Implementacija uređivača teksta s mogućnošću suradnje u stvarnom vremenu

Knežić, Kristijan

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Diplomski sveučilišni studij računarstva

Diplomski rad

IMPLEMENTACIJA UREĐIVAČA TEKSTA S MOGUĆNOŠĆU SURADNJE U STVARNOM VREMENU

Kristijan Knežić

0069078003

Rijeka, rujan 2022.

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Diplomski sveučilišni studij računarstva

Diplomski rad

IMPLEMENTACIJA UREĐIVAČA TEKSTA S MOGUĆNOŠĆU SURADNJE U STVARNOM VREMENU

Mentor: doc. dr. sc. Sandi Ljubić

Kristijan Knežić

0069078003

Rijeka, rujan 2022.

UMJESTO OVOG PAPIRA UBACITI ORIGINALNI PAPIR S ZADATKOM

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA DIPLOMSKE ISPITE

Rijeka, 3. ožujka 2022.

Zavod: **Zavod za računarstvo**
Predmet: **Razvoj mobilnih aplikacija**
Grana: **2.09.01 arhitektura računalnih sustava**

ZADATAK ZA DIPLOMSKI RAD

Pristupnik: **Kristijan Knežić (0069078003)**
Studij: Diplomski sveučilišni studij računarstva
Modul: Računalni sustavi

Zadatak: **Implementacija uređivača teksta s mogućnošću suradnje u stvarnom vremenu / Implementation of a Real-time Collaboration Text Editor**

Opis zadatka:

U sklopu diplomskog rada potrebno je istražiti postojeće tehnologije za održavanje konzistentnosti i kontrolu višekorisničkog paralelnog rada u naprednim programskim sustavima. Primijeniti stečena saznanja pri implementaciji vlastite inačice uređivača teksta s mogućnošću suradnje u kontekstu zajedničkog uređivanja dokumenata. Procijeniti mogućnosti nadogradnje rješenja jednostavnim sustavom za upravljanje inačicama dokumenata. Provjeriti ispravnost implementiranog rješenja korištenjem istoga u stolnoj i mobilnoj domeni.

Rad mora biti napisan prema Uputama za pisanje diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

Zadatak uručen pristupniku: 21. ožujka 2022.

Mentor:

Doc. dr. sc. Sandi Ljubić

Predsjednik povjerenstva za diplomski ispit:

Prof. dr. sc. Kristijan Lenac

Izjava o samostalnoj izradi rada

Izjavljujem da sam samostalno izradio ovaj rad.

Rijeka, rujan 2022.

_____

Kristijan Knežić

ZAHVALA

Veliko hvala mom mentoru doc. dr. sc. Sandiju Ljubiću za pomoć, savjete i podršku tokom pisanja ovog rada.

Posebnu zahvalu želim uputiti svojim roditeljima i obitelji na potpori kojom su uvelike doprinijeli mom uspjehu.

Za kraj, ovaj rad posvećujem svojoj djevojci Luciji kao zahvalu za nesebičnu pomoć, strpljenje i podršku tijekom studiranja.

SADRŽAJ

1. UVOD

Posljednjih nekoliko godina, zbog pandemijskih razloga, ali i novih trendova, način na koji osobe surađuju značajno se promijenio. Suradnja među ljudima vrlo je važan aspekt svake timske aktivnosti i stoga je važna za bilo koju organizaciju, posao, znanost, obrazovanje, politički ili društveni institut. Prilagođavanjem na trenutnu situaciju, često se javlja potreba za suradnjom između fizički udaljenih osoba. Kako bi suradnja između udaljenih osoba što više nalikovala suradnji uživo, postavlja se zahtjev za suradnjom u stvarnom vremenu. Kao što naziv govori, suradnja u stvarnom vremenu omogućuje sudionicima da zadatke izvršavaju zajedno u stvarnom vremenu, neovisno o lokaciji gdje se nalaze.

Jedan od oblika suradnje u stvarnom vremenu je uređivanja teksta. Proces pisanja tekstova, akademskih ili druge namjene, oduvijek je bio dio društvene prirode, budući da se naše misli nadovezuju na misli drugih koji su došli prije nas. Tradicionalni proces suradnje uključivao je isključivo linearno uređivanje teksta. Uređivanje teksta vršilo se u različito vrijeme, pri čemu su ga suradnici između sebe slali naprijed-nazad. Članovi grupe uređivali bi kopije dokumenata u izolaciji te ih zatim međusobno usklađivali kako bi stvorili zajednički dokument. Takav proces primjer je asinkrone suradnje. Sinkronom suradnjom ili suradnjom u stvarnom vremenu preinake članova odmah se prikazuju ostalim članovima grupe [1]. Suradnja u stvarnom vremenu može biti od velike pomoći kod uređivanja teksta, gdje suradnici ne moraju trošiti vrijeme na usklađivanje svojih kopija već simultano mogu uređivati tekst imajući pritom u vidu namjere i promjene napravljene od drugih suradnika.

Ovaj rad nastojat će u sljedećim poglavljima predstaviti problematiku uređivanja teksta sa suradnjom u stvarnom vremenu, objasniti način na koji funkcioniraju dvije reprezentativne tehnologije suradnje te pokazati kako iste rješavaju predstavljenu problematiku istovremenog uređivanja. U sklopu rada razvijena je ogledna aplikacija koja sadrži uređivač teksta s mogućnošću suradnje u stvarnom vremenu. Opisat će se razvoj i alati korišteni za razvoj uređivača teksta te će se, popraćena zaslonskim snimkama, ispitati ispravnost rješenja simulacijom istovremenog uređivanja nekih specifičnih slučajeva. Također, uređivač teksta nadograđen je jednostavnim sustavom za upravljanje inačicama čiji će razvoj kao i mogućnosti biti opisani u radu.

