DOKUMENTACJA WSTĘPNA

System współbieżnej edycji pliku tekstowego

Jan Prugarewicz
Jakub Świerczyński
Michał Makoś
Bartłomiej Wolski

1.Opis problemu

System służący do współpracy online nad edycją dokumentu tekstowego. Każdy użytkownik, mający uprawnienia do edycji dokumentu, powinien móc dowolnie manipulować jego treścią (usuwać i dodawać elementy). System zapewnia uwierzytelnienie tj. rejestrację i logowanie. Aplikacje będą posiadać proste interfejsy graficzne, a nawigować będzie się przy edycji będzie można za pomocą myszki lub strzałek.

2. Założenia

- Edycja dokumentu polega na dodawaniu i usuwaniu znaków.
- Dwóch lub więcej użytkowników może edytować ten sam fragment tekstu.
- Dostęp do dokumentu jest przyznawany poprzez uzyskanie unikalnego kodu id dokumentu lub poprzez nadanie uprawniania użytkownikowi przez twórce dokumentu.
- Nad jednym dokumentem nie może pracować więcej niż N użytkowników.
- Jest wyznaczony maksymalny rozmiar jaki może osiągnąć dokument.
- Można edytować tylko jeden dokument na raz.

3.Funkcjonalności

- Zapewniony jest bezpieczny kanał przesyłania informacji.
- Odporność na zakłócenia/przerwania w ruchu sieciowym.
- System zapewnia serwer, aplikacje klienckie i aplikację administracyjną.
- System powinien raportować błędy użytkownikowi w razie wystąpienia takiego.
- Automatyczny zapis dokumentu na serwerze.
- System powinien w sposób wydajny korzystać z zasobów sprzętowych, zarówno serwera jak i użytkownika.
- Aplikacje powinny działać bez wyraźnych opóźnień, tak aby wszystkie zmiany dokonane w dokumencie były od razu widoczne w interfejsach użytkowników.
- System zapewnia bezpieczne kończenie sesji.
- Możliwość zapisywania dokumentu na dysku lokalnym przez aplikacje kliencką.
- Moduł administracyjny ma dostęp tylko do danych o użytkownikach i metadanych.

4. Przypadki użycia

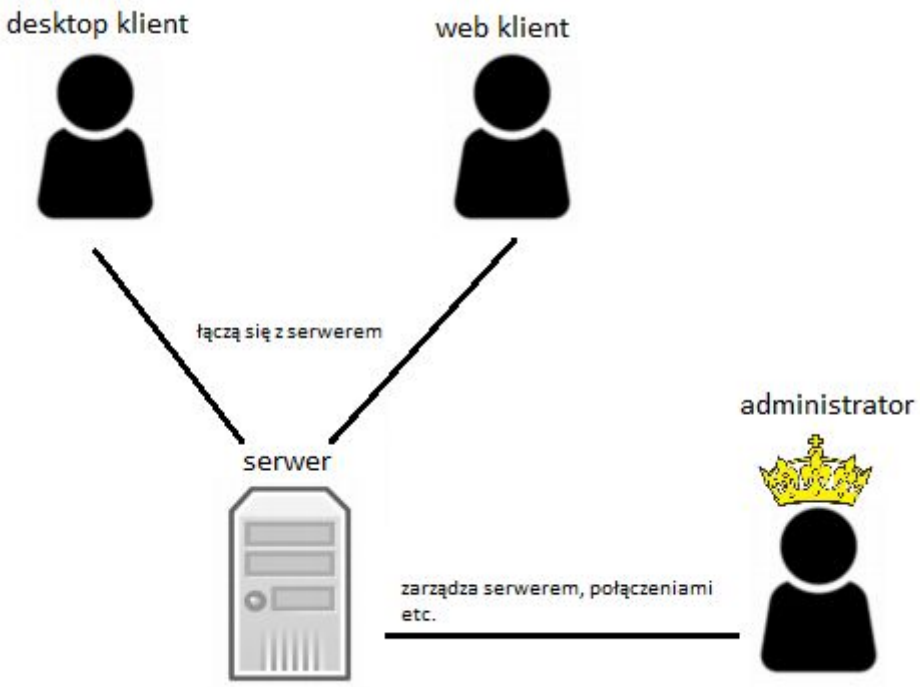
- logowanie
- rejestracja
- utworzenie dokumentu
- import dokumentu z dysku lokalnego
- przyznanie praw do edycji dokumentu
- edycja dokumentu
- zapisanie dokumentu na dysku lokalnym.

5.Technologie i platforma docelowa

Planowane technologie do wykorzystania w projekcie to C++, Java, Python oraz framework Spring. Ewentualne biblioteki i inne frameworki będą uzgodnione podczas implementacji poszczególnych modułów. Platformą docelową rozwiązania jest Linux.

6.Architektura rozwiązania

System będzie się składać z czterech głównych komponentów tj. aplikacji serwera, klienta desktopowego, klienta webowego (uruchamianego w przeglądarce) oraz aplikacji administrującej, ściśle współpracującej z serwerem.



Serwer, w celu komfortowego i optymalnego użytkowania, będzie prowadził komunikację z klientami w sposób asynchroniczny. Podstawą całości komunikacji będzie wykorzystanie gniazd i właściwości protokołu TCP. Asynchroniczność ma na celu dostarczenie nieblokującej usługi tzn. jeden użytkownik wykonujący operację edycji nie wywłaszcza całej funkcjonalności systemu na swoją wyłączność w trakcie trwania operacji. Problem asynchroniczności zostanie rozwiązany przez wsparcie bibliotek, które dostarczają odpowiednie funkcje umożliwiające osiągnięcie prawidłowej komunikacji, przykładowo Boost.Asio do języka C++.

Serwer powinien także przechowywać pliki tekstowe użytkowników (z punktu widzenia użytkownika - w chmurze), tak żeby ten po połączeniu się z usługą był w stanie wybrać dowolny ze swoich plików do edycji lub stworzyć nowy.
Dodatkowo, w celach weryfikacyjnych, w pamięci serwera powinny być przechowywane loginy oraz skróty haseł użytkowników korzystających z usługi kolaboracji.

Aplikacje klienckie (desktopowa i webowa) przy łączeniu z usługą wyświetlą odpowiedni monit logowania, które zapewni weryfikację danego użytkownika. Klienci wyposażeni będą również w możliwość komunikacji asynchronicznej w celu utrzymania wydajności całego systemu. Dany użytkownik będzie miał opcję wyboru wylistowanych plików w celu rozpoczęciu edycji. Klient będzie miał możliwość dołączyć się do edycji pliku innego użytkownika posiadając odpowiedni kod, a także przyznać uprawnienia do edycji innym. Po skończonej edycji możliwe będzie pobranie pliku na lokalną maszynę.

Aplikacja administrująca odpowiada za kontrolowanie serwera. Posiada prawo do manipulowania prawami użytkowników i nadzorowania nad wszystkimi aktywnymi użytkownikami, więc ma możliwość rozłączenia od usługi danego klienta. Dodatkowo aplikacja administrująca posłuży za generowanie inicjalnych danych nowych użytkowników (przydział tych danych na wzór wydziałowego serwera studia). Odpowiednie moduły w aplikacji administrującej będą odpowiedzialne za powyższe zadania. Aplikacja ta będzie ściśle współpracowała z serwerem.

7.Protokoły
Podstawowym protokołem komunikacji gniazd, nawiązywania i utrzymywania połączeń jest TCP. Do bezpiecznej komunikacji wykorzystany będzie protokół TLS(SSL).

Komunikaty będą przesyłane prawdopodobnie własnym protokołem pakietu np. w XML/JSON lub przy pomocy protokołu HTTP. Możliwe, że przy wysyłaniu większego pliku wykorzystany zostanie protokół FTP.

8.Sytuacje krytyczne i propozycje ich rozwiązania

- Rozłączenie użytkownika z serwerem - w przypadku rozłączenia klient automatycznie próbuje odzyskać połączenie. Serwer czeka przez określony czas po czym bezpiecznie zamyka sesję. Po stronie klienta wypisywany jest monit błędu “utracono połączenie z serwerem”.
- Opóźnienia związane z przesyłaniem danych i sytuacje asynchroniczne - algorytm synchronizacji zakłada asynchroniczne modyfikacje, zaś pakiety które dotrą ze znacząco dużym opóźnieniem będą musiały być wycofane.
- Właściciel dokumentu usuwa dokument w momencie gdy inni użytkownicy nad nim pracują - plik zostaje bezpowrotnie usunięty z pamięci serwera, po stronie klienta użytkowników jest wyświetlony monit “plik który próbujesz edytować został usunięty przez właściciela”, aplikacja wraca do listy dokumentów pomniejszonej o usunięty dokument.

- Właściciel dokumentu odbiera prawa do edycji dokumentu użytkownikowi - po stronie klienta jest wyświetlony monit “straciłeś uprawnienia do edycji tego dokumentu”, aplikacja wraca do listy dokumentów pomniejszonej o dokument, do którego uprawnienia zostały utracone.
- Wykrywanie utraconego połączenia - implementacja klasy opakowującej dane wysyłane w postaci wiadomości: prefix (wielkość bufora) + bufor danych w celu sprawdzenia czy połączenie istnieje zostanie wysłana wiadomość z nullem w prefixie. Takie odpytywanie nie wygeneruje dużego ruchu sieciowego ze względu na null a jest w stanie wykryć pół-otwarte połączenie ze względu na przesył danych.

9. Algorytm synchronizacji

Każdemu znakowi przyporządkowany jest wektor par. Para ta przechowuje liczbę naturalną, która odpowiada za pozycję znaku w tekście (kolejność znaków) oraz id użytkownika. Kiedy użytkownik chce dodać ciąg znaków w dowolnym miejscu w dokumencie znakom z tego ciągu przypisujemy wektor ze znaku, który znajdował się przed wstawianym ciągiem. Następnie do wektorów znaków z wstawianego ciągu dodajemy po jednej parze, gdzie jako pozycję przyjmujemy pozycję znaku w nowym ciągu a jako id użytkownika przyjmujemy id użytkownika wstawiającego dany ciąg znaków.

Takie rozwiązanie zapewnia prawidłową synchronizację operacji, gdyż:
- unikalne, posortowane id znaku zapewnia, że zawsze wstawimy nasz ciąg znaków we właściwym miejscu, nawet jak coś usunie lub doda przed miejscem, w którym chcemy coś wstawić
- id użytkownika przypisane do znaku zapewnia rozróżnialność znaków w przypadku gdy kilku użytkowników chce wstawić znak w to samo miejsce w tej samej chwili - inaczej wygenerowaliby to samo id.

Przykład:

F [(0,0)] t=0	**F [(0,0)] t=0**
o [(1,0)] t=1	**o [(1,0)] t=1**
r [(2,0)] t=2	**r [(2,0)] t=2**
m [(3,0)] t=3	**m [(3,0)] t=3**
u [(4,0)] t=4	**a [(3,0), (0,1)] t=6**
j [(5,0)] t=5	**t [(3,0), (1,1)] t=7**
	u [(4,0)] t=4
	j [(5,0)] t=5

Na początku użytkownik 0 napisał “formuj”. Następnie użytkownik 1 dopisał “at”, więc znakom ‘a’ i ‘t’ został przypisany wektor znaku ‘m’ oraz dodany kolejny element z id użytkownika 1 oraz pozycją w dodanym ciągu.

10.Testowanie

W etapie pierwszym do testowania połączeń gniazd zaplanowane zostało użycie programu netcat lub wireshark.

Testy jednostkowe, strukturalne i aplikacyjne zostaną zaprojektowane w późniejszym etapie pracy nad projektem.

11. Wstępny podział prac w zespole

Klient desktopowy (Java) - Michał Makoś

Klient webowy (Java Spring) - Jan Prugarewicz

Serwer (C++ z użyciem biblioteki Boost.Asio) - Jakub Świerczyński

Moduł/Aplikacja administrujący/a (Python) - Bartłomiej Wolski