witrynie internetowej. Zostały napisane 3 wizualizacje:

- najbardziej popularnej ścieżki z zadanej strony
- wszystkich stron docelowych z zadanej strony
- wszystkich ścieżek z zadanej strony



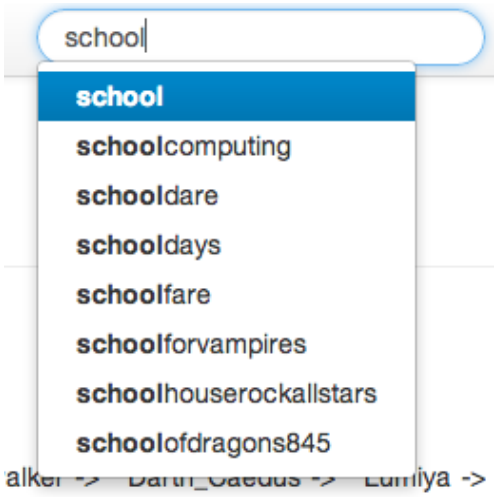
Ilustracja 1: Główne okno programu

Aby je wyświetlić należy uruchomić serwer, a następnie wejść na odpowiednią stronę np.:

```
localhost:2000/#graph
locahhost:2000/#paths
localhost:2000/#paths/starwars/Luke_Skywalker
```

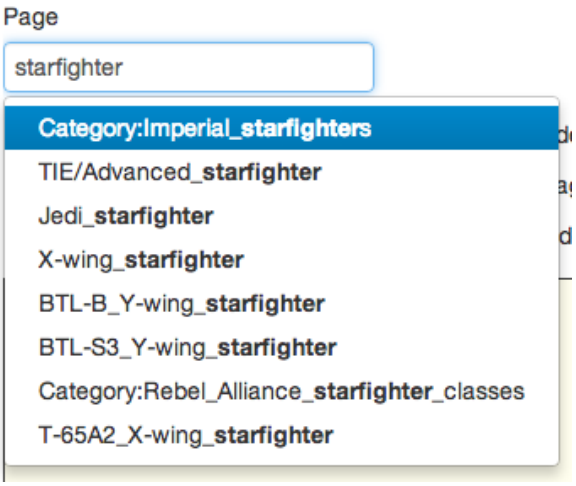
Wszystkie wizualizacje są animowane oraz interaktywne. Każda strona reprezentowana jest za pomocą małego koła oraz jej nazwy, a przejścia reprezentowane są poprzez linie zakończone grotami skierowanymi w kierunku przejścia, które rozciągają się pomiędzy stroną, z której użytkownik przeszedł a stroną, na którą się udał. We wszystkich wizualizacjach występuje również model fizyczny. Strony

oddziaływają na siebie z siłą ujemną przez co odpychają się od siebie się z odpowiednią mocą. Linki pomiędzy nimi utrzymują je przy sobie. Im częściej użytkownicy korzystają z danego połączenia tym jest ono mocniejsze i sztywniejsze. Po najechaniu na stronę nazwa jej powiększy się w celu łatwiejszego jej rozpoznania. Po kliknięciu na stronę możemy ją zawieść w przestrzeni, co daje możliwość ułożenia stron i ścieżek w sposób który jest w danym momencie najwygodniejszy, co jest bardzo przydatne podczas analizy wyników. Używając kółka myszki można przybliżyć oraz oddalić prezentację, co daje możliwość skupienia się na całości prezentacji, bądź wybranym fragmencie. Prezentację można również przeciągać we wszystkie strony aby wynaleźć interesujący jej fragment. Aby wybrać witrynę internetową do pracy należy wprowadzić jej nazwę w górnym pasku. Zaimplementowana została funkcja podpowiedzi, dzięki temu po wprowadzeniu pierwszego znaku system wyszuka witryny, które odpowiadają wyszukiwaniu i podpowie nam je w małym oknie pod polem tekstowym.



Ilustracja 2: Okno podpowiedzi witryny, opracowanie własne

Podobne rozwiązanie zostało zaimplementowana przy polu tekstowym służącym do wyboru strony dla której system załadować ma dane.



Ilustracja 3: Okno podpowiedzi strony, opracowanie własne

Kolejnym zaimplementowanym udogodnieniem jest generowanie linków do konkretnej wizualizacji. Klikając w link „*Perma link*" zostanie wygenerowany URL, który po otworzeniu przejdzie prosto do danej wizualizacji. Przykład takiego linku:

```
localhost:2000/#paths/starwars/Luke_Skywalker
localhost:2000/#get_all_paths/harrypotter/Hermiona
localhost:2000/#get_most_visited/vim/Shortcuts
```

Link taki jednak nie zapisuje stanu aktualnego grafu tj. układu stron na wizualizacji czy dodanych stron ścieżek do wizualizacji grafu.

6.1 Biblioteki

W celu ułatwienia implementacji wizualizacji oraz interfejsu graficznego wykorzystane zostały biblioteki jQuery, Bootstrap, Mustache oraz D3. Wszystkie te biblioteki są otwarto źródłowe oraz pozwalają na wykorzystanie ich w projektach bez wykupowania ich licencji.

6.1.1 jQuery

Jest to lekka biblioteka programistyczna dla języka JavaScript. Ułatwia ona korzystanie z tego języka oraz manipulacji drzewem DOM (ang. *Document Object Model*). Pomaga dodać do strony efekty animacji, dynamiczne zmienianie strony, wykonywać zapytania AJAX. Większość wtyczek i skryptów opartych na tej bibliotece działa na stronach bez wymagania zmian w kodzie HTML. Biblioteka ta nie umożliwia żadnych nowych rzeczy, jednakże znacząco ułatwia ona pracę z językiem JavaScript. Sprawiając, że do wykonania zadania trzeba napisać mniej, prostszego, bardziej przejrzystego kodu. W projekcie, jQuery został wykorzystany w wersji 2.0.3, głównie do zapytań AJAX do napisanego serwera API, manipulacji elementami na stronie oraz jako biblioteka od której zależna jest biblioteka Bootstrap.

6.1.2 Bootstrap

Jest to darmowa biblioteka, rozwijana przez programistów firmy Twitter, wydana na licencji *Apache Licence 2.0*. Zawiera on zestaw przydatnych narzędzi ułatwiających tworzenie interfejsów graficznych aplikacji oraz stron internetowych. Bazuje ona głównie na gotowych rozwiązaniach HTML oraz CSS. Wykorzystać ją można do stylizacji wszystkich elementów z których składa się typowa aplikacja czy strona internetowa tj. tekst, formularze, przyciski, wykresy, nawigacje oraz wielu innych. Posiada ona również rozszerzenia dla języka JavaScript, która daje możliwość stworzenia menu, podpowiadania wyników wyszukiwania czy interfejsu w formie zakładek. W projekcie biblioteka ta wykorzystana została w wersji 2.3.2, do stworzenia głównego interfejsu użytkownika, pól tekstowych z podpowiedziami, menu oraz wykonania strony w formie zakładek.

6.1.3 Mustache

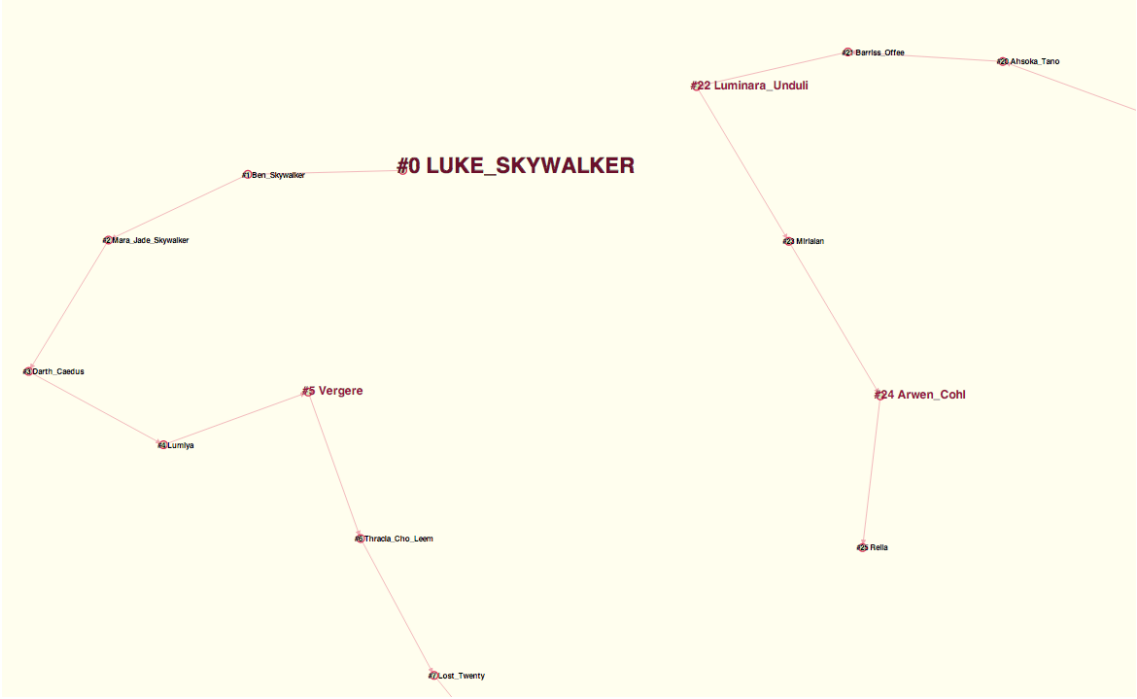
Mustache to prosty system szablonów z implementacją dostępną dla ActionScript, C++, Clojure, CoffeScript, ColdFusion, D, Erlang, Fantom, Go, Java, JavaScript, Lua, .NET, Objective-C, Perl, PHP, Python, Ruby, Scala, XQuery. Opisywana jest jako system bez logiki gdyż nie posiada wyraźnej obsługi zapisu kontroli przepływu takich jak if/else czy pętli. Możliwe jednak jest to do uzyskania poprzez tagi sekcji, listy oraz funkcje. Nazwany „Mustache" ponieważ głównym znakiem wykorzystywanym przez bibliotekę jest „}" oraz „{", które przypominają wąsy. W projekcie biblioteka ta wykorzystana została do generowania tabeli na stronie która ukazuje najczęściej odwiedzane strony z zadanej strony.

6.1.4 D3

D3 to biblioteka stworzona do manipulacji dokumentami bazująca na danych. Pomaga ona 'ożywić' dane wykorzystując HTML, SVG oraz CSS. Kładzie ona nacisk na standardy i udostępnia ona pełne możliwości nowoczesnych przeglądarek bez związywania z dodatkowymi bibliotekami. Łączy ona rozbudowane komponenty wizualizacji oraz podejście sterowane danymi do manipulacji DOM. D3 umożliwia łączyć dowolne dane do drzewa DOM a następnie zastosować transformacje sterowane tymi danymi do dokumentu. Na przykład, wykorzystując D3 możemy wygenerować tabelkę HTML z danymi z tablicy danych, lub użyć te same dane do stworzenia interaktywnego wykresu z animacjami wykorzystując SVG. W projekcie biblioteka ta wykorzystana została w wersji 3, do generowania animowanych oraz interaktywnych grafów przejść użytkowników.

6.2 Ścieżki

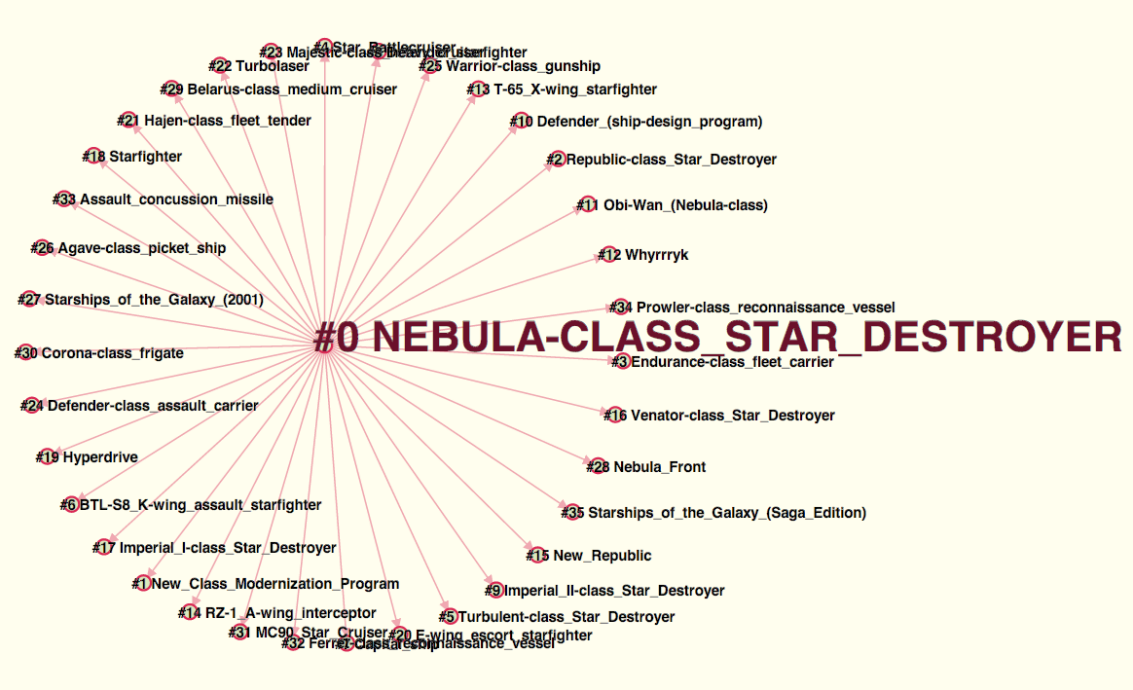
Wizualizacja ukazana na ilustracji 4 ukazuje ścieżkę najczęstszych przejść z zadanej strony. Daje ona możliwość znalezienia statystycznie najciekawszych stron dla użytkowników, którzy weszli na daną stronę. Pozwala ona przejrzeć na jakie strony użytkownicy najczęściej udają się z zadanej strony i w jakiej kolejności.



Ilustracja 4: Przykład wygenerowanej ścieżki dla strony Luke Skywalker, opracowanie własne

6.3 Najczęściej odwiedzane strony

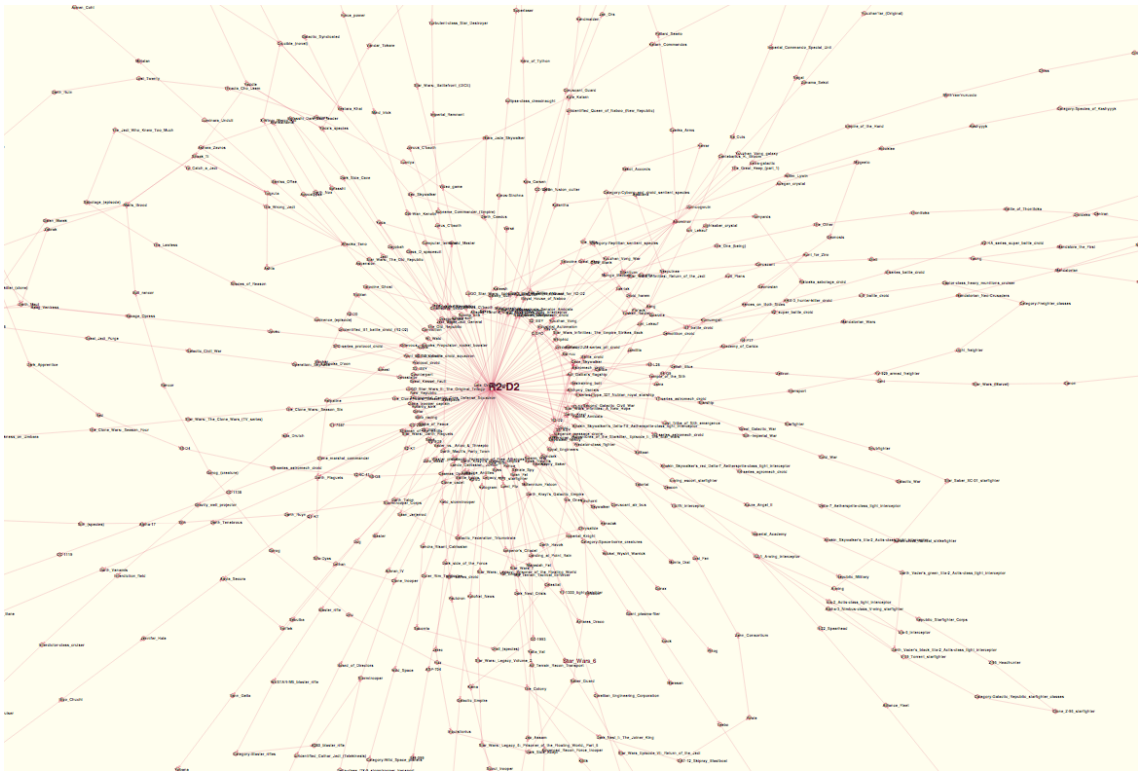
Ilustracja 5 to wizualizacja, która ukazuje wszystkie strony na jakie przeszli użytkownicy z zadanej strony. Pokazuje to gdzie użytkownicy przechodzą oraz z z jaką częstotliwością. Prezentacja ta pozwala na podejrzenie, które linki na danej stronie są wykorzystywane.



Ilustracja 5: Przykład przejść z zadanej strony, opracowanie własne

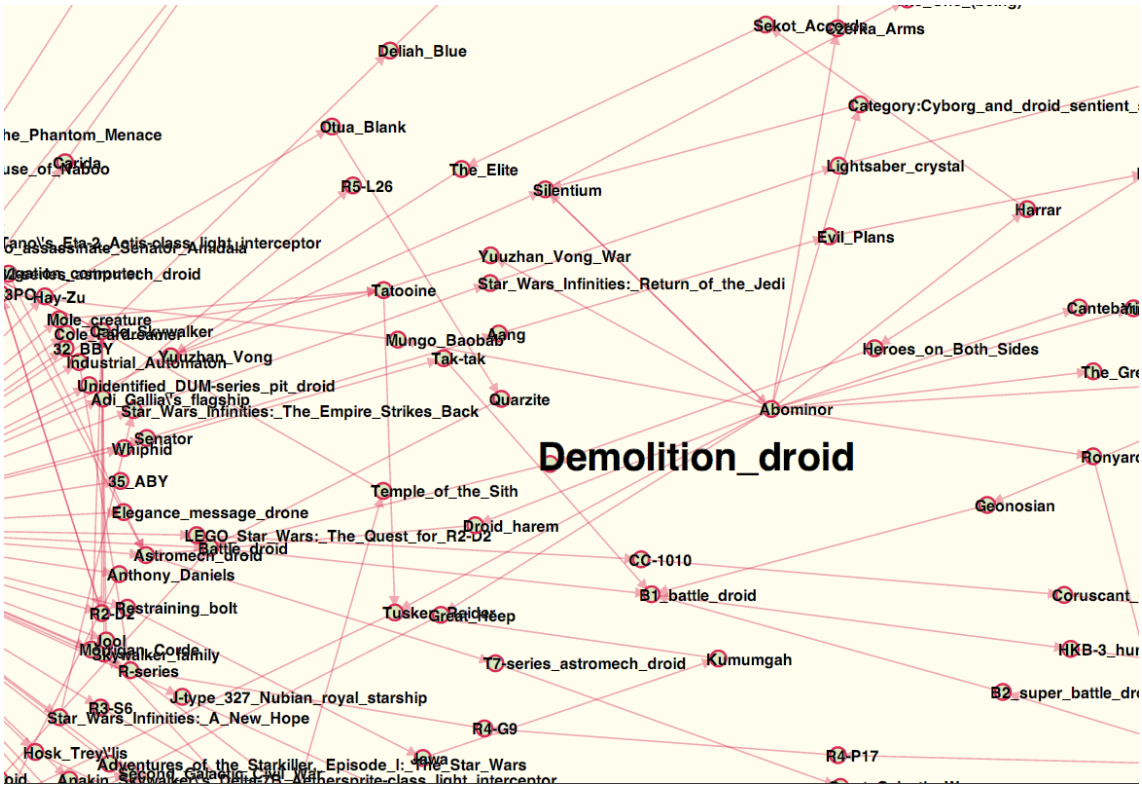
6.4 Graf przejść

Jest to najbardziej rozbudowana wizualizacja z przygotowanych. Ukazuje on wszystkie znalezione ścieżki z zadanej strony. Widok ukazany na ilustracji 6 umożliwia większą eksplorację danych. Po odpowiedniej konfiguracji tego widoku daje on możliwość spojrzenia na strukturę danej witryny. Pozwala zobaczyć jak użytkownicy poruszają się po całym serwisie, które tematy są bardziej interesujące, a które są odwiedzane rzadziej.



Ilustracja 6: Przykład grafu przejść dla strony R2-D2, opracowanie własne

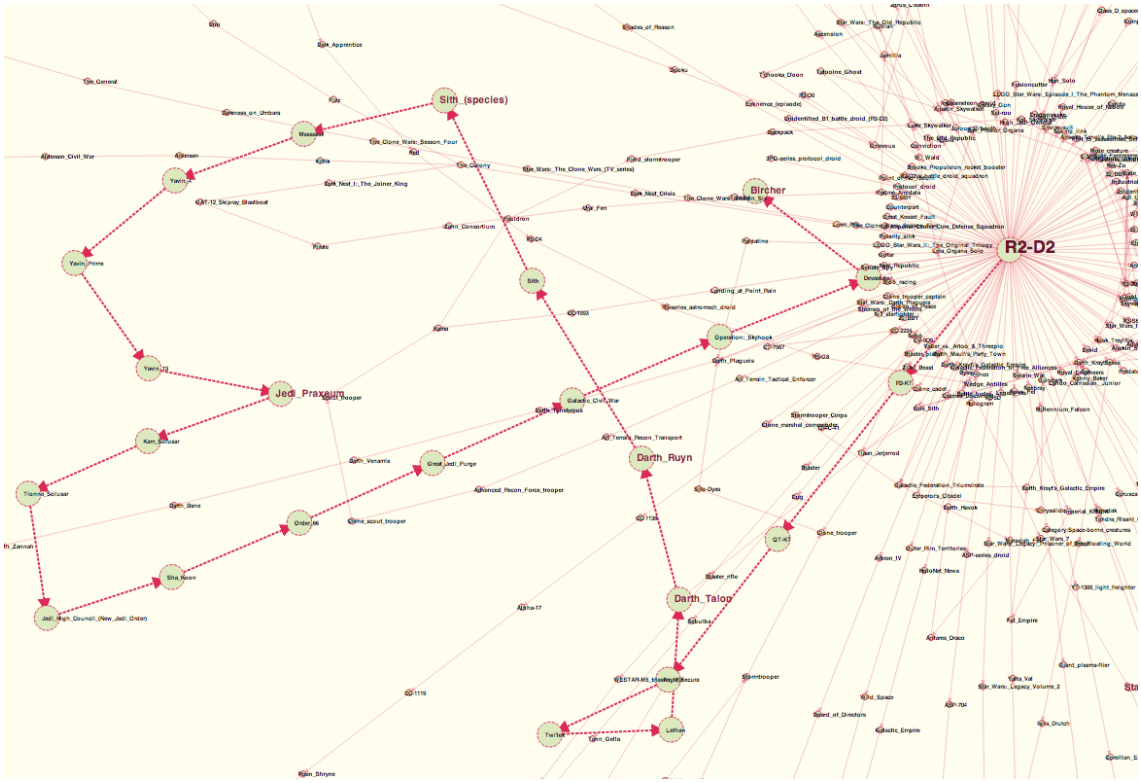
Na tej prezentacji bardzo przydatną funkcją jest wcześniej wspomniana możliwość powiększania oraz pomniejszania wizualizacji, jest to pokazane na ilustracji 7. Daje to możliwość lepszego przyjrzenia się wynikom.



Ilustracja 7: Przybliżenie na fragment wizualizacji, opracowanie własne

Na tej wizualizacji dodatkowo została zaimplementowana obsługa dodawania kolejnych stron i ich połączeń. Klikając na stronę z wciśniętym przyciskiem *alt* na klawiaturze, system pobierze z bazy danych wszystkie ścieżki dla strony, która została kliknięta i doda je do grafu. Umożliwia to dokładniejsze badanie obszarów strony internetowej.

W celu łatwiejszego znajdowania ścieżek, w tej prezentacji dodana została obsługa animowanego podświetlania ścieżek, która pokazana jest na ilustracji 8, by ją aktywować należy najechać kursorem na daną ścieżka z wciśniętym klawiszem *alt* na klawiaturze.



Ilustracja 8: Przykład pokreślenia ścieżki, opracowanie własne

7. MOŻLIWOŚCI ROZSZERZENIA

Program jest w pełni gotowy do użytku nawet na stronie z duża ilością odwiedzających, jednak istnieje miejsce na rozwój tej pracy. Poniżej zawarte zostały kolejne pomysły oraz moduły, które mogły by zostać dodane do projektu.

7.1 Automatyczne uruchamianie skryptu

Proces pozyskiwania danych może zostać zautomatyzowany. Podejście takie umożliwiło by również posiadanie bardziej aktualnych danych w bazie danych. Można to wykonać bazując na czasie bądź zmianach w systemie plików.

W celu automatyzacji uruchamiania skryptu na bazie czasu można wykorzystać takie narzędzia jak CRON, hcron i inne. Można je tak zaprogramować żeby uruchamiały skrypt co zadany okres lub o konkretnej porze dnia lub tygodnia. Wymagało by to jednak dodanie logiki do skryptu, która zrozumieć była by w stanie czy dany plik lub część pliku została już załadowana do systemu, w celu nie duplikowania wpisów.

Drugim podejściem do tego problemu byłoby wykorzystanie nasłuchiwania zmian w systemie plików. W języku Ruby napisany został moduł *listen*, którego zadaniem jest wyszukiwanie takich zmian. Po uruchomieniu nasłuchuje zmian w zadanym folderze i uruchomić może zadane polecenia. Jest on w stanie zasygnalizować zmiany w plikach oraz jakie pliki zostały usunięte, a jakie dodane. To rozwiązanie nie wymagało by praktycznie zmian w napisanym programie jako, że moduł ten zwracałby tylko nowe pliki, przez co nie było by konieczności pisania logiki do rozróżniana już przetworzonych plików od nowych.

7.2 Filtrowanie wyników z uwzględnieniem daty

Dodając do przechowywanych danych informacje o czasie wystąpienia przejścia dało by to możliwość odpowiedzi na kolejne pytania. Na przykład:

- Jak zmienia się ruch użytkowników o konkretnych porach dnia

- Które linki zyskały na popularności w ostatnim czasie

7.3 Monitorowanie ścieżek

Posiadając dane historyczne można by zbudować system który monitoruje zadane ścieżki i byłby w stanie wygenerować listę zmian jakie nastąpiły i w jakim czasie. Daje to duże możliwości do badania tego jak zmiany, na przykład układu strony wpływają na to jak poruszają się użytkownicy po serwisie. Jeśli dany system daje możliwość pracy z danymi w czasie rzeczywistym daje to też możliwość monitorowania jakości linków i jeśli ktoś, na przykład przez przypadek coś zmieni, możliwe by było takie zmiany wychwycić i naprawić.

7.4 Parsowanie strumienia danych w serwerze Scribe

W celu optymalizacji potrzebnych zasobów do działania programu, mógłby on działać na zasadzie rozszerzenia serwera Scribe. Zlikwidowałoby to potrzebę monitorowania systemu plików, co znacząco podniosło by jego wydajność. Rozwiązanie takie możliwe jest wtedy kiedy mamy dostęp do konfiguracji takiego serwera. W zależności od konfiguracji takiego serwera w danym serwisie dało by to też możliwość zbudowania systemu który zbierał by oraz dawał dostęp do danych w czasie rzeczywistym.

8. APLIKACJE

Wykorzystując dane, które udostępnia program napisać można wiele ciekawych i przydatnych narzędzi. Rozdział ten opisuje przydatność napisanej aplikacji wizualizującej dane jak i stara się rozwinąć temat jakie aplikacje mogą zostać stworzone w oparciu o bazę danych zawierającą wagę linków pomiędzy stronami.

8.1 Wykonana aplikacja

Strona Internetowa, która wyświetla wizualizacje danych w formie ścieżek i grafów jest to pierwsza z możliwych aplikacji wykorzystania danych zapisanych przez stworzony skrypt. Daje ona możliwość badania oraz eksploracji zapisanych danych. Możliwe jest wyszukanie na nich wzorów poruszania się użytkowników po zadanym serwisie, czy specyficznych ich zachowań. Pozwolić ona może również wynaleźć strony do których użytkownicy się nie udają, dając możliwość administratorowi danej strony na udoskonalenie nawigacji czy linków.

8.2 Możliwe aplikacje

Zestaw danych, które generuje skrypt pozwolić może na napisanie wielu programów, które pomogą administratorom stron jak i jego użytkownikom. Takie jak przyspieszenie ładowania się stron czy też ciekawy model podpowiedzi kolejnych stron które mogą odwiedzić.

8.2.1 Wczesne ładowanie dokumentów

Mechanizm wczesnego ładowania linków [7] (ang. Link prefetching mechanism) wykorzystuje stan spoczynku przeglądarki do ściągnięcia dokumentów, które użytkownik może odwiedzić w przyszłości. Strona może posiadać zestaw wskazówek dla przeglądarki, które dokumenty powinna ściągnąć zaraz po załadowaniu strony. W momencie kiedy użytkownik będzie chciał odwiedzić taki dokument będzie on mógł być zaserwowany prosto z pamięci podręcznej przeglądarki. Działa to zarówno z protokołem HTTP jak i HTTPS. Wskazówki, które można przesłać do przeglądarki internetowej mogą przyjąć różne formy, takie jak:

- element HTML link:

```
<link rel=”prefetch” href=”path/to/document”>
```

- element HTML meta:

```
<meta http-equiv=”Link” content=”</path/to/document>;
rel=prefetch”>
```

- nagłówek odpowiedzi protokołu HTTP:

```
Link: </path/to/document>; rel=prefetch
```

W łatwy sposób można wykorzystać tę technikę w celu przyspieszenia ładowania się użytkownikom stron które logicznie następują po sobie na przykład:

- Galeria zdjęć/filmów/prac
- Kolejne części artykułu
- Kolejne strony wyszukiwania

Znacząco może to podnieść komfort pracy użytkownika z danym serwisem. Jednak jak wygenerować takie podpowiedzi dla stron, które nie mają tak logicznej kolejności? Wykorzystując ten mechanizm z bazą danych najczęściej odwiedzanych stron z zadanej strony wygenerowanej przez napisaną aplikację, możliwe jest wygenerowanie takich podpowiedzi dla przeglądarki w celu znacznego przyspieszenia ładowania się również takich stron użytkownikom. Jeśli struktura serwisu się zmieni i użytkownicy zaczną przechodzić częściej na inne strony, wygenerowana podpowiedź, również zmieni się tak szybko jak często dany skrypt będzie uruchamiany dla danego serwisu.

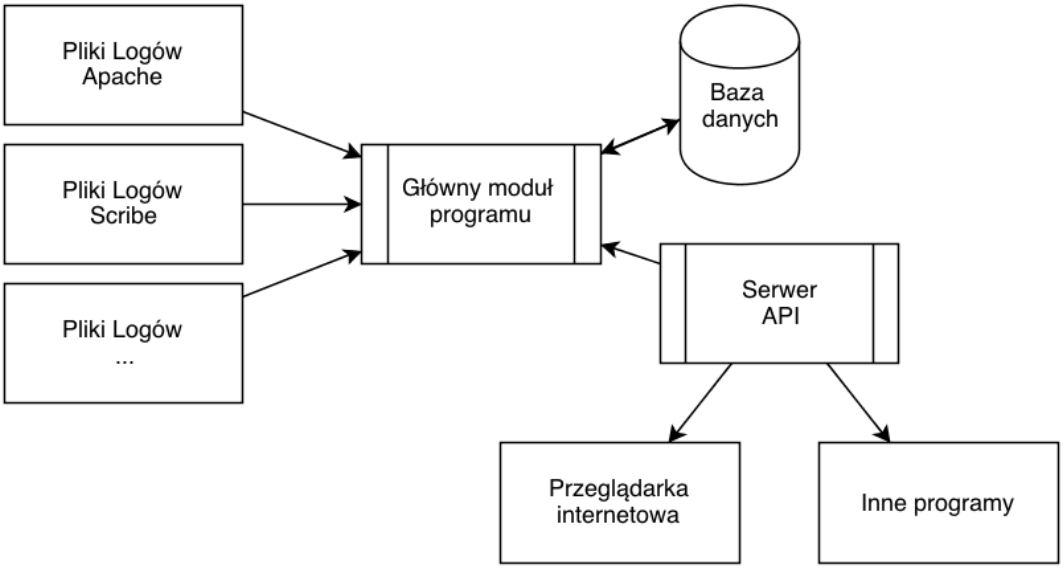
8.2.2 Tworzenie tuneli

Moduł generujący ścieżki przejść wykorzystać również można do zaimplementowania tak zwanego „Inni oglądali również”, który dostosowywał by się do tego co rzeczywiście inni użytkownicy zobaczyli i jako cała społeczność wybrali za najciekawszy cel przejścia z danej strony. Stworzyło by to pewnego rodzaju tunele, w których użytkownicy mogli by się poruszać. Co ciekawe jeśli naprawdę uważali by, że podpowiadana strona jest tą, którą wybrali by za kolejną, i kliknęli na proponowany link podnosiło by to jednocześnie jego wagę.

9. IMPLEMENTACJA

Implementacja systemu znajduje się w folderze PathFinder oraz jego podfolderach. Najważniejsze pliki systemu to:

- pathfinder.rb – skrypt uruchomienia systemu
- config.yaml – przykładowa konfiguracja dla aplikacji
- PathFinder.rb – część odpowiedzialna za łączenie z bazą oraz odczyt oraz zapis danych do niej
- server.rb – prosty serwer HTTP
- index.html – część prezentacyjna systemu



Ilustracja 9: Diagram elementów programu

Ilustracja 9 ukazuje diagram poszczególnych elementów składowych programu. Najważniejszy jest główny moduł programu, który odpowiedzialny jest za zbieranie danych jak i ich pobieranie z bazy danych. System został napisany w oparciu o język

Ruby oraz JavaScript. Do parsowania logów został również użyty zestaw standardowych narzędzi, w które wyposażony jest system z rodziny unix takich jak: tail, head, grep, sed, perl, cat oraz gzcat.

tail – polecenie wypisujące podaną ilość ostatnich linijek pliku lub potoku. Może być stosowane jako narzędzie diagnostyczne do sprawdzania zmian w plikach.

head – podobne polecenie jak tail jednakże wypisuje podaną ilość pierwszych linijek pliku lub potoku

grep – jeden z podstawowych programów wchodzących w skład systemu Unix. Służy do znajdowania w strumieniu wejścia linii zawierających ciąg znaków pasujących do danego wyrażenia regularnego.

cat – (ang. *catenate* – łączyć) również jest poleceniem systemu Unix. Służy ono do łączenia plików oraz wyświetlania zawartości plików.

gzcat – podobnie jak polecenie cat z tym, że pracuje z plikami skompresowanymi poleceniem gzip

9.1 Skrypt uruchomienia systemu

Umożliwia on pracę z napisanym programem, może uruchomić procedurę zbierania danych, naprawy danych oraz serwer. Jeśli przekażemy -h jako parametr wyświetli on listę parametrów, które przyjmuje. Są to:

- -c lub –config – ścieżka do pliku konfiguracyjnego
- -m lub –mode – Tryb w jakim uruchomiony ma zostać program
- -s lub –site – Witryna z którą pracować ma program, wymagane tylko w trybie fix

Dostępne tryby to:

- etl – tryb ten uruchomi program w celu zbierania danych
- server – tryb ten uruchomi prosty serwer HTTP potrzebny do realizacji wizualizacji
- fix – tryb napisany w celu naprawienia danych w bazie danych w razie takiej potrzeby

Tryb etl oraz server zostaną opisane w kolejnych podrozdziałach. Tryb fix został napisany z racji tego, że pliki logów zapisywanych przez serwery Scribe wykorzystywane w firmie Wikia nie przechowują informacji na jaką udał się użytkownik w formie jej nazwy, a w formie unikatowego identyfikatora, w celu zminimalizowania ilości danych, które przechowywane są na tych serwerach. W tym trybie program wymaga podania witryny, gdyż naprawienie danych dla całej bazy było by zbyt intensywne dla komputera na którym skrypt jest uruchamiany oraz dla serwerów Wikii odpowiedzialnych za obsługę API. W trybie tym pobierane są z bazy danych unikatowe identyfikatory wszystkich stron, dla której nie ma nazwy w postaci tytułu i generowane jest zapytanie do serwerów Wikii o taką nazwę. Proces ten w innym wypadku nie jest wymagany i może zostać pominięty.

9.2 Część zbierająca dane

Ten fragment systemu odpowiedzialny jest za połączenie z bazą danych oraz zapis i odczyt danych z niej. Z jednej strony konwertuje ona dane, które dostarczane są z odpowiedniego adaptera i zapisuje je w odpowiednich tablach bazy danych. Z drugiej daje ona usystematyzowany zestaw metod do pobierania danych.

W bazie danych tworzona jest tabela dla każdego serwisu, który poddaje zostaje badaniu. Zostało to zrobione z powodu zwiększenia wydajności zapytań jak i łatwości przechowywania danych dla serwisów, które zawierają wiele pod serwisów. Dzięki temu parsując jeden plik logów stworzyć możemy tabele dla wszystkich portali. Jeśli w logu zostanie znalezione wiele wpisów dla danego serwisu, są one grupowane w ten sposób aby wykonać jak najmniej zapytań do bazy danych tj. dla każdego serwisu wykonywane są dwa zapytania:

- CREATE TABLE ... – w celu utworzenia tabeli jeśli taka jeszcze nie istnieje w bazie danych

- INSERT INTO ... – w celu dodania zestawu znalezionych przejść

Podczas ładowania danych tworzona jest instancja adaptera, który dla każdego sparsowanego pliku wykonywać będzie blok kodu odpowiedzialny za przetworzenie tych danych na zapytanie SQL.

9.3 Adaptery

Napisane zostały dwa adaptery dla logów pochodzących z serwerów Scribe oraz serwerów Apache. Głównie składają się one z wyrażeń regularnych, które przekształcają logi do stanu w którym zrozumiałe są dla części odpowiedzialnej za zapisywanie danych do bazy danych. W obu adapterach dodano możliwość uruchomienia wielu wątków w celu przyspieszenia ekstrakcji danych

```
while(threads.count >= @threads) do sleep 0.5  end
threads << Thread.new {...
```

Jeśli program nie przekroczył maksymalnej ilości wątków tworzy on nowy i dodaje on go do listy uruchomionych wątków. W przeciwnym wypadku czeka on na moment w którym będzie mógł on stworzyć nowy tj. na moment w którym poprzedni wątek skończy prace z plikiem.

Dla adaptera Apache najważniejsza linią kodu to

```
sed –E "s/^(\w*)\s–>\s(\w*)$/\1;\2/"
```

Wykorzystując zewnętrzny program sed do ekstrakcji danych z pliku do potrzebnego formatu. Sed to edytor strumieniowy zawarty w systemach unixowych. Służy on do przetwarzania plików tekstowych. Znajduje on zastosowanie przy pisaniu programów konwertujących.

Logi pochodzące z serwera Scribe mogą być bardziej skomplikowane, dlatego zdecydowano się na użycie języka Perl. Jest to interpretowany język programowania,

początkowo przeznaczony głównie do pracy z danymi tekstowymi, obecnie używany do wielu innych rozwiązań. Wzorowany jest na takich językach jak C, sed, awk i wielu innych.

```
perl –ne 'use URI::Escape; if($_ =~ /x=([^&]
+).*&a=(\d+).*n=(?:0|14)&r=.*?\/(?=\p{Lu})(\p{Lu}[^&]+)/){ print
"$1;", uri_unescape(uri_unescape($3)), ";$2\n"}; '
```

Linijka ta sprawdza poprawność danych oraz przekształca je do formatu potrzebnego dla dalszej części systemu. Podwójne wywołanie funkcji uri_unescape spowodowane jest tym, że logi takie często są podwójnie przetwarzane przez wyższe warstwy aplikacji.

9.4 Serwer

Część ta to implementacja prostego serwera HTTP. Umożliwia ona serwowanie danych z bazy danych jak i serwowanie części odpowiedzialnej za wizualizacje. Znajduje się ona w pliku server.rb.

```
server = TCPServer.open(port)
```

Linia ta tworzy serwer protokołu TCP i nasłuchuje na zadanym porcie. Port pochodzi z pliku konfiguracji i jest w niej wymagany.

```
loop do
  Thread.start(server.accept) do |client|
    get = client.gets.strip.split[1][1..-1]

    if get.include? 'api'
      route, command, site, page = get.split('/')
    end
...
```

W tej części tworzona jest nieskończona pętla, która nie pozwoli programowi zakończyć działania i umożliwi ciągłe nasłuchiwanie nadchodzących połączeń. Dla każdego nowego połączania tworzony jest nowy wątek, a następnie żądanie rozbijane jest na mniejsze części w celu jego dalszej analizy. Jeśli zapytanie zawiera słowo 'api', rozbijane jest ono na części potrzebne do zrealizowania żądania:

- *command* – komenda do wykonania np.: get_path lub get_sites
- *site* – witryna, dla której pobrać dane
- *page* – strona, dla której pobrać dane

Po przetworzeniu zapytania i pobraniu odpowiednich danych z bazy bądź plików, tworzona jest odpowiedź:

```
client.print get_headers status, ret.bytesize, mime
client.print ret
```

Generowany jest nagłówek odpowiedzi z odpowiednim statusem protokołu HTTP, wielkością w bajtach odpowiedzi oraz odpowiedniego opisu zawartości pliku MIME. Zaimplementowane statusy HTTP to:

- odpowiedź poprawna - HTTP/1.1 200 OK
- nie znalezienie dokumentu - HTTP/1.1 404 Not Found
- inne błędy serwera - HTTP/1.1 500 Internal Server Error

9.5 Część odpowiedzialna za wizualizacje

Część ta została napisana w HTML, CSS oraz JavaScript. Główne części programu to:

- script.js – odpowiedzialny za obsługę ogólnych aspektów programu takich jak podpowiadanie nazw serwisów i stron dostępnych w bazie danych, obsługę stałego linku do wizualizacji czy też tworzenia wizualizacji z podanych danych za pomocą biblioteki d3.
- getpath.js – odpowiedzialny za pobieranie danych na temat konkretnej ścieżki oraz wyświetlaniu jej prezentacji

- getmostvisited.js – odpowiedzialny za pobieranie danych na temat wszystkich stron na które użytkownicy przeszli z zadanej strony oraz wyświetlenia ich prezentacji

- graph.js – odpowiedzialny za pobieranie danych na temat wszystkich ścieżek z zadanej strony, obsługę podświetlania ścieżek oraz wyświetlania prezentacji

Pliki getpath.js, getmostvisited.js, graph.js odpowiedzialne są za obsługę takich zdarzeń jak wybór strony do prezentacji, załadowanie odpowiednich danych do wizualizacji, generowanie wizualizacja za pomocą funkcji pomocniczej, która znajduje się w pliku script.js.

Najważniejszą funkcją w systemie generowania prezentacji jest funkcja getD3 – która z zadanych danych wygeneruje odpowiednią prezentację. Przyjmuję ona 5 parametrów:

```
pf.getD3 = function(pages, links, selector, number, onMeta){ …
```

- *pages* – lista stron do dla danej wizualizacji

- *links* – lista linków pomiędzy stronami

- *selector* – selektor jakiego funkcja ma użyć w celu odnalezienia miejsca, w którym wyrysowana ma zostać prezentacja.

- *number* – wartość logiczna odpowiadająca za to czy funkcja ta ma dodać kolejny numer dla danej strony na wizualizacji

- onMeta – funkcja która ma być wykonana w momencie kiedy użytkownik kliknie na stronę z wciśniętym klawiszem *alt* na klawiaturze

Kolejno ustalane są dane do prezentacji:

```
var width = 1150,
  height = 800,
  linkColor = "#DA2A55",
```

```
  nodeColor = "#DCE9BE",
  force = d3.layout.force()
    .size([width, height])
    .charge(-20)
    .gravity(0)
    .linkDistance(function(d){
      return 400-~~Math.max(100, Math.min(~~(d.weight*10),300));
    })
    .linkStrength(function(d){
      return ~~Math.max(10, Math.min(~~(d.weight/10), 20));
    })
    .on("tick", tick),
  drag = force.drag().on("dragstart",
     dragstart).on('dragend', dragend),
  zoom = d3.behavior.zoom()
    .translate([0, 0])
    .scale(1)
    .scaleExtent([.1, 15]),
...
```

width, height – to zmienne, które ustalają szerokość oraz wysokość elementu svg który służy do prezentacji

linkColor, nodeColor – to zmienne trzymające wartość dla kolorów dla linku oraz strony odpowiednio

d3.layout.force() - umożliwia on dodanie fizyki do prezentacji i skonfigurować jej do własnych wymagań. W celach prezentacji przejść użyte zostały następujące opcje:

- *size* – w celu określenia wielkości świata fizycznego
- *charge* – ustala z jaką siłą oddziaływać maja na siebie strony. Wartość ujemna oznacza że strony będą się nawzajem odpychać od siebie
- *gravity* – pozwala określić jak silna będzie grawitacja w centrum prezentacji. Wartość 0 wyłącza grawitację
- *linkDistance* – pozwala określić jak długie mają być połączenia pomiędzy stronami. W tej prezentacji jest ona zależna od ilości użytkowników, którzy przeszli z jednej strony na drugą.

- *linkStrength* – pozwala określić siłę/sztywność linku, ta wartość również zależna jest od ilości użytkowników, którzy przeszli z jednej strony na drugą

- on('tick', function … - pozwala wykonać jakoś akcje w każdej klatce animacji prezentacji

force.drag() – pozwala podpiąć akcję pod zdarzenia związane z myszką takie jak kliknięcie na element, przesuwanie elementu czy też jego puszczenie.

d3.behaviour.zoom() - pozwala dodać opcję przybliżania, oddalania, przesuwania prezentacji oraz skonfigurować to zachowanie:

- translate – początkowe przesunięcie prezentacji

- scale – początkowa skala prezentacji

- scaleExtent – maksymalna oraz minimalna wartość oddalenia oraz przybliżenia

W dalszej części funkcji strony oraz linki dodawane są do elementu svg w celu wyrysowania ich przez przeglądarkę. W tym celu napisana została funkcja

```
addElements(pages, links);
```

która dla wszystkich stron oraz linków tworzy odpowiednie elementy, ustala im odpowiednie atrybuty i dodaje do elementu svg. Ostatni krok to:

```
force
  .nodes(pages)
  .links(links)
  .start();
```

gdzie elementy animacji dodawane są do świata fizyki oraz cała prezentacja zostaje uruchomiona. Po uruchomieniu wizualizacja należy poczekać parę sekund w celu wyliczenia położenia każdego elementu oraz w celu stabilizacji animacji.

9.6 Test

Część programu odpowiedzialna za zbieranie danych została przetestowana manualnie poprzez spreparowane pliku logu dla którego znane są dane wejściowe oraz znane, są ścieżki jakie powinny zostać wygenerowane. Wyniki testu są pozytywne tj. dla zadanych danych otrzymane zostały oczekiwane ścieżki.

Test wykonany dla pliku *view.log-00-00.1375833600.gz* oraz wiki *starwars*.

Ścieżka oczekiwana	**Ścieżka otrzymana**
Death Star	Death Star
Yuuzhan_Vong	Yuuzhan_Vong
Quoreal	Quoreal

Tabela 1: Test dla strony Death Star

Ścieżka oczekiwana	**Ścieżka otrzymana**
Lightsaber	Lightsaber
Lightsaber_combat	Lightsaber_combat
Form_II:_Makashi	Form_II:_Makashi
Form_III:_Soresu	Form_III:_Soresu
Form_I:_Shii-Cho	Form_I:_Shii-Cho

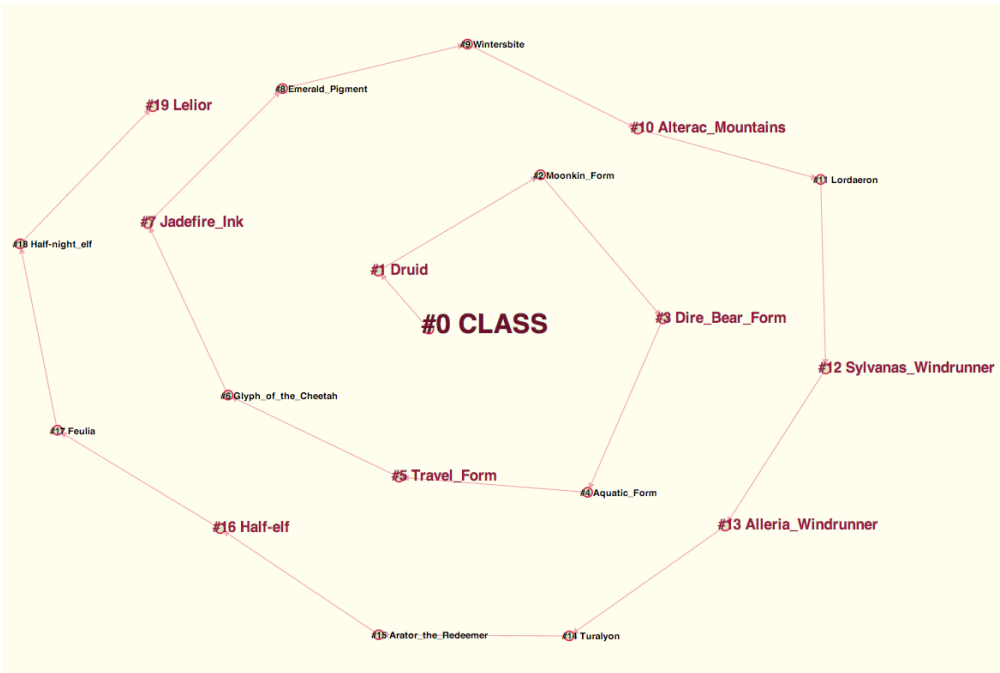
Tabela 2: Test dla strony Lightsaber

10. WYNIKI

Szczegółowej analizie poddane zostały dwie wiki przechowywane na serwerach firmy Wikia. Daje to możliwość sprawdzenia jakości otrzymanych wyników na bardzo dużej ilości odwiedzin jak i na zbadaniu jak algorytm wyszukiwania ścieżek zachowuje się z dwoma zupełnie odmiennymi społecznościami. Analizowane dane są z dnia: 20 sierpnia 2013. Algorytm zawsze jako następną stronę wybiera tę, na która ilościowo przeszło najwięcej użytkowników, także przy kolejnych analizowanych stronach nie ma danych ilościowych aby zwiększyć czytelność danych.

10.1 World of Warcraft Wiki

wowwiki.com – to wikia dedykowana skatalogowaniu świata Warcraft stworzonego przez Blizzard Entertainment z naciskiem na grę World of Warcraft [8]. Opisuje ona całą serię gier, książki, nowele i inne źródła na temat około tej gry. Posiada ona ponad 98000 stron i około 500 tys. odwiedzin dziennie. Jest to jedna z większych i najstarszych wiki, która serwowana jest z serwerów Wikii. Test dla strony „Class” - klasa w świecie „World of Warcraft” przedstawia profesję postaci, w którą gracz się wciela.



Ilustracja 10: Ścieżka wygenerowana dla strony Class, opracowanie własne

Tytuł strony	Krótki opis strony
Class	klasa to główny styl charakteru postaci w grze, która opisuje jakich broni oraz pancerzy może używać oraz jakie posiada zdolności, moce, umiejętności oraz czary
Druid	jest to jedna z klas dostępnych w grze, która specjalizuje się w rzucaniu czarów oraz zmienianiu swojej formy
Moonkin Form	jedna z form, w która zmienić może się postać o klasie druid
Dire Bear Form, Aquatic Form, Travel Form	to kolejne formy, w które może zmieniać się postać klasy druid w celu dodania pewnych zdolności oraz mocy
Glyph of the Cheetah	za pomocą tego przedmiotu druid, może zmienić swoją podróżniczą postać w formę Geparda
Jadefire Ink, Emerald Pigment, Wintersbite	kolejne przedmioty, których klasa druid może używać
Alterac Mountains	jest to miejsce w grze w której gracze znaleźć mogą przedmiot *Wintersbite*
Lordaeron	jest to kontynent, na którym znajduje się miejsce *Alterac Mountains*
Sylvanas Windrunner	jest to jedna z władczyń na kontynencie *Lordaeron*
Aleria Windrunner	jest to postać w grze, która jest siostrą *Sylvanas Windrunner*
Turalyon	jest to postać, która jest kochankiem *Aleria Windrunner*
Arator The Redeemer	syn dwóch powyższych postaci

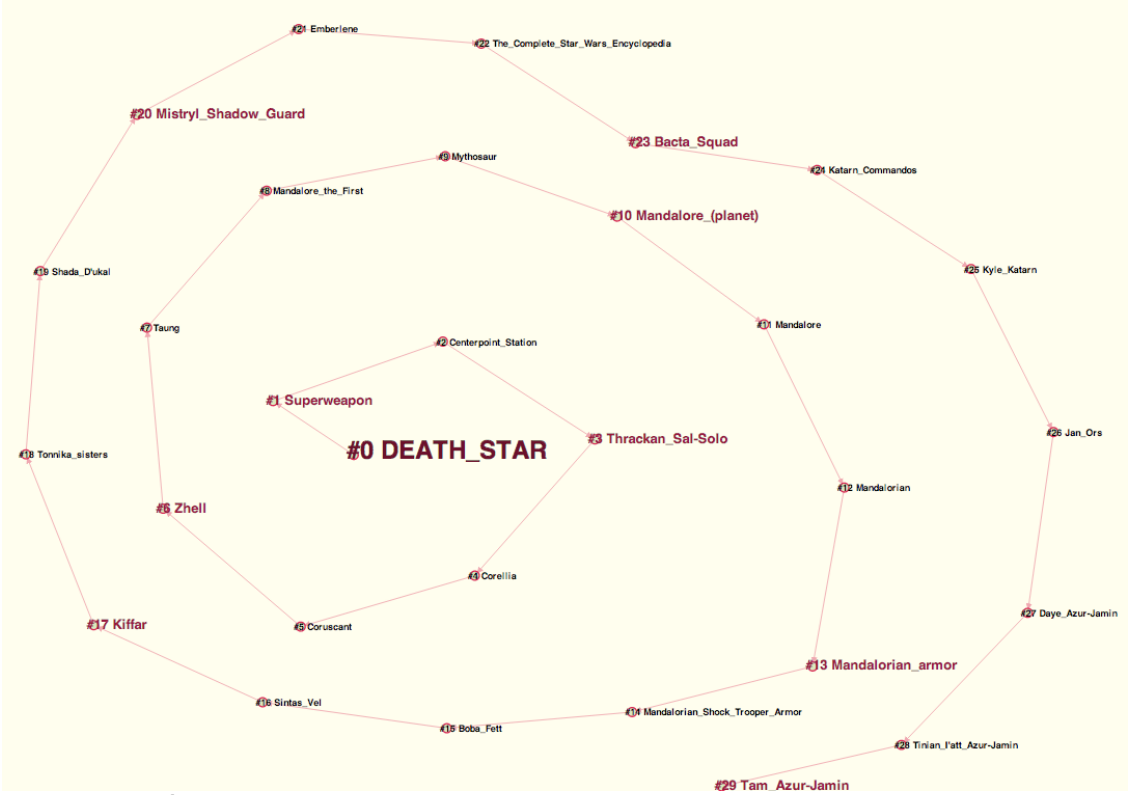
Half Elf	rasa postaci *Arator The Redeemer*
Feulia	jest to postać, która jest rasy podobnej do *Half Elf*
Half Night Elf	inna forma rasy *Half Elf*
Lelior	postać w grze oraz przedstawiciel rasy *Half Night Elf*

Tabela 3: List przejść dla witryny o „World of Warcraft" dla strony „Class”

Z tabeli 3 wyczytać można, że użytkownicy którzy odwiedzili stronę Class na wiki o grze War of Warcraft, bardzo chętnie odwiedzali stronę o klasie druid. Następnie zgłębiali tę wiedzę poprzez wchodzenie na kolejne strony, które łączyły się tematycznie z poprzednimi.

10.2 Wookiepedia

starwars.wikia.com - Wookiepedia to wikia która dedykowana jest informacjom na temat filmów, książek oraz gier „Star Wars” [9]. Nazwa to połączenie nazwy „Wikipedia” oraz słowa Wookie – popularna rasa w świecie Star Wars. Posiada ona ponad 106 tys. stron oraz około 550 tys odwiedzin dziennie.



Ilustracja 11: Ścieżka wygenerowana dla strony Death Star, opracowanie własne

Tytuł strony	Krótki opis strony
Death Star	jest to wojskowa baza wielkości księżyca uzbrojona w laser zdolny niszczyć całe planety
Superweapon	jest to broń przewyższająca mocą broń konwencjonalną, z siła zdolną niszczyć planety oraz całe układy słoneczne często wymagając ogromnych nakładów do ich stworzenia
Centerpoint Station	jednostka wojskowa zdolna przenosić całe planety
Thrackan Sal-Solo	utworzył on ruch znany jako Liga Ludzi i zjednał się z trójcą Saccorian którzy posiadali kontrolę nad Centerpoint Station
Corellia	główna planeta układu Corellian, która została zaatakowana przez trójcę Saccorian
Coruscant	planeta-miasto, wielu wierzy że stąd przesiedlono ludzi na planetę Corellia
Zhell	to ludzie oryginalnie zamieszkujący planetę Coruscant
Taung	kolejna rasa, która zamieszkiwała planetę Coruscant
Mandalore the First	przedstawiciel rasy Taung – jako pierwszy podbił planetę Mandalore

Mythosaur	Rasa zwierząt, która zamieszkiwała planetę Mandalore
Mandalore (planet)	piąta planeta układu Mandalore
Mandalorian armor	zbroja, która jest najbardziej rozpoznawalnym symbolem kultury Mandalorskiej
Mandalorian Shock Trooper Armor	jeden z wielu modeli zbroi, noszonej przez rasy zamieszkujące układ słoneczny Mandalore
Boba Fett	łowca nagród pochodzący z planety *Mandalore* oraz noszący zbroję *Mandalorian Shock Trooper Armor*
Sintas Vel	łowca nagród oraz przedstawiciel rasy Kiffar oraz żona Boba Fett
Kiffar	rasa podobna do człowieka zamieszkująca planety Kiffu oraz Kiffex
Tonnika sisters	przedstawicielki rasy Kiffar, znane głównie za podszywanie się pod innych w celach rabunków i wymuszeń
Shada D'ukal	zabójczyni, która podszyła się pod jedną z sióstr Tonnika w celu wynajęcia statku do przewiezienia prototypu super-lasera
Mistryl Shadow Guard	elitarna jednostka zabójczyń wysłana z ich rodzimej planety Emberlane w celu zarobku pieniędzy by pomóc zniszczonej

	społeczności z ich planety
Emberlene	planeta najbardziej znana z *Mistryl Shadow Guard*

Tabela 4: Tabela przejść dla witryny o „Star Wars" dla strony „Death Star"

Test na stronie poświęconej serii Star Wars, również ukazuje że zastosowana metoda poszukiwania ścieżek użytkowników przynosi interesujące wyniki, które mogłyby być wykorzystane w celu jeszcze lepszego ulepszenia tego serwisu.

11. PODSUMOWANIE

Analizując wyniki pracy zauważyć można, że poprzez analizę statystyczną odwiedzin zadanej witryny stworzyć można ciekawe i przydatne narzędzia do zarządzania nawigacją strony oraz jej usprawnienia, treścią strony oraz do usprawnienia wydajności ładownia się stron w przeglądarkach użytkowników. Projekt można w prosty sposób rozszerzać o nowe funkcjonalności, usprawnić go wydajnościowo oraz ulepszać już istniejące funkcje.

Cel pracy został osiągnięty. W szczególności udało się wykazać, że bez naruszania prywatności użytkowników, możliwe jest badanie ich poruszania się po witrynie internetowej. Zbudowany program umożliwia nie tylko zbieranie danych, ale również ich analizę poprzez czytelne grafy oraz tabele. Daje to możliwość analizy witryny internetowej z poziomu jego odwiedzających. Napisany prosty serwer umożliwia również użycie tych danych w innych programach poprzez prostą implementację API opartą o architekturę REST. Testy wykazały, również, że generowane ścieżki mają semantyczne znaczenie i strony po sobie następujące, łączą się ich znaczeniem czy tematem.

Projekt ten będzie dalej rozwijany w celu jego udoskonalenia oraz wdrożenia w firmie Wikia, która chce wykorzystać dane zbierane przez ten system do wizualizacji jej bazy danych artykułów, ulepszenia stron użytkowników, jak i wydajności ładowania się stron.

12. LITERATURA

1. Bezpieczeństwo w sieci 10/08/2013
   http://www.google.com/intl/pl_pl/goodtoknow/online-safety/
2. Ben Laurie, Peter Laurie, 2003. Apache. Przewodnik encyklopedyczny. Wydanie III. Wydawnictwo: Helion
3. Scribe 18/08/2013
   https://github.com/facebook/scribe/wiki/Scribe-Overview
4. David Flanagan, Yukihiro Matsumoto. 2008. Ruby. Programowanie. Wydawnictwo: Helion
5. YAML 15/08/2013
   http://pl.wikipedia.org/wiki/YAML
6. Ruby 18/08/2013
   http://ruby-doc.org/core-1.9.3/
7. Link Prefetching 20/08/2013
   https://developer.mozilla.org/en-US/docs/Link_prefetching_FAQ
8. World of Warcraft wiki 05/09/2013
   http://wowwiki.com
9. Star Wars wiki 06/09/2013
   http://starwars.wikia.com/

13. ZAŁĄCZNIK

Płyta CD z następującą zawartością:

- tekst pracy w formie pdf, doc oraz odt
- pliki źródłowe
- przykładowe pliki konfiguracji
- przykładowe pliki logów
- instrukcja obsługi