2. KONKURENTNO UREĐIVANJE TEKSTA U STVARNOM VREMENU

„Uređivanje teksta s mogućnošću suradnje u stvarnom vremenu omogućuje većem broju korisnika da istovremeno uređuju isti dokument.“ Tvrdnja je ovo autora Attiya [2] koja pobliže opisuje uređivače teksta s mogućnošću suradnje u stvarnom vremenu. Ovakvi sustavi rade na način da stvaraju kopije dokumenta na uređaju korisnika (inačice), nad kojima korisnici rade promjene koje se propagiraju na ostalim inačicama dokumenata. Propagacija promjena može se vršiti putem središnjeg poslužitelja (engl. *centralized server*) ili ravnopravnih čvorova (engl. *peer to peer*). To dovodi do osnovne značajke uređivača teksta sa suradnjom u stvarnom vremenu: sve napravljene promjene se proslijede na sve inačice dokumenta te se na dosljedan način uklope u sam dokument. Time se garantira dosljednost između svih inačica dokumenta te sinkronizacija promjena svih korisnika u stvarnom vremenu.

Svrha uređivanja dokumenta u stvarnom vremenu do izražaja dolazi ako se problematizira nesinkrono zajedničko uređivanje, gdje promjene ne propagiraju u stvarnom vremenu već po pozivu korisnika. Takvim načinom rada postoji mogućnost da više korisnika istovremeno uređuje istu riječ ili redak dokumenta, a da toga nisu svjesni. Prilikom sinkronizacije promjena, rezultat različitog uređivanja istog dijela dokumenta dovest će do konflikta uređivanja (engl. *merge conflict*). Konflikt kao takav zahtjeva ručno spajanje promjena, zahtijevajući od korisnika prihvaćanje svojih ili promjena drugih korisnika. Više o konfliktima opisano je u jednom od sljedećih poglavlja.

2.1. Formatiranje teksta

Kada se govori o uređivaču teksta važno je spomenuti da sam tekst može imati podatke o samom sebi, a to su takozvani meta podaci (engl. *metadata*). Meta podaci služe se za opisivanje svojstava samog teksta, kao što su primjerice boja, stil pisma, podebljanje, kurziv i slično te definiraju kako će tekst izgledati u uređivaču. Posjedovanje takvih podataka o tekstu, tekst dijeli na čisti tekst (engl. *plain text*) i formatirani tekst (engl. *rich text*). Shodno tome, uređivači teksta mogu se podijeliti prema tipu teksta kojeg obrađuju: uređivači čistog teksta te uređivači bogatog teksta.

Prvi uređivači čistog teksta nastali su kao uređivači programskog koda, razvijeni s prioritetom na djelotvornost i štednju računalnih resursa ispred upotrebljivosti i korisničkog

iskustva [3]. Neki od primjera uređivača čistog teksta su *vi* (razvijen za *Unix* operacijski sustav), *Simpletext* (razvijen od *Applea* za *Mac* operacijski sustav) te *Notepad* (razvijen za *Windows* operacijski sustav). Zapis čistog teksta sastoji se od liste znakova (engl. *characters*) te kontrolnih znakova (engl. *control characters*) kao što je novi redak ili uvlačenje odlomka. Osim što je jednostavnije manipuliranje čistim tekstom, može se koristiti i u bogatim tekstualnim uređivačima. Svaki bogati uređivač može prepoznati i koristiti datoteku čistog teksta. U usporedbi s čistim, uređivači bogatog teksta danas su mnogo češće korišteni zbog potrebe za formatiranjem teksta. Primjeri takvih uređivača su *Microsoft Word*, *Google Docs* te *TinyMCE*.

2.2. Konflikti kod uređivanja teksta

Prilikom korištenja uređivača teksta s mogućnošću suradnje (u ovom slučaju nije nužno da se suradnja odvija u stvarnom vremenu) korisnici na svojim inačicama dokumenta stvaraju promjene koje se u nekom trenutku spajaju. Prilikom uređivanja teksta moguće je da više korisnika radi promjene na istom dijelu sadržaja, bila to određena riječ ili redak u tekstu. Kod spajanja takvih promjena može doći do konflikta. Konflikt se događa kada dva ili više korisnika izmjenjuju isti objekt s različitim namjerama [4]. Primjerice, ako je izvorni tekst nekog dokumenta bio „Ana jede jabuku.“, korisnik A odlučio je u svojoj inačici dokumenta napraviti promjenu u „Ana jede krušku.“, dok je korisnik B odlučio napraviti promjenu u „Ana jede kruh.“. Spajanje takvih promjena rezultirat će konfliktom. Konflikti se mogu kategorizirati u dvije vrste, isključivi (engl. *exclusive*) i neisključivi (engl. *non-exclusive*).

Kod isključivih konflikata promjene se ne mogu istovremeno primijeniti, odnosno ako se promjene primjenjuju na dokument, primijenit će se serijski (na dokument se prvo primjenjuje promjena A pa tek onda promjena B). To znači da će prethodna promjena biti prepisana (engl. *overwritten*) onom kasnijom. Primjer isključivog konflikta je ako korisnik A nad nekom riječi radi promjenu veličine slova u 12 pt, dok korisnik B nad istom riječi radi promjenu veličine slova u 14 pt. Kako se obje promjene ne mogu primijeniti zajedno, jedna promjena će prepisati drugu. Za razliku od isključivih, promjene kod neisključivih konflikata mogu se primijeniti istovremeno na izvorni dokument te ne prepisuju jedna drugu. Primjerice, ako je izvorna riječ „mrak“, korisnik A u svom dokumentu riječi dodaje slova „su“ i radi promjenu u „sumrak“, dok korisnik B u svom dokumentu slovo „k“ zamjenjuje sa slovom „v“ i radi promjenu u „mrav“. Spajanjem ovakvih promjena nastaje konflikt jer su oba korisnika uređivala istu riječ s drugačijom namjerom, međutim, obje promjene mogu se primijeniti bez

da jedna prepiše drugu. Primjena obje promjene rezultirati će riječju „sumrav“. Takva riječ može se označiti kao rezultat konflikta te kasnije promijeniti u nešto smisleno.

