UNIWERSYTET JAGIELLOŃSKI
Wydział Fizyki, Astronomii i Informatyki Stosowanej

kierunek: Informatyka Stosowana
specjalność: Modelowanie i Animacja 3D

Natalia Stanko

Współdzielenie i edycja dokumentów w chmurze

Praca magisterska napisana pod kierunkiem dr. Andrzeja Łachwy

Kraków 2012

Spis treści

1 . Wstęp

1.1 . Uzasadnienie wyboru tematu

W dzisiejszych czasach - w dobie internetu, komunikujemy się z innymi ludźmi w większości za pomocą słowa pisanego. Każdego dnia przesyłamy drogą elektroniczną ogromną ilość informacji. Tworzymy dokumenty tekstowe prywatnie i w pracy, dzielimy się nimi.

Zapewne każdy z nas słyszał czy korzystał z darmowego pakietu biurowego udostępnianego nam przez firmę Google Inc. Wielu korzysta z powyższych narzędzi codziennie i ja również należe do tego grona. Ile jednak osób zainteresował sposób działania dokumentów on-line umożliwiający edycję tekstu w tym samym czasie przez wielu zalogowanych użytkowników? Mnie intrygowało to od dłuższego czasu. Ostatecznie istnienie Google Docs natchnęło mnie do stworzenia własnego edytora tekstu on-line zapewniającego zapis i wymianę informacji przez wielu użytkowników jednocześnie. Stąd zrodził się pomysł na temat pracy magisterskiej. Innymi słowy temat wybrałam z własnej ciekawości, chcąc poznać nowe technologie, sprawdzić sposoby wykonania, możliwe rozwiązania i algorytmy, które są istotą mechanizmu omawianej aplikacji. Problematyka wydawała mi się od samego początku bardzo trudna. Z pewnością jako temat pracy magisterskiej nadawałaby się sama analiza i porównanie działania istniejących już edytorów umożliwiających edycję tekstu czy obrazów w czasie rzeczywistym. Zagadnienie to zdawało mi się jednak równie interesujące – i nie zawiodłam się. Podjęłam wyzwanie, napotkałam wiele przeszkód na drodze do celu. Nawet jeśli nie znalazłam wszystkich odpowiedzi na zadane sobie w tej pracy pytania, a stworzona przeze mnie aplikacja nie jest tworem idealnym ani do niego zbliżonym, odkryłam wiele nowych dla mnie możliwości i narzędzi pracy. Wyzwanie to ubogadziło mnie o cenne doświadczenia mogące się przydać w przyszłości.

Komunikacja z definicji jest procesem wymiany informacji i temu właśnie procesowi poświęcony został ten tekst i mój czas potrzebny do stworzenia działającej i spełniającej cele pracy aplikacji internetowej.

1.2 . Zasadnicze cele pracy

Celem tej pracy jest stworzenie narzędzia – edytora tekstu działającego w przeglądarce interntowej, który umożliwia:

1. Rejestrację, edycję konta użytkowników oraz ich autoryzację
2. Tworzenie dokumentu tekstowego, zapisanie go pod dowolną nazwą z możliwością odszukania na liście dokumentów użytkowika

3. Udostępnienie dokumentu do edycji istniejącemu użytkownikowi
4. Jednoczesną edycję przez wiele zalogowanych osób współdzielonego dokumentu w przeglądarce internetowej, czyli:
   a) odczyt zawsze aktualnej treści dokumentu po jego otwarciu
   b) manualny zapis treści dokumentu w dowolnie wybranej chwili
   c) regularny autozapis zmian wprowadzonych w dokumencie
   d) autoodświeżanie zmian dokonanych przez innych użytkowników aktualnie edytujących dokument
   e) edycję przy zachowanie spójności tekstu
5. Udostępnienie dokumentu osobie niezarejestrowanej w serwisie na określony czas do odczytu
6. Wizaulny podział na strony
7. Podgląd wydruku treści

Celem pracy jest również założenie, że aplikacja będzie miała formę notatnika bez środków typograficznych.

Przed ustaleniem ostatecznej formy celów i założeń była ich podwojona ilość.

W kolejnych rozdziałach opiszę napotkane przeszkody w pisaniu aplikacji, sposoby jakie stosowałam do ich rozwiązania. Omówię również problematykę zagadnień, których nie zaimplementowałam w aplikacji.

Google Inc jest przodową firmą w branży internetowej. Ma ogromne doświadczenie ze składowaniem, wyszukiwaniem i edycją informacji w sieci. Nad swoim własnym pakietem dokumentów w postaci aplikacji webowej pracowała latami. Dzięki stworzeniu bezkonkurencyjnej usługi Google Wave zyskała dodatkowy bagaż doświadczeń i poszerzyła horyzonty - własne, ale i zafascynowała innych. Porzuciła jednak ten projekt na rzecz innych swoich produktów, m. in. Google Drive, gdzie zarządzanie zawartością przypomina Google Wave, jednak w sposób bardziej uporządkowany i schematyczny. Pomysły Google często są rewolucyjne, inspirujące i zaskakują użytkowników internetu. Dzięki temu firma symuluje rozwój w sieci i powstawanie innowacyjnych aplikacji webowych, coraz częściej nazywanych aplikacjami w chmurze.

W pracy tej nie chodzi o sworzenie konkurencji, ale o eksperyment, próbę odwzorowania działających dokumentów Google Docs, jako że stąd wzięła się motywacja do napisania podobnego tworu.

Jeśli tworzoa aplikacja po odpowiednich testach będzie działać zadowalająco na podstawowym poziomie i spełniać założenia pracy, planuję rozwijać projekt w przyszłości. Wiąże się z tym przede wszystkim walka z optymalizacją działania programu.

Dołączona do pracy aplikacja webowa zawiera xxx' xxx linii kodu, z czego co najmniej xx'xxx linii kodu została stworzona samodzielnie.

1.3 . Grupa docelowa odbiorców

Potencjalnym użytkownikiem tworzonej aplikacji może być każda osoba, o ile posiada

dostęp do internetu. Aktualnie aplikacja jest adresowana do osób zainteresowanych tematyką edycji dokumentu tekstowego przez wiele osób jednocześnie oraz do chętnych przetestowania sprawności działania.

W przyszłości jednak grono odbiorców może zostać zawężone, np. w przypadku decyzji o rozwijaniu serwisu świadczącego usługi dla developerów konkretnych języków, gdzie zastosowane zostanie podświatlanie składni itp. Jest to jedna z możliwości, a ten przykład oznacza, że zrezygnuje się np. z problematyki i implementacji formatowania tekstu w przyszłości całkowicie. W istocie wszystko zależy od decyzji podjętych w przyszłości, gdy sprzeczne kierunki i dalsze cele zostaną rozstrzygnięte. O ile perspektywy rozwoju zostaną omówione w tej pracy, to kierunek w którym podąży projekt nie ma większego znaczenia.

Faktycznym odbiorcą jest tu jednak osoba mająca potrzebę tworzenia i składowania wszelkiego rodzaju tekstów z możliwością edycji tego tekstu przez inne osoby oraz gdzie formatowanie tekstu nie jest konieczne.

1.4 . Stan opisywanego tematu w internecie

W sieci istnieje wiele współdzielonych edytorow czasu rzeczywistego. Są to nie tylko edytory tekstu, ale również grafiki, w tym grafiki 3D. Nie wszystkie są określane jako "działające w czasie rzeczywistym" sprawiedliwie, bowiem znaczna ich część działa z wyczuwalnym dla użytkownika opóźnieniem.

Pierwszy taki edytor został zademonstrowany przez Douglasa Engelbarta w 1968 r. Zainteresowanie tematem wzrosło jednak zupełnie niedawno – podczas epoki web 2.0.

Zalety takich edytorow w sieci to głównie:

- Globalna baza danych – za tym idzie globalna dostępność, aktualne dane można mieć zawsze przy sobie
- Oszczędność czasu – dokumenty są przechowywane w jednym miejscu; nie muszą być duplikowane i rozdzielane pomiędzy różne osoby; utrzymują stale aktualną wersję treści; dostępne dla każdego uprawnionego o każdej porze dnia i nocy.
- Nie trzeba instalować żadnego dodatkowego oprogramowania – wystarczy przeglądarka internetowa – bazowe narzędzie przy korzystaniu z sieci internetowej
- Jednoczesna edycja tekstu przez wielu użytkowników w tym samym czasie – pozwala oszczędzić czas oraz lepiej się zgrać w trakcie tworzenia tekstu; łatwiej widoczny postęp pracy nad dokumentem.

Wady niestety również istnieją i oto kilka z nich:

- Konieczny jest dostęp do internetu – potrzebny głównie do odczytu danych, choć możliwe jest cache'owanie na dysku urządzenia ostatniej wersji dokumentu; niektóre edytory umożliwiają korzystanie z programu w trybie off-line, wysyłając do zapisu dane wkrótce po połączeniu się z internetem.
- Przypadek awarii sieci internetowej - choć nie powinien mieć miejsca i dzisiejsze technologie dają możliwość zabezpieczenia na wypadek odłączenia części dostępnych serwerów lub utraty danych - wszystko jest możliwe, a najgorszy scenariusz grzebie

w głodzie i ciemności ludzkość z powodu braku zasilania elektrycznego.

1.5 . Materiały

Informacji na temat istniejących edytorów czasu rzeczywistego oraz sposobu ich działania, w tym cennego kodu źródłowego nie sposób znaleźć w papierowej książce. Moim źródłem informacji jest internet: artykuły, oficjalne blogi organizacji lub firm wydających szukane przeze mnie produkty, nade wszystko jednak kod udostępniony do empirycznego wykorzystania na konkretnej licencji. To dzięki implementacji i analizie tego kodu mogłam bliżej przyjrzeć się procesom zachodzącym podczas wymiany i scalania danych wykorzystującym różne dostępne technonolgie i zrozumieć je, by opracować własną strategię implementacji do osiągnięcia celu.

2 . Pojęcie i zastosowania chmury obliczeniowej

Chmura, dla przypomnienia (ang. cloud computing) to idea możliwośi wykonywania pracy na swoich dokumentach w każdym miejscu i na każdym komputerze bez instalowania żadnych programów. W technologii tej oprogramowanie jest instalowane tylko na jednym serwerze. Wystarczy przeglądarka internetowa. Chmura daje nam możliwość łatwego dzielenia się owocami pracy, publikowania ich, a nawet udostępniania ich do edycji innym osobom. Dzięki temu można np. zacząć pisać dokument na jedmym urządzeniu, a dokończyć na innym.

Zwykli użytkownicy łączą się z internetem i pracując w sieci tworząc chmurę. W chmurze chodzi o użytkowanie określonej usługi, użytkownik nie musi wykupywać licencji za oprogramowanie itp.

[dopisać więcej]

3 . Struktura serwisu

3.1 . Układ strony

[tu znajdzie się screen z aplikacji oraz jego opis]

3.2 . Poszczególne działy serwisu

Serwis nie jest rozbudowany, gdyż jego istotą jest działający edytor tekstu. Posiada kilka zakładek menu. Mapa serwisu jest prosta:

- Strona powitalna
- Strona informacyjna
- Strona o autorze
- Dokumenty
    - lista dokumentów
    - możliwość utworzenia nowego dokumentu
    - możliwość usunięcia dokumentu
    - możiwość przejścia do edycji dokumentu
        - okno edycji dokumentu
- Konto użytkownika
    - możliwość edycji konta
    - wylogowanie

3.3 . Struktura katalogów i plików

[tu znajdzie się screen lub tekstowe wylistowanie drzewa plików z omówieniem celów poszczególnych katalogów]

4 . Środki techniczne zastosowane do wykonania serwisu

4.1 . Przechowywanie danych

4.1.1 Wybór bazy danych

W projekcie skorzystano ze wsparcia systemu zarządzania bazą danych o nazwie MongoDB. Wybór nie był prosty, jednak szybko stało się jasne, że korzystanie z jakiegokolwiek systemu relacyjnego SQL planując rozwój aplikacji w przyszłości mija się z celem z powodu wiadomej niewydajności możliwych zapytań, operacji JOIN oraz głównie zbyt kosztownego czasu połączenia z bazą. Konieczne było wdrożenie systemu podołającemu przypuszczalnej ilości zapytań do bazy w tym samym czasie.

Możliwą alternatywą dla systemów SQL są systemy NoSQL, zaprojektowane z myślą o wydajności zapytań przy ogromnych ilościach danych, realizujące z powodzeniem politykę rozproszonego przechowywania ich na wielu serwerach. Rozwiązania NoSQL powinny sprawdzić się w aplikacji działającej w czasie rzeczywistym. Wybór padł na MongoDB, ponieważ posiadam już pewne doświadczenie z tą bazą i zdecydowałam się ją wdrożyć w projekt, ciekawa efektu i nowych doświadczeń.

W trakcie rozwijania aplikacji zrozumiałam, że wybór nie był idealny, głównie z powodu braku symulowania relacji między zagnieżdżonymi dokumentami. Tworzenie referencji (mających działać analogicznie do relacji w bazach typu SQL) ogólnie mija się z głównym celem korzystania z MongoDB, jednak jest to możliwe i stosowane. Referencje jednak były mi potrzebne przy wyświetlaniu tekstu dokumentu przechowywanego w bazie jako lista dwukierunkowa, mająca z natury swojego poprzednika i następnika – i tu owe referencje do obiektów mogłyby występować. Na szczęście poradziłam sobie w nienajgorszy sposób z tym problemem tworząc samodzielnie tablicę obiektów przekazywaną do widoku. Nie żałuję jednak wyboru, ponieważ nie znałam lepszej alternatywy, a implementacja tej bazy dała wiele do myślenia.

4.1.2 Struktura bazy danych

Jako że MongoDB należy do rodziny nierelacyjnych systemów bazodanowych NoSQL, określony schemat bazy nie istnieje. Dane przechowywane są w postaci kolekcji dokumentów – obiektów w formacie tekstowym wymiany danych JSON. Sam dokument ma postać "klucz" => "wartość". Możemy dowolnie zagnieżdżać kolekcje dokumentów wewnątrz innych.

Zapis do bazy danych i ich pobieranie odbywa się przy pomocy składni języka JavaScript. Dzięki temu zapytania wykonywane przez serwer MongoDB, a co za tym dostęp do danych w formacie JSON, jest szybkie i łatwe.

W omawianej aplikacji schemat bazy jest wyznaczany przez model (klasę) obiektu. Zależy również od tego, które właściwości obiektu posiadają ustaloną wartość w momencie zapisu danych.

4.2 . Podstawowe narzędzia i technologie

4.2.1 Websocket

Narzędzie Webcocket jest standardem W3C (World Wide Web Consortium) i wchodzi w skład technologi HTML5. Zostało opracowane w celu wykorzystywania go do komunikacji między przeglądarką internetową a serwerem webowym. Umożliwia ono komunikację dwukierunkową, gdzie możliwe jest nadawanie i odbieranie informacji w obu kierunkach. Komunikacja jest możliwa za pośrednictwem jednego gniazda TCP (Transmission Control Protocol) . Technologia websocet umożliwia otwarcie gniazda i trzymania otwartego polaczenia z serwerem.

Specyfikacja WebSocket definiuje dwa nowe URI (Uniform Resource Identifier), ws: oraz wss:, dla nieszyfrowanych i szyfrowanych połączeń. URI - jest standardem internetowym umożliwiającym łatwą identyfikację zasobów w sieci. URI jest zazwyczaj łańcuchem znaków, zapisanym zgodnie ze składnią określoną w standardzie.

Zalety korzystania z tej technologii to m.in. trzymanie otwartego polaczenienia do komunikacji. Klient może nasłuchiwać wiadomości od serwera, a kiedy wiadomomość przychodzi wykonuje sie callback. Jeśli zamknę dokument, stronę lub sie wyloguję to połączenie zostaje zamknięte. Nie potrzeba otwierać nowego połaczenia co kilka sekund i wysyłać zapytania do bazy danych, co tak drogo kosztuje, tylko po to, by sprawdzać czy istnieją jakieś nowe wiadmości dla jednego otwartego dokumentu.

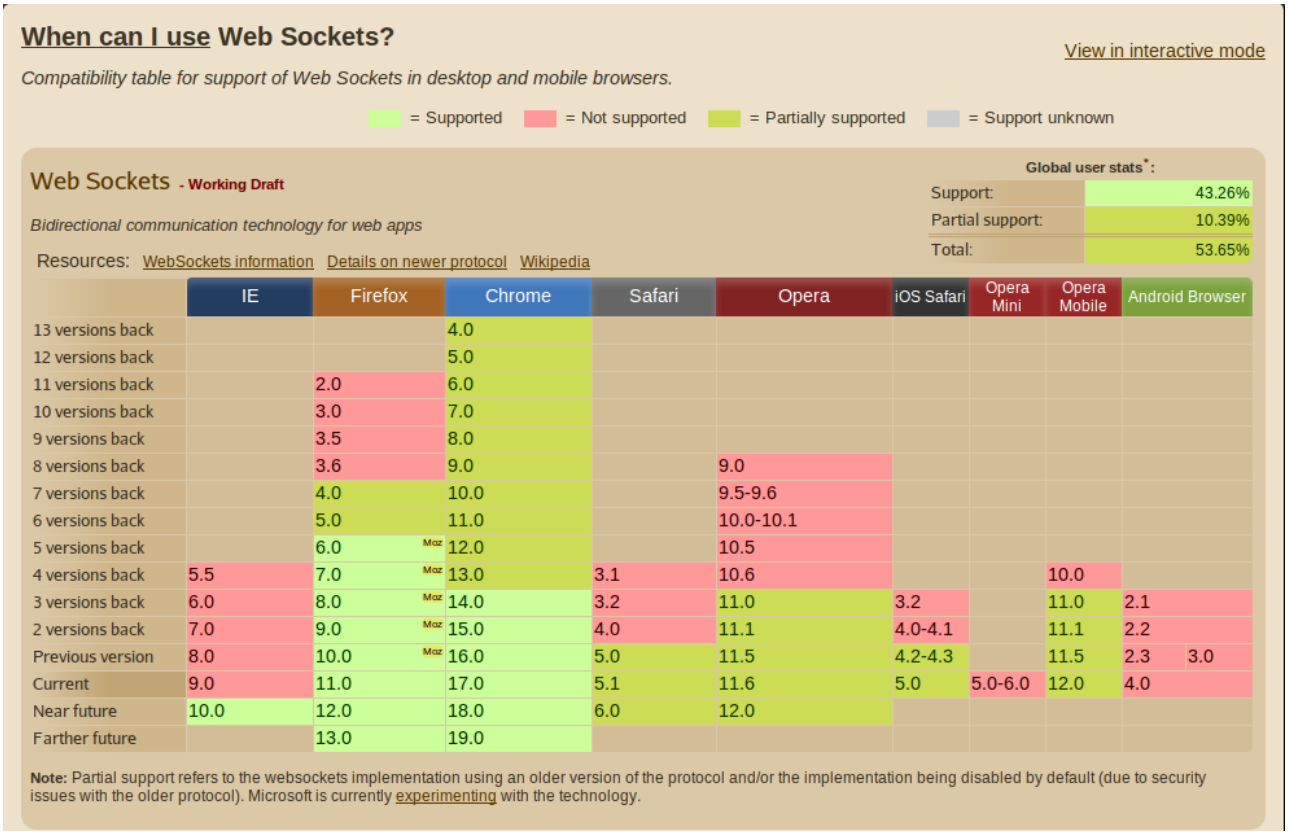
Wykorzystanie tej technologii wiąże się jednak z pewnymi ograniczeniami. Przede wszystkim starsze przeglądarki oraz te nie mające wsparcia dla websocket nie obsłużą go poprawnie. Kolejną wadą jest zatroszczenie się o działanie odzielnego serwera aplikacji, który w moim przypadku podpina się w Ruby Rack - mikroframework, ktory działa jeszcze przed uruchomieniem się frameworka Rails i potrafi przechwytywac urle, po czym kieruje ruch do aplikacji Rails lub do innej. Poza tym websoket jest protokołem nowym. Z tego wynia, że nie wiadomo, ile w nim jeszcze jest luk bezpieczenstwa. Dla porównania, protokol HTTP jest uzywany przez miliardy stron, jest rozwiązaniem sprawdzonym.

Chciałam pokazać w jaki sposób można stworzyć edytor przy wykorzystaniu technologii HTML5. Do obsługi websocket wykorzystuję biblioteke socky. Socky została napisana w JavaScript oraz w ruby:

- biblioteka JavaScript - po stronie klienta jako klasa, która upraszcza korzystanie z websocket gotowymi metodami do wykorzystania wedle uznania
- server ruby – w oparciu o serwer thin (gem) implementujący socky i obsluguje protokół websocket i akutoryzację użytkownika

Czy bedzie działać na urządzeniach mobilnych zależy już tylko od przegladarki.
Android – chrome (web-kit)
Można sprawdzić klasy przegladarek:

A - ma wszystko (js),
B - nie ma ajaxa, opea mini
C - podstawowe



When can I use Web Sockets? View in interactive mode

Compatibility table for support of Web Sockets in desktop and mobile browsers.

= Supported = Not supported = Partially supported = Support unknown

Web Sockets - Working Draft

Bidirectional communication technology for web apps

Resources: WebSockets information Details on newer protocol Wikipedia

Global user stats*:

Support:	43.26%
Partial support:	10.39%
Total:	53.65%

	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Opera Mobile	Android Browser	
13 versions back			4.0							
12 versions back			5.0							
11 versions back		2.0	6.0							
10 versions back		3.0	7.0							
9 versions back		3.5	8.0							
8 versions back		3.6	9.0		9.0					
7 versions back		4.0	10.0		9.5-9.6					
6 versions back		5.0	11.0		10.0-10.1					
5 versions back		6.0 Moz	12.0		10.5					
4 versions back	5.5	7.0 Moz	13.0	3.1	10.6			10.0		
3 versions back	6.0	8.0 Moz	14.0	3.2	11.0	3.2		11.0	2.1	
2 versions back	7.0	9.0 Moz	15.0	4.0	11.1	4.0-4.1		11.1	2.2	
Previous version	8.0	10.0 Moz	16.0	5.0	11.5	4.2-4.3		11.5	2.3	3.0
Current	9.0	11.0	17.0	5.1	11.6	5.0	5.0-6.0	12.0	4.0	
Near future	10.0	12.0	18.0	6.0	12.0					
Farther future		13.0	19.0							

Note: Partial support refers to the websockets implementation using an older version of the protocol and/or the implementation being disabled by default (due to security issues with the older protocol). Microsoft is currently experimenting with the technology.

[Opis obrazka]

4.2.2 Javascript

JavaScript należy do głównych narzędzi w projekcie, ponieważ odpowiada za sklejanie oraz poprawne odświeżanie przychodzącego tekstu dokumentu u każdego z użytkowników z osobna. To właśnie po stronie klienta z omawianej aplikacji wykonuje się algorytm wykrywania zmian oraz algorytm transformacji operacyjnej. Dzięki skryptom w tym języku można skorzystać z cennych rozwiązań websocket.

JavaScript jest niezbędną technologią w przypadku jakichkolwiek manipulacji strukturą wyświetlanego tekstu.

[tu jeszcze dopiszę]

4.2.3 Ruby on Rails

Podstawa aplikacji napisana jest w znanym frameworku webowym ruby - Ruby on Rails. Jest korzeniem, bez którego reszta kodu jest bezużyteczna. Ruby on Rails zapewnia efektywne i szybkie rozwijanie kodu. Minimalizuje niezbędną konfigurację do minimum zalecanymi i sprawdzonymi rozwiązaniami.

Zapewnia zarżadzanie modelami obiektów mapowanych na bazę danych przy pomocy wybranego przeze mnie rozszerzenia o nazwie Mongoid. Mongoid ma rozbudowaną i dokładną dokumentację, dzięki czemu wygodnie się korzystało z tego narzędzia.

Pisząc w tym frameworku korzystałam również z języka znaczników HAML, dzięki któremu struktura HTML dokumentu jest przejrzysta i prosta.

[tu jeszcze dopiszę]

4.2.4 Algorytmy

a) Algorytm google-diff-match-patch - ten algorytm posłużył mi do wykrycia zmian - co dokładnie sie zmieniło oraz w którym miejscu. Po przekazaniu mu dwóch kawałków tekstów otrzymujemy tablicę zmian dotyczących do dotyczących. Poznamy wtedy pozycję usunięcia znaku (bądź ciągu znaków), dodania lub jego zamiany na inny.

Jest to algorytm open source Google'a i wykrywa różnice na tekście nieformatowanym.

b) Algorytm Transformacji Operacyjnej (Operational Transformation) – wspiera szereg funkcjonalności w zaawansowanych systemach współpracy. Został opracowany do rozwiązywania konfliktów oraz utrzymania spójności tekstu w ramach edycji współdzielonej, początkowo tylko dla tekstu nieformatowanego. Na tym algorytmie opierają się rozwiązania Google, Etherpad. Jego pionierami są C. Ellis oraz S. Gibbs.

W mojej aplikacji w postaci biblioteki JavaScript jinfinote. Implementuje ona bufor na łańcuchach znaków i operacje na nich. Ma również kolejke transofmacji - jeśli przyjdą 2 transformacje, to wykonuje je w odpowiednio. Istnieją dwie operacje: insertion i deletion. które biorą OT do nowej gamy aplikacji internetowych.

[tu jeszcze dopiszę]

4.3 . Narzędzia pomocnicze

[tu jeszcze dopiszę]

5 . Spełnione założenia

- Forma dokumentu tekstowego bez formatowania tekstu .
- Edycja dokumentu - komunikacja przez websocket z zapisem do bazy.
- Websocket
- Algorytm OT po stronie klienta
- Udostępnianie dokumentów
- Podgląd dokumentu
- Działanie na wielu paragrafach

[i inne]

6 . Instalacja

Należy się upewnić, że środowisko RVM, Ruby, Rails oraz MongoDB jest zainstalowane.

Uruchomienie serwera Ruby on Rails:
W głównym katalogu aplikacji:

```
$ rails server
```

Uruchomienie serwera Socky:
W katalogu socky-server:

```
$ thin -R config.ru -p3001 start
```

[uzupełnić o konkretne wersje zainstalowanych paczek]

7 . Możliwość rozbudowy

[tu jeszcze dopiszę]

8 . Współdzielona edycja tekstu w czasie

rzeczywistym

8.1 . Problematyka tematu

8.1.1 WYSIWYG – What You See Is What You Get

Konieczna jest obsługa wstawiania tagów html z pomocą JavaScript . W edytorach problem ten jest rozwiązywany w na jeden z dwóch możliwych sposobów:

- Tinymce i iframe - za textarea wstawia iframe z kodem (dokumentem) z atrybutem na elemencie body contenteditable="true". Zapisywane do bazy danych zostaje źródło tego dokumentu. Jest to stary sposób.
- Dla elementu, najczęściej body lub div, ustawia się attrybut contenteditable z wartością ”true”

[opisać dokładniej]

8.1.2 Porcjowanie informacji w bazie danych

[nadmienione trochę w "wybór bazy danych", dopisać]

8.1.3 Porcjowanie struktury na stronie dokumentu

Wyświetlanie:

- Lista dwukierunkowa
- Backbone - modele (te same co backend), trzymają dane cały czas po odświeżeniu strony, można je pobrac live.
- Nowe linie dopiero po enter – każda taka linia to 1 obiekt.

Formatowanie tekstu:

- Mobwrite – ma iframe, span z kazdym ciagiem znakow, które maja inny styl, również kazde slowo w spana. Co 10 s robi query i wysyła dane do zapisu. Przewidywanie cziconek i znakow (przede wszystkim znakow wielojezycznych) (utf-8).
- Stare google docs mialy iframe'a. (screen) i html 4. nowy google doc jest od okolo 1,4 roku.
- Etherpad - ma 2 iframe'y, jeden wewnatrz drugiego. Kazda linia to nowy div, reszta w spanach. Zapisuje autora dodanego tekstu (w przypadku usuniecia nie zapisuje). Komunikacja przez websocket chyba

8.1.4 Odświeżanie niezależnie od wysyłanych informacji przez użytkownika

Klient może prosic o odswiezenie co x czasu.

Technologia websocet HTML5 umozliwia otwarcie gniazda i trzymania otwartego polaczenia z serwerem (innym wydajniejszym niż apache) .

1) opencoweb: Transformacje po stronie klienta (js), 2 podstawowe funkcje JavaScriptowe – jedna wysyła zmiany, druga odbiera. Ta, która odbiera odpowiada za transformacje i sklejanie konfliktow. I funkcja sprawdzajaca czy są wysłane zmiany od kogoś innego, serwer nie zwraca od razu odpowiedzi tylko czeka dopuki nie będzie info o nowych zmianach, by je zwrocic (on ma cały czas status loading i czeka np. max 60 s) – pętla sleep, nie blokuje tylko czeka. Tego się nie ad zaimplementowac w apacheu, bo zajał by 1 watek, a a pache ma ograniczona ilosc watkow do ok 100, więc zablokowaloby to caly serwer, dlatego stosuje się np. nodejs lub tornado pythona, która respektuje wieksza ilosc polaczen jednoczesnych. Div z editable, niceedit

2) Zamiast oczekujacego – window set interwal i wysylanie funkcji jsowej co jakiś czas w celu pobrania zmian. Ajax co 10 s. Mozna wtedy na apacheu.

Nodejs ma swoj serwer, bardzo szybki, jest to srodowisko - wiele biliotek, silnik js chrome'a. Server ten nasluchuje na eventy, obsługuje naraz wiele polaczen (kilka tydziecy, dla porownania apache np. 100) jednoczesnych. Etherpad korzysta z niego w swoim edytorze. I przekazywane są informacje w tablocy json.

8.1.5 Konflikty i konwergencja

Konflikty i konwergencja (schodzenie się, zbieganie się tekstu, merge)
czyli edycja przez użytkowników tego samego tekstu w tym samym czasie.

Co w sytuacji, gdy na serwerze jest wersja nowsza niż wersja danego użytkownika i w dodatku użytkownik ten chce wysłac swoje zmiany w tym samym miejscu, co wysłał inny uzytkownik przed chwila?

W systemach kontroli wersji, aby wysłać swoje zmiany w takim przypadku użytkownik musi mieć pobraną najnowszą wersję dokumentu. Następnie mając najnowszą wersję pliku użytkownik może wysłać swoje zmiany. Załóżmy, że mamy konflikt. Użytkownik

musi go rozwiązać sam, decyzyjnie.

W przypadku współdzielenia i jednoczesnej edycji tekstu w czasie rzeczywistym nie możemy stosować rozwiązania z systemów kontroli wersji, wymagać pobrania aktualnej wesji przed wysłaniem swich zmian, ponieważ jest to edycja live i jest nie do wyobrazenia, za nagle wystąpi jakiś konflikt.

Pomocnym tutaj może być Agorytm OT.

Skleja intelligentnie zmiany lub zastępuje, jeśli nie da się skleić. Zastępuje, jeśli zmieniono to samo, i przyjmuje za ostateczną wersję wersję tej osoby, która jest ważniejsza. Kryteria ważniejszości mogą być różne. Lub odbywa się głosowanie którą wersję przyjąć.

operational transformation (OT) algorithm

Użytkownicy mogą być informowani o swoich wzajemnych zmianach, z poślizgiem czasowym minimum czas transmisji danych.

Transformacja operacyjna, przykład działania:

u1: Insert[0, "x"] (wstawienie znaku "x" na pozycji 0)
u2: Delete[2, "c"] (usunięcie znaku "c" na pozycji 2)

u1: insert[0, "x"] (wstawienie znaku "x" na pozycji 0)
u2: delete[2, "c"] (usunięcie znaku "c" na pozycji 2)
1. => "xabc"

u1: insert[0, "x"] (wstawienie znaku "x" na pozycji 0)
u2: delete[2, "c"] (usunięcie znaku "c" na pozycji 2)
1. => "xabc"
2. delete[3, "c"]

u1: insert[0, "x"] (wstawienie znaku "x" na pozycji 0)
u2: delete[2, "c"] (usunięcie znaku "c" na pozycji 2)
1. => "xabc"
2. delete[3, "c"]
3. => "xab"

8.1.6 Historia wersji

Zawsze 1 aktualny dokument na serwerze, archiwizowane wersje.

8.1.7 Podział na strony

Nie będzie pamietana ta informacja, będzie liczona w locie, za kazda zmiana (jeśli usune linie powyzej, to tekst przeskoczy o linie w gore, może zniknac strona) – podglad wydruku. Innym sposobem byloby to bardzo uciazliwe (takie jest moje przemyslenie, nie szukałam na to odpowiedzi specjalnie, na prace nie muszę tego osiagnac) .

8.1.8 Formatowanie a spójność tekstu

Trudne ze względu na to, że algorytmu wykrywania zmian muszą być udoskonalone. Problem pojawia się również, kiedy różni użytkownicy korzystają z różnych przeglądarek. Interakcję w każdej przeglądarce trzeba inaczej zaplanować i przewidzieć.

8.1.9 Przechwytywanie kursorów tekstu formatowanego

Koncepcja google – przechwytuje kazda litere i wstawie w odpowiednie miejsce, kluczowa rola: kursor w witrualnym divie (gdzie kursor, to div), na poziomie wyzszym niż html (na warstwie wyżej html).
http://jquerymobile.com/gbs/

8.2 . Sposób działania istniejących edytorów

Przykłady edytorów tekstu bazujące na przeglądarkach:

- Etherpad - open source, wiele edytorów opiera się na jego kodzie, został m. in. przejęty do integracji z Google Wave; edytor ten był pierwszą webową aplikacją działającą wydajnie, ale ma spore wymagania sprzętowe; pozwala na wyświetlanie zmian każdego użytkwonika innym kolorem; dostępny chat w dokumencie, dokument określany jest tu słowem "pad"; zapisuje dane w określonych odstępach czasu ale daje możliwość zapisu w dowolnym momencie; dostępna historia zmian dokumentu; możliwość pobrania dokumentu w postaci pliku w wielu formatach; bazuje na algorytmie OT; licencja Apache License; wydany w 2008 r.
- Etherpad Lite – to przepisany Etherpad do JavaScript z wykorzystaniem serwera node.js; wydajność porównywalna z Etherpad; http://beta.etherpad.org;

- Google Docs – znany każdemu wielofunkcyjny darmowy pakiet biurowy do tworzenia dokumentów on-line, 2005 r; jego następcą jest produkt Google Drive, 2012 r;
- Google Wave – produkt Google, nierozwijany od 2010 r., oparty na technologiach HTML5 umożliwiająca współtworzenie w czasie rzeczywistym dokumentów różnego formatu; jego elementy zostały wykorzystane w innych produktach tej samej firmy.
- OpenCoweb – project Dojo Foundation, bazuje na algorytmie OT zaimplementowanym z języku JavaScript; działa z serwerem Tornado (python), łatwo go zainstalować dzięki szczegółowej dokumentacji; napisany w python; http://demos.opencoweb.org .
- Zoho Writer
- Mobwrite, (Google, plain), http://mobwrite3.appspot.com/static/demos/editor.html

Istnieje również cała gama podobnych edytorów, ale desktopowych, nie bazujących na przeglądarce internetowej, oto kilka z nich:

- SubEthaEdit – edytor dla systemów Mac OS X, m. in. z edytowalną funkcją podświetlania składni dla ponad 40 języków programowania; tekst renderowany za pomocą HTML i silnika przeglądarek WebKit; bazuje na algorytmie OT, pierwsze wydanie 2003 r.
- Abiword - wieloplatformowy i darmowy edytor o otwartym źródle (licencja GNU GPL); posiada wsparcie dla wielu formatów plików, m.in. LaTeX, HTML, Microsoft Word, a także ODF.
- ACE - Ajax.org Cloud9 Editor, wieloplatformowy i darmowy edytor kodu; udostępnia swój kod na licencji BSD, w tym algorytmy podświetlania składni wielu języków programowania; napisany w języku JavaScript, bazuje na algorytmie OT; nawet ponad 100 000 linii kodu nie jest dla niego problemem; wydany w 2010 r.
- CoWord – produkt Microsoft, edytor w postaci Microsoft Word z możliwością edycji przez wielu użytkowników jednocześnie; jest częścią CoOffice, bazuje na algorytmie OT;
- GNU Emacs - wparcie dla edytora czasu rzeczywistego można osiągnąć dzięki rozszerzeniom lub wbudowanym komendom; licencja GNU GPL;
- Gobby – darmowy iw ieloplatformowy projekt, szyfruje przesyłane informacje, posiada wbudowany chat czasu rzeczywistego, bazuje na algorytmie OT, 2005 r;
- MoonEdit – wieloplatformowy, open source, prosty edytor tekstu czasu rzeczywistego; daje możliwość podglądu kursorów innych użytkowników aktualnie edytujących dokument oraz cofania wprowadzonych zmian.
- CoMaya (Autodesk Maya, 3D)

[szczegółowiej opisać]

8.3 . Zastosowane rozwiązania

Przy założeniu, że cała treść dokumentu podzielona jest na linie, można skorzystać z

właściwości listy dwukierunkowej . Nowa linia jest nowym obiektem w bazie danych, a także otoczona nowym tagiem html. Koniec linii wyznaczany jest przez klawisz enter. Każda linia ma swojego poprzednika i następnika - `id` innej linii. W ten sposób w momencie usunięcia bądź dodania nowej linii edytujemy maksymalnie 3 rekordy, zamist przenumerowywać wszystkie linie w dokumencie.

Każda taka linia ma własną instancję klasy jinfinote i jest kontrolowana przez transformację operacyjną. Jest również 1 porcją zapisywaną do bazy i ciągiem znaków przeszukiwanych w celu wykrycia zmian. Oczekiwanie na wysłanie zmian występuje tylko na linii aktualnie edytowanej, czyli na tej, gdzie aktywny jest kursor tekstu.

Websocket

[te opisane już wyżej i inne]

9 . Podsumowanie

[uzupełnić]

10 . Źródła informacji

10.1 . Bibliografia

1. http://www.youtube.com/watch?v=3ykZYKCK7AM&feature=player_embedded#! , Google engineer David Wang, „Google Wave: Live collaborative editing”
2. http://en.wikipedia.org/wiki/Operational_transformation
3. https://github.com/Pita/etherpad-lite
4. http://documentcloud.github.com/backbone/
5. http://mobwrite3.appspot.com/static/demos/editor.html
6. https://github.com/benjamn/kix-standalone
7. http://opencoweb.org/ocwdocs/intro/openg.html
8. http://dev.w3.org/html5/websockets/ - specyfikacja WebSocket
9. http://http://mongoid.org/ - mapper Ruby (MongoDB)
10. https://github.com/sveith/jinfinote - biblioteki JavaScript (OT)
11. https://github.com/socky - bilioteki Socky (serwer, js)
12. http://code.google.com/p/google-diff-match-patch/ - algorytm Google Wave
13. http://twitter.github.com/bootstrap/ - css bootstrap
14. http://haml-lang.com/ - haml
15. http://caniuse.com/websocket - dostępność WebSocket

16. http://www.youtube.com/watch?v=3ykZYKCK7AM&feature=player_em
17. Google engineer David Wang, „Google Wave: Live
18. collaborative editing”
19. http://en.wikipedia.org/wiki/Operational_transformation
20. https://github.com/Pita/etherpad-lite
21. http://documentcloud.github.com/backbone/
22. http://mobwrite3.appspot.com/static/demos/editor.html
23. https://github.com/benjamn/kix-standalone
24. http://opencoweb.org/ocwdocs/intro/openg.html
25. http://www.sxc.hu
26. http://sprawnymarketing.pl/artykuly/wszystkie-kleski-google/

10.2 . Cytowane i wzmiankowane narzędzia

- HTML5 - język html posłużył do budowy struktury strony
- CSS - zapewniający jednolity wygląd strony na różnych przeglądarkach, czyli kaskadowe arkusze stylów.
- Haml (XHTML Abstraction Markup Language) jest językiem znaczników używanym do prostego i przejrzystego opisywania
- JavaScript -
- Mozilla Firefox -
- Chrome -
- JavaScript -
- WebSocket -
- MongoDB -
- Mongoid Mapper -
- Ruby on Rails -

[uzupełnić]

10.3 . Serwisy internetowe użytych narzędzi

[uzupełnić]

10.4 . Spis ilustracji

[uzupełnić]

11 . Załączniki

[uzupełnić]