**Autorzy:**

Wojciech Urbański

Marcin Waligórski

Daniel Wegner

Łukasz Wróblewski

Inteligentne wyszukiwanie informacji

WikiWizir - Wizualizacja Wikipedii

Opis zadania

Naszym zadaniem projektowym było stworzenie wizualizacji powiązań artykułów angielskiej Wikipedii w postaci Simple. Zawiera ona 56565 artykułów oraz 2248532 połączeń pomiędzy nimi. Pomimo i tak pomniejszonej objętości w stosunku do pełnej Wikipedii, stanowi ona dość trudne zadanie do zrealizowania - zarówno jeśli chodzi o przedstawienie informacji w czytelnej formie, jak i względy optymalizacyjne.

Rozpoznanie problemu

Pracę nad projektem rozpoczęliśmy od zapoznania się z dostępnymi gotowymi rozwiązaniami wizualizującymi Wikipedię oraz przeglądem istniejących bibliotek wspomagających rysowanie grafów i drzew. Już podczas korzystania z istniejących programów pojawił się pierwszy z problemów, z którymi musieliśmy się zmierzyć. Otóż żaden z programów nie radził sobie z danymi wejściowymi o tak dużym rozmiarze. Do każdego z nich staraliśmy się dostosować dane do jednego z obsługiwanych formatów – najczęściej miały one format listy wierzchołków w postaci: indeks i nazwa oraz listy krawędzi, opisanych jako 2 indeksy wierzchołków, które łączą. Niezależnie od technologii w jakiej programy był wykonany; Java, C, C++, HTML z CSS i Javascript, żaden z nich nie potrafił jednocześnie załadować takiej ilości danych. Kończyło się to całkowitym wyczerpaniem całej dostępnej pamięci (nawet do 4GB) lub po prostu brakiem reakcji ze strony aplikacji. W ramach testów próbowaliśmy wizualizować tylko część danych np. 5-10 tys. wierzchołków. Jeśli nawet, już któryś z programów sobie poradził to wykres był tak gęsty lub skomplikowany, że nie dało się z nic z niego odczytać lub w ogóle korzystać z programu.

Po pierwszym etapie, podjęliśmy decyzję, że spróbujemy zaimplementować samodzielnie jeden z algorytmów służących do rysowania grafów i dodatkowo wykorzystamy istniejącą bibliotekę do wspomagania samej wizualizacji. Wybór padł na dość często wykorzystywany algorytm force-based layout (sam algorytm został opisany w dalszej części sprawozdania). Kolejnym krokiem był wybór technologii w jakiej zostanie wykonany projekt. Podczas przeglądu bibliotek zdecydowaliśmy się na skorzystanie z biblioteki d3.js pozwalającej wizualizować dane z użyciem SVG. Fakt, że została ona napisana w Javascript w oczywisty sposób wpłynął na fakt, że sam interfejs aplikacji postanowiliśmy stworzyć w oparciu o HTML 5 i CSS3 oraz część back-endową opartą na framework’u Fuel w języku PHP. Dodatkowym atutem d3.js jest to, że posiada możliwość wizualizacji grafów z wykorzystaniem wybranego przez nas algorytmu. Podczas tworzenia naszej wersji wzorowaliśmy się na dostarczonej z biblioteką implementacji, dostosowując ją do naszego projektu, głównie w kwestiach optymalizacyjnych.

To wszystko jednak nie wystarczyło do poradzenia sobie w pełni z ilością danych do wyrenderowania. Po ostatecznych konsultacjach z prowadzącym, uzgodniliśmy, że będziemy wizualizować mniejsze podgrafy wszystkich danych - tzn. użytkownik będzie mógł wybrać, jaki artykuł go interesuje i następnie zostanie narysowany graf połączeń tego artykułu z innymi. Nasza aplikacji umożliwi wtedy przejście do kolejnego artykułu i wyświetlenie jego sąsiadów. Umożliwi to klarowniejszą prezentację dostępnych danych (mniej danych do przetworzenia dla użytkownika naraz, czytelniejsza reprezentacja).

Algorytm

W projekcie wykorzystaliśmy algorytm Force-Based Layout. Służy on do rysowania grafów w dwu i trzy wymiarowych przestrzeniach w taki sposób, aby każda krawędź, była mniej więcej równej długości i było jak najmniej przecięć. Osiąga on to poprzez traktowanie krawędzi jak sprężyn, a wierzchołków jak naładowane elektrycznie cząsteczki. Cały graf jest następnie symulowany jak system fizyczny. Siły na siebie oddziaływają przyciągając lub oddalając wierzchołki od siebie. Te akcje są powtarzane iteracyjnie, aż układ osiągnie stan spoczynku, tzn. zmiany w każdej iteracji będą coraz mniejsze. Dodatkowo algorytm implementuje siłę pseudo-grawitacyjną, która utrzymuję graf w widocznej strefie i uniemożliwia jego ucieknięcie poza ekran.

Zaletami wybranego przez nas algorytmu są głównie dobre rezultaty dla małych i średnich grafów z jakimi mamy do czynienia w naszym projekcie. Dodatkowo jest on dość elastyczny i umożliwią rozbudowę o dodatkowe parametry. Wadami za to jest, że niektóre większe grafy potrzebują dłuższej chwili na wygenerowanie ale jest to akceptowalny czas oraz w naszym przypadku nie takich grafów aż tak dużo.

Rezultat

Rezultatem projektu jest aplikacja, składająca się z bazy danych, w której przetrzymywane są artykuły oraz połączenia do nich, serwera PHP, który służy do wyciągania danych z bazy MySql oraz interfejsu w HTML 5 i CSS3 oraz Javascript. Obsługa aplikacji polega na wpisaniu nazwy artykułu w polu wyszukiwarki (która posiada opcję autouzupełniania ze sprawdzeniem poprawności – czy taki artykuł jest dostępny). Następnie, po wciśnięciu przycisku „Pokaż” wyświetlony zostanie graf z artykułami powiązanymi z wyszukiwanym przez nas artykułem. Po lewej stronnie znajduje się lista powiązanych artykułów, poruszając się po niej podświetlają się odpowiednie wierzchołki i połączenia z nich wychodzące ( tak samo dzieje się gdy, przesuwamy kursor bo wyrenderowanym grafie). Klikając w element na liście lub odpowiadający mu wierzchołek na grafie, zostaniemy przeniesieni do kolejnego grafu, tym razem sąsiadów wybranego przez nas, przed chwilą, artykułu. Dodatkowo przy każdej pozycji na liście jest przycisk umożliwiający wyświetlenie danego artykułu w Wikipedii w osadzonym oknie co bardzo przydaje się przy przeglądaniu grafu. Na górnej belce jest także przycisk ustawień, którym możemy skonfigurować parametry działania algorytmu takie jak początkowy ładunek wierzchołków oraz odległość pomiędzy nimi. Przeglądanie wszystkich grafów polega na wygenerowaniu „hash’a” z jego nazwy i doklejenie do adresu strony jako parametr. Dzięki temu, korzystając z przycisków „wstecz’ i „dalej” w przeglądarce, możemy wygodnie poruszać się po historii przeglądanych przez nas artykułów. Łatwo też zapisać sobie też interesujący nas graf w formie linku i wrócić do niego później. Warto wspomnieć, że cała aplikacja działa w oparciu o technologię AJAX, dzięki czemu nie jest wymagane przeładowywanie strony.

Uruchomienie projektu

Serwer www znajduje się na wirtualnej maszynie. Należy w parametrach wirtualnej maszyny dodać nową kartę sieciową jako mostkowaną. A następnie skonfigurować IP zewnętrzne poleceniem:

[root@archlinux-srv](mailto:root@archlinux-srv) $ ifconfig eth1 adres

[root@archlinux-srv](mailto:root@archlinux-srv) $ ifconfig eth1 up

W przypadku niedziałającego serwera HTTP / MySQL należy uruchomić go poleceniem:

[root@archlinux-srv](mailto:root@archlinux-srv) $ /etc/rc.d/httpd start

[root@archlinux-srv](mailto:root@archlinux-srv) $ /etc/rc.d/mysql start

Parametry konta root:

hasło: iwi

Parametry użytkownika MySQL:

użytkownik: iwi

hasło: iwi

host: dowolny

Backend oparty jest o Framework FuelPHP, który wymaga mod\_rewrite i dodania AllowOverride All w konfiguracji wirtualnego hostu.

Główny katalog serwera www to /srv/http/public