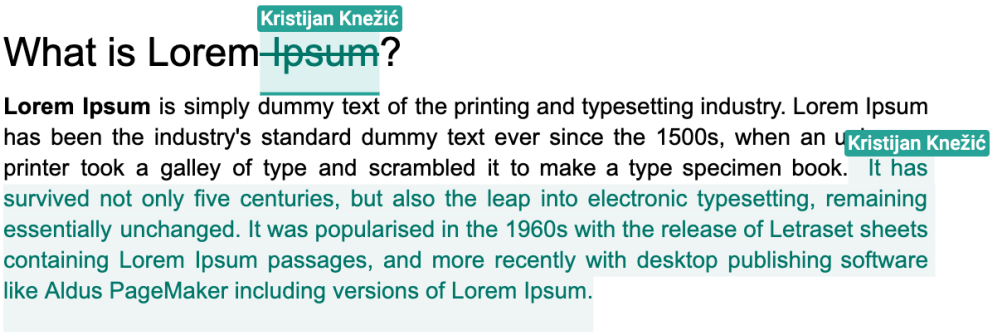
Postoji nekoliko pristupa kojim se konflikti mogu izbjeći ili, ako se dogode, riješiti. Neki od njih su: pristup zaključavanjem, pristup s više inačica (engl. *multi-versioning*) i pristup korištenjem sustava suradnje u stvarnom vremenu. Pristup zaključavanjem ne rješava konflikte već se svodi na prevenciju konflikata. Pristup zaključavanjem dopušta da u određenom vremenu samo jedan korisnik uređuje tekst ili dio teksta. Za to vrijeme drugi korisnici su u zaključanom stanju te čekaju da prvi korisnik završi s uređivanjem. Potom u svojoj inačici teksta dohvaćaju njegove promjene. Po završetku uređivanja prvog korisnika, sljedeći dobiva pristup uređivanju. Pristup korištenja više inačica omogućuje višekorisnički rad u isto vrijeme, a koristi se za očuvanje sukobljenih promjena koje su dovele do konflikta. Iz sukobljenih promjena stvaraju se različite inačice dokumenata. Ovim pristupom omogućena je manipulacija isključivim i neisključivim konfliktima jer korisnici mogu vidjeti namjere ostalih korisnika te na temelju njih uskladiti uređivanje dokumenta. Pristup korištenjem sustava suradnje u stvarnom vremenu uspješno manipulira neisključivim konfliktima, ali nije prilagođen za isključive konflikte. Promjene nastale u isto vrijeme, koje dovode do isključivih konflikata, serijalizira te ih serijski primjenjuje. To dovodi do toga da se početne promjene prepišu zadnjom primijenjenom [4]. Više o načinu kako funkcioniraju sustavi suradnje u stvarnom vremenu biti će rečeno u Poglavlju 3.

2.3. Upravljanje inačicama

Tijekom uređivanja dokumenta vrlo često dolazi do potrebe za dohvaćanjem sadržaja koji je nekad bio sadržan u dokumentu ili jednostavno vraćanje dokumenta u neko prošlo stanje. Za ovakve potrebe koristi se sustav upravljanja inačicama. „Važna značajka programske podrške je mogućnost pohranjivanja više verzija istog dokumenta. Ovo pokazuje evoluciju dokumenta i osobito je korisno kod dokumenata koji se izmjenjuju i ponovno objavljuju“ [5]. Sustav upravljanja inačicama također služi kao dodatna razina sigurnosti od gubljenja podataka, ukoliko je iz nekog razloga tekst nekog dokumenta obrisan, korištenjem inačica moguće je tekst vratiti.

Inačice dokumenta mogu se stvarati na dva načina, ručno i automatski. Ručnim stvaranjem inačica korisnici, ovisno o potrebi, sami određuju kada žele stvoriti novu inačicu dokumenta. Napredniji način stvaranja inačica je automatski, kod kojega se nove inačice stvaraju nakon određenog vremena (to može biti period od nekoliko sekundi do nekoliko sati)

ili automatski po korisnikovu završetku uređivanja teksta. Način stvaranja inačica po završetku korisnikova uređivanja koristi *Googleov* uređivač teksta *Google Docs*. *Google Docs* dodatno u svaku inačicu pohranjuje i korisnika koji je svojim promjenama inicirao stvaranje inačice. Ovakvim stalnim praćenjem promjena i stvaranjem inačica uz informaciju o tvorcu promjena, kod uređivača teksta s mogućnošću suradnje, dobiva se detaljna povijest uređivanja dokumenta. Dobiva se uvid u sve promjene napravljene nad dokumentom kao i autore tih promjena, što je vrlo često korisna značajka kod višekorisničke suradnje. Kod naprednijih sustava upravljanja inačicama često se nudi mogućnost usporedbe sadržaja inačice, najčešće s trenutnim sadržajem dokumenta. Koristeći boje i/ili stil slova za prikaz razlika između inačica na vrlo diskretan i lako shvatljiv način dobiva se uvid u nastale promjene. Primjerice, u uređivaču *Google Docs* dodani tekst označen je sjenčanjem, dok je precrtavanjem označen obrisani tekst (Slika 1.1). Osim samog sadržaja dokumenta, najčešće dodatne informacije koje jedna inačica sadrži su vrijeme stvaranja, autor i naziv inačice.



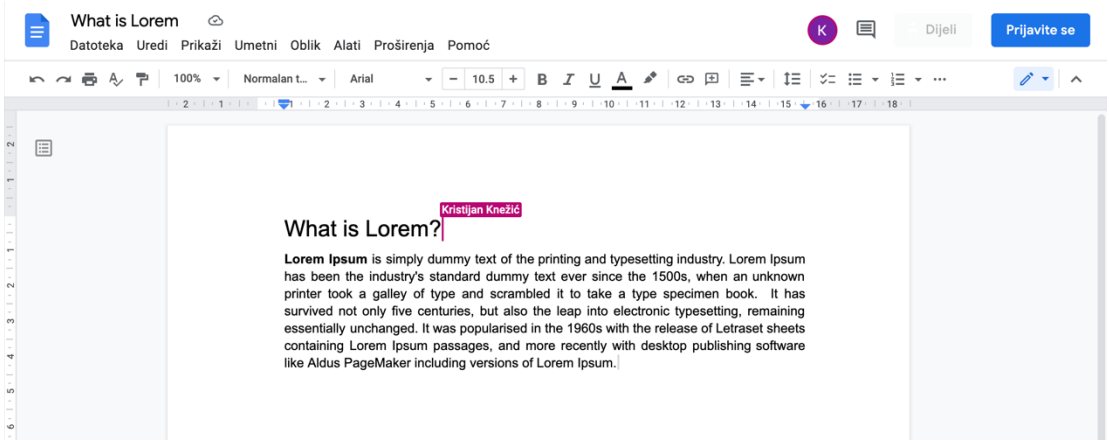
Slika 1.1. *Usporedba inačica u uređivaču Google Docs*

2.4. Primjeri uređivača teksta sa suradnjom u stvarnom vremenu

U ovom poglavlju spomenut će se neki od danas često korištenih uređivača teksta kao i uređivač koji je bio začetnik uređivanja teksta sa suradnjom u stvarnom vremenu. Tehnologija kojom je podržana suradnja u stvarnom vremenu u svakom tekstualnom uređivaču ovdje će biti samo spomenuta, dok će u Poglavlju 3 biti detaljnije opisana.

Jedan od najpoznatijih i najčešće korištenih uređivača teksta sa suradnjom u stvarnom vremenu je *Google Docs* (Slika 1.2.). Besplatni uređivač teksta razvijen na web tehnologijama od strane *Googlea*, nudi se kao dio *Google Workspace* paketa. Dostupan je gotovo na svim

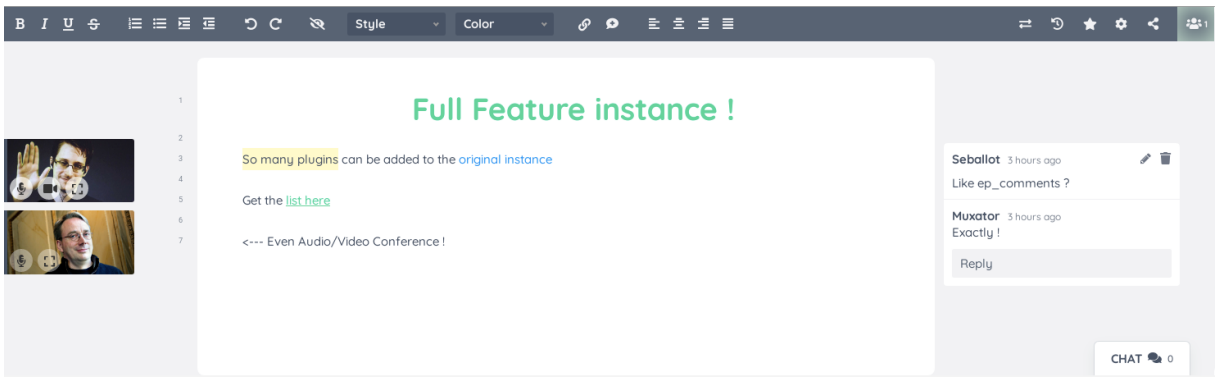
uređajima i platformama, a za pokretanje je potreban samo web preglednik, odnosno kod mobilnih uređaja *Google Docs* aplikacija. Posjeduje podršku za nekoliko različitih tipova tekstualnih datoteka, kao što su .docx, .txt, .rtf i .odt. Kao što je spomenuto u prethodnom poglavlju, osim suradnje u stvarnom vremenu, posjeduje vrlo napredan sustav upravljanja inačicama. *Google Docs* nije programski alat otvorenog koda (engl. *open source*), ali poznato je da suradnju u stvarnom vremenu ostvaruje pomoću tehnologije *operational transformation* (OT) [6].



Slika 1.2. *Grafičko sučelje uređivača teksta Google Docs [7]*

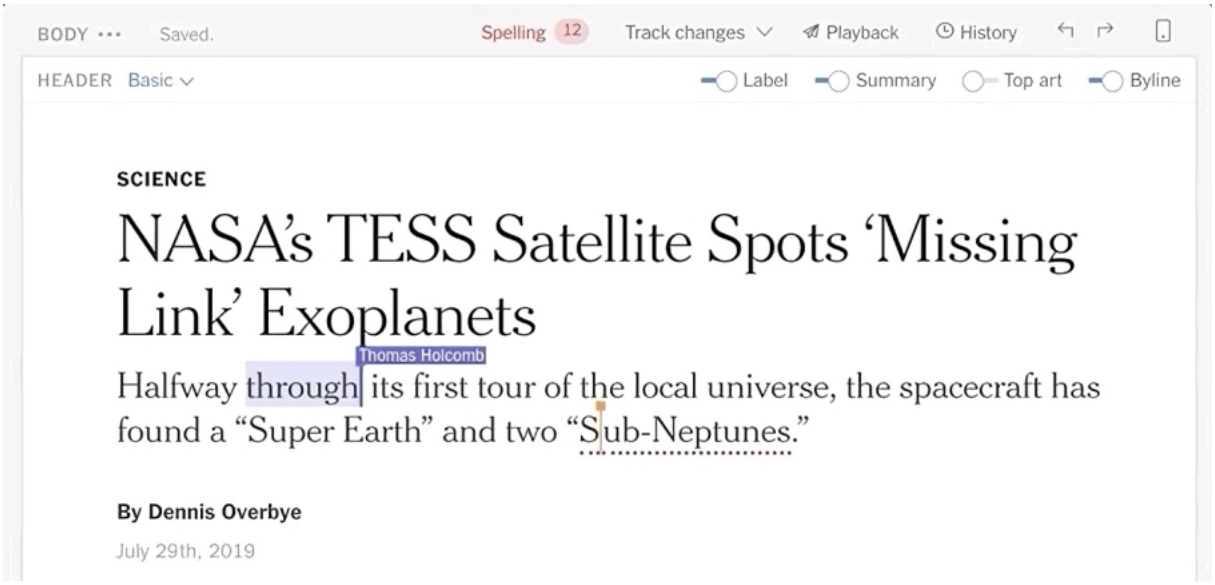
Uz *Google Docs* trenutno najčešće korišteni uređivač teksta sa suradnjom u stvarnom vremenu je *Etherpad* (Slika 1.3.). *Etherpad* je programski alat otvorenog koda pokretan web tehnologijama, a prvi puta se pojavljuje 2008. godine [8]. Zbog pristupa programskom kodu često je korišten u mnogim neprofitabilnim organizacijama, kao što je *Wikimedia*. Kao i *Google Docs, Etherpad* nudi brojne mogućnosti uređivanja teksta, što govori i podatak da postoji oko 290 opcionalnih proširenja na osnovnu funkciju uređivača. Kao programski alat otvorenog koda, *Etherpad* potiče korisnike da donesu svoj doprinos daljnjem razvoju *Etherpada*, upravo zato velik broj proširenja napravljen je od pojedinaca, organizacija i tvrtki koje koriste *Etherpad*. Za ostvarivanje mogućnosti suradnje u stvarnom vremenu, *Etherpad* također koristi tehnologiju *operational transformation*. Zanimljivost vezana za *Google Docs* i *Etherpad* uređivače je da je 2016. godine znanstveni rad pod nazivom „Performance of real-time collaborative editors at large scale: user perspective“ [8] proveo testiranje performansi kod korištenja uređivača s većim brojem suradnika. Rezultati testiranja pokazali su da kod većeg broja korisnika (više od 10) u oba uređivača značajno opadaju performanse. Vrijeme potrebno za sinkronizaciju promjena između korisnika traje i do 15-ak sekundi. Međutim, unazad

nekoliko godina, oba uređivača objavljuju unaprjeđenje sustava suradnje u stvarnom vremenu; *Etherpad* tvrdi da je moguć broj istovremenih korisnika čak tisuću te za taj slučaj nude i kalkulator potrebnih računalnih resursa za posluživanje (engl. *hosting*) *Etherpad* poslužiteljske instance [9].



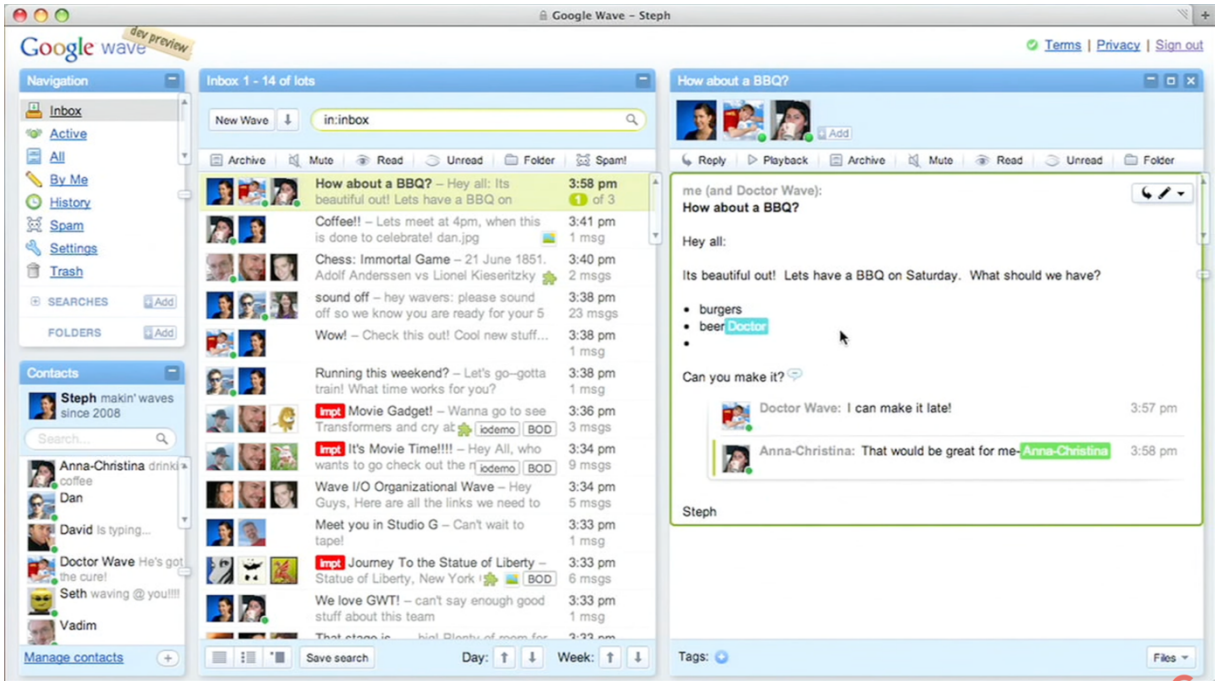
Slika 1.3. *Grafičko sučelje uređivača teksta Etherpad [10]*

Uređivač teksta sa mogućnošću suradnje u stvarnom vremenu razvijen za potrebe jedne organizacije je *Oak*. Razvojni tim *New York Timesa* razvio je *Oak* uređivač za svoje potrebe pisanja članaka (Slika 1.4. prikazuje grafičko sučelje *Oak* uređivača). Iz *New York Timesa* tvrde kako je za objavljivanje članka potrebno nekoliko ljudi te da ponekad oko uređivanja nastane pomutnja [11]. Kako bi tome doskočili i istovremeno osigurali da više autora može uređivati sadržaj članaka, a pritom ne prepisivati sadržaj jedni drugima, odlučili su razviti svoj uređivač. Za razliku od uređivača koji je do tada korišten (samo je jedan korisnik mogao uređivati sadržaj u nekom vremenu, pristup zaključavanjem), novi uređivač razvijen je uz mogućnost istovremene suradnje više korisnika koristeći alat *Prosemirror* [12]. *Prosemirror* je alat otvorenog koda s ciljem pojednostavljenja razvoja uređivača teksta, a za suradnju u stvarnom vremenu koristi sustav temeljen na središnjem tijelu (engl. *central authority*) koje određuje redoslijed primjene promjena korisnika [12].



Slika 1.4. *Grafičko sučelje uređivača teksta Oak [11]*

Posljednji primjer uređivača teksta je razvijen kao dio *Google Wave* platforme. Kasnije poznat i kao *Apache Wave*, bio je nova internetska komunikacijska platforma predstavljena sintagmom „Kako bi izgledao e-mail da je izmišljen danas“. Izvorno ga je razvio *Google* i najavio 28. svibnja 2009. U početku je korišten samo za potrebe programera, a 2010. godine postaje javno dostupan za svu populaciju. Glavni cilj bio je unaprjeđenje komunikacije između korisnika, a to se postizalo komunikacijom u stvarnom vremenu. Komunikacija između korisnika odvijala se unutar „Wave“-a, cjeline predstavljene dokumentom kojeg su korisnici mogli uređivati u stvarnom vremenu. Na Slici 1.5. desni okvir predstavlja dokument koji su korisnici uređivali. Uređivač teksta kojim se uređivao dokument omogućavao je stiliziranje i formatiranje teksta slično današnjim uređivačima. Uređivanje se odvijalo u stvarnom vremenu, a korisnici su mogli vidjeti pokazivače drugih korisnika te njihove promjene u dokumentu. Sustav suradnje u stvarnom vremenu oslanjao se na tehnologiju *operational transformation*. Zbog pada popularnosti i broja korisnika, *Google Wave* gubi programsku podršku te biva ugašen.



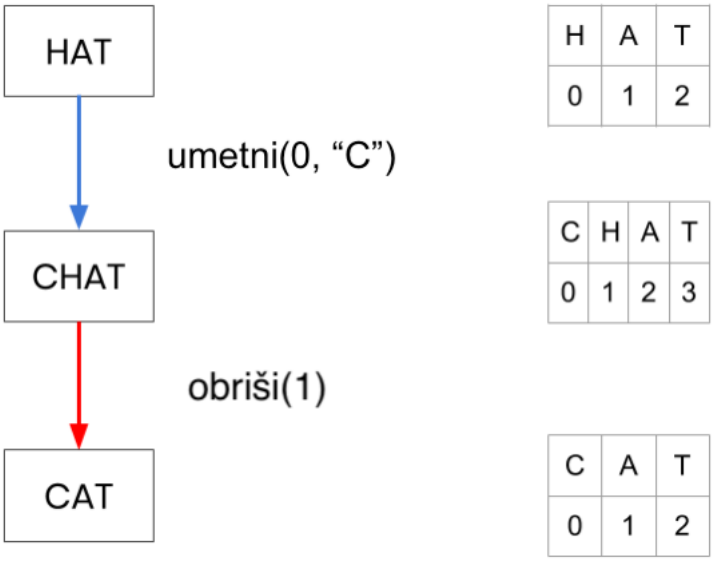
Slika 1.5. *Grafičko sučelje platforme Google Wave*

3. TEHNOLOGIJE SUSTAVA SURADNJE U STVARNOM VREMENU

Kako bi uređivač teksta podržavao suradnju u stvarnom vremenu, nužno je da koristiti određeni sustav suradnje. Danas se sustavi suradnje najčešće temelje na tehnologiji *operational transformation* ili tehnologiji korištenjem *conflict-free replicated data type* (CRDT) strukture podataka. Gotovo svi moderni uređivači teksta s mogućnošću suradnje u stvarnom vremenu usvojili su pristup replicirane arhitekture. To znači da se kod svake strane koja sudjeluje u suradnji replicira tekstualni dokument. Korisnik može izravno uređivati repliku dokumenta lokalno i odmah vidjeti učinak uređivanja. Međutim, lokalna uređivanja, osim što se primjenjuju lokalno u dokumentu, moraju se propagirati i na inačice dokumenta ostalih korisnika. Za propagaciju uređivanja na dokument ostalih korisnika postoje dva pristupa. Prvi pristup je propagacija uređivanja kao operacija (uređivanje se može smatrati skupom operacija), dok je drugi pristup propagacija uređivanja kao stanja (propagira se stanje nastalo uređivanjem). OT i CRDT sustavi suradnje za slanje uređivanja na dokumente ostalih korisnika koriste pristup propagacije operacija.

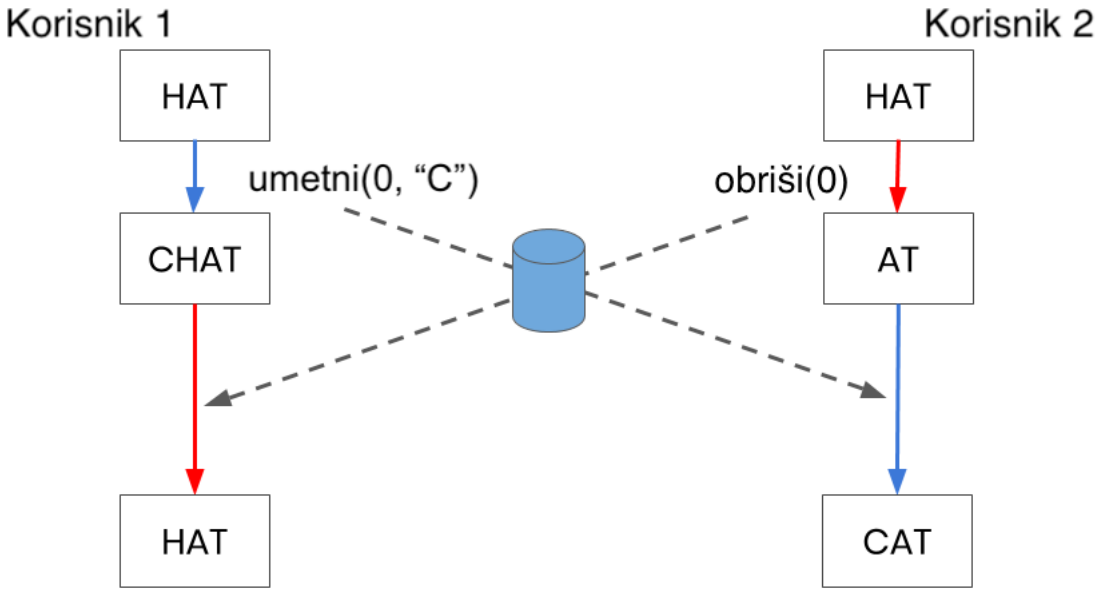
Za razumijevanje tehnologija sustava suradnje važno je pobliže objasniti uređivač teksta i način na koji uređivač teksta funkcionira. Uređivač teksta je mjesto gdje se mogu umetati ili brisati tekstualni znakovi. Svaki znak ima svoju vrijednost i numerički indeks koji određuje njegovu poziciju u dokumentu. Primjer je riječ „HAT“, gdje prvi znak ima vrijednost „H“ s indeksom pozicije 0, drugi znak vrijednost „A“ s pozicijom 1 i treći znak vrijednost „T“ s pozicijom 2. Operacije kojima se uređuje tekst su umetanje ili brisanje znakova. Operacija umetanja vrši se umetanjem vrijednosti znaka na određenu poziciju – npr. umetni(3, „A“), dok se za operaciju brisanja potrebno referencirati na indeks pozicije znaka koji se briše – npr. obriši(1). Ovakve operacije utječu na indeks pozicije ostalih znakova. To znači da ako je na poziciju 0 umetnut znak, indeksi pozicije svih ostalih znakova bit će uvećani za 1.

Ako se na tekstu „HAT“ redom prvo primijeni operacija umetanja znaka „C“ na početnu poziciju - umetni(0, „C“) te potom operacija brisanja znaka H - obriši(1), rezultat će biti tekst „CAT“ (Slika 3.1). Ovakvo uređivanje opisuje uređivanje teksta kada ga uređuje samo jedan korisnik, bez suradnje.



Slika 3.1. *Operacije umetanja i brisanja znakova*

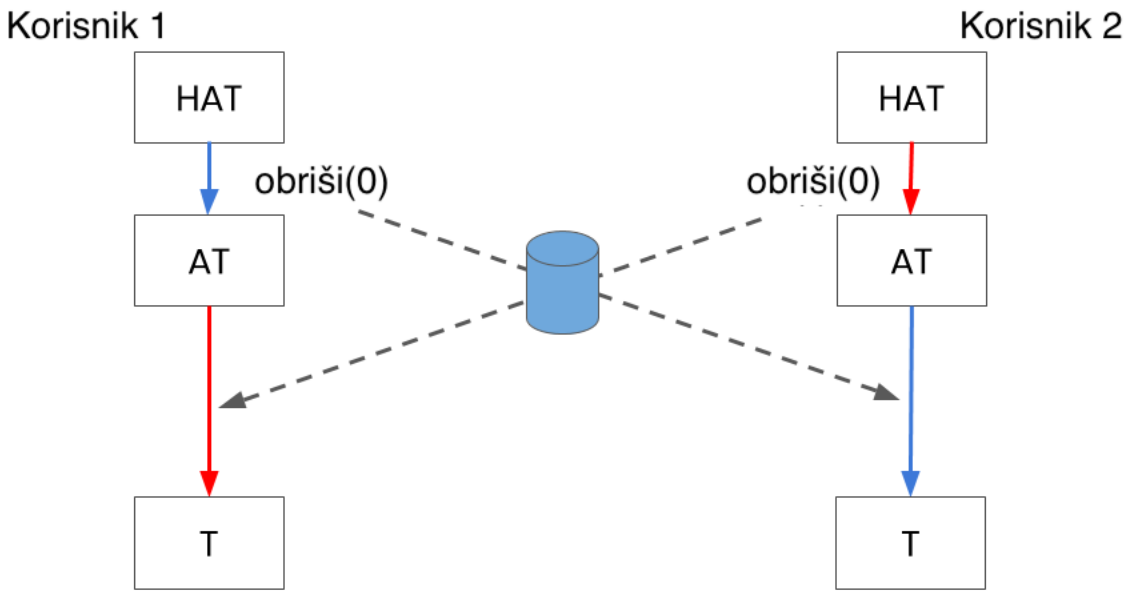
Kod suradnje više korisnika uređivanje postaje složenije. Kako bi problematika suradničkog uređivanja u stvarnom vremenu bila jasnije opisana koristit će se primjer suradnje dva korisnika. Pretpostavlja se da svaki korisnik ima svoju kopiju dokumenta te da su dokumenti korisnika povezani centralnim poslužiteljem. U ovom primjeru centralni poslužitelj funkcionira kao vrlo jednostavan i naivan sustav suradnje koji samo replicira operacije jednog korisnika u dokument drugog. Koristit će se riječ „HAT“ i operacije iz prethodnog primjera, ali sada će svaku od operacija napraviti jedan korisnik u svom dokumentu. U isto vrijeme korisnik 1 dodavat će znak C na početak reda (napraviti operaciju umetni(0, „C“)), dok će drugi brisati znak H (napraviti operaciju obriši(0)). Nakon što su operacije napravili lokalno, tekst u njihovim dokumentima više nije isti. Sustav suradnje sada replicira operacije jednog korisnika u tekst drugog i to dovodi do različitog krajnjeg rezultata (Slika 3.2.). Kako njihovi tekstovi nisu bili isti, umetanje i brisanje se izvršilo na drugoj lokaciji od predviđene. Korisnik 1 u svom dokumentu ima riječ „HAT“, dok korisnik 2 u svom ima riječ „CAT“. Različita stanja dokumenata korisnika pokazuju kako dokumenti nisu konvergirali u isto stanje.



Slika 3.2. *Suradnja s istovremenim operacijama umetanja i brisanja*

Razlog zašto dokumenti korisnika nisu konvergirali u isto stanje je taj što su operacije umetanja i brisanja kod svakog korisnika primijenjene drugačijim redoslijedom. Operacije nisu komutativne i zato krajnji tekst korisnika nije isti. Komutativnost kod operacija bi značila uvijek isti krajnji rezultat, neovisno o redoslijedu izvršavanja operacija.

Drugi problematičan slučaj kod suradnje u stvarnom vremenu je neidempotentnost operacija. Neidempotentne operacije mogu se objasniti sljedećim primjerom. Pretpostavka je da oba korisnika opet imaju tekst „HAT“ te žele dobiti tekst „AT“ brisanjem znaka „H“ s pozicije 0. Oba korisnika u istom vremenu rade operaciju obriši(0) jer je u njihovim dokumentima znak „H“ na poziciji 0. Kada se operacije jednog repliciraju u dokument drugog, to rezultira ukupno dvostrukim brisanjem znaka na poziciji 0. Od teksta kod svakog ostaje samo znak „T“ (Slika 3.3.). Ovog puta operacije jesu komutativne te su njihovi dokumenti konvergirali u isto stanje, ali krajnji tekst nije kakav su korisnici zamislili. To se događa jer operacije brisanja nisu idempotentne. Kada bi operacije bile idempotentne, konstantno izvršavanje takvih operacija rezultiralo bi svaki puta istim rezultatom. Za usporedbu, u matematičkoj domeni, množenje brojem 1 je idempotentna operacija i zato izvršavanjem takve operacije rezultat uvijek ostaje isti.



Slika 3.3. *Suradnja s istovremenim operacijama brisanja*

Ovim primjerima daje se uvid u komutativne i idempotentne operacije, problematiku suradnje u stvarnom vremenu te izazove koje tehnologija sustava suradnje u stvarnom vremenu mora riješiti kako bi suradnja u stvarnom vremenu bila smislena i ispravna. U sljedećim poglavljima rada objasnit će se kako i na koji način tehnologija *operational transformation* suradnje i tehnologija suradnje zasnovana na *conflict-free replicated data type* strukturi podataka pristupaju ovim izazovima.

3.1. *Operational transformation*

Tehnologiju *operational transformation* predstavili su 1989. godine C. A. Ellis i J. Gibbs [13]. Tehnologija potječe iz istraživanja o računalom podržanom suradničkom radu i suradničkom uređivanju. Prvi puta je primijenjena u GROVE-u, uređivaču razvijenom u sklopu istraživanja [14]. Ideja tehnologije *operational transformation* je da suradnici rade na istom skupu podataka dok se između njih razmjenjuju operacije koje oni vrše nad tim skupom podataka. Operacije koje korisnik lokalno generira nad podacima izvršavaju se istog trenutka dok se udaljene operacije (operacije ostalih suradnika) transformiraju prije izvršavanja ne bi li se otklonile nedosljednosti uređivanja [13]. OT tehnologija rješava nedosljednosti operacija te time jamči isti krajnji rezultat u svim inačicama podataka korisnika.

OT tehnologija može se podijeliti u dva sloja, transformacijski kontrolni algoritam (engl. *transformation control algorithm*) i transformacijske funkcije (engl. *transformation*

functions). Transformacijski kontrolni algoritam odlučuje koje je transformacijske funkcije potrebno primijeniti na istovremene operacije. S druge strane, transformacijske funkcije koriste se za transformiranje istovremenih operacija [13]. Kod svake od strana koja sudjeluje u suradnji transformacijski kontrolni algoritam održava privremeni spremnik (engl. *buffer*) u kojem sprema izvršene lokalne operacije koje bi mogle biti istovremene s dolazećim operacijama drugih strana.

Životni ciklus operacija u OT sustavu suradnje je sljedeći:

- Kada korisnik napravi neku operaciju u svom tekstualnom dokumentu, operacija se odmah izvršava u lokalnom dokumentu korisnika. Po izvršavanju operacije, operacija se zapisuje u lokalni privremeni spremnik uz vremensku oznaku (engl. *timestamp*) izvršavanja. Razlog zapisivanja u privremeni spremnik je mogućnost vremenskog preklapanja (istovremenog izvršavanja) s nekom od pristiglih operacija druge strane. Sljedeći korak je propagacija operacije na udaljene druge strane putem internetske veze.
- Po dolasku propagirane operacije na drugu stranu, transformacijski kontrolni algoritam druge strane, na temelju vremenske oznake, uspoređuje pristiglu operaciju s operacijama iz svog lokalnog privremenog spremnika. Ukoliko postoji vremensko preklapanje pristigle operacije s nekom od operacija u privremenom spremniku, pozivaju se transformacijske funkcije. Transformacijske funkcije transformiraju pristiglu operaciju u odnosu na istovremenu operaciju iz lokalnog spremnika i tako stvaraju novu transformiranu operaciju. Nova transformirana operacija se izvršava te se promjene prve strane sada vide i u dokumentu druge strane. Također, nova transformirana operacija po izvršavanju se zapisuje u lokalni privremeni spremnik.

Za uređivač čistog teksta, kod kojega su moguće samo operacije umetanja i brisanja, postoje četiri vrste transformacijskih funkcija. Kombiniranjem dvije vrste operacija (umetanje i brisanje) dobivaju se ukupno četiri kombinacije: umetanje - umetanje, umetanje - brisanje, brisanje – umetanje te brisanje - brisanje. Četiri transformacijske funkcije koje slijedno odgovaraju ovim kombinacijama označavaju se s Tii, Tid, Tdi, Tdd (oznaka *i* dolazi od engleske riječi *insert* za ubacivanje, dok je oznaka *d* od engleske riječi *delete* za brisanje). Svaka od transformacijskih funkcija kao atribute prima dvije operacije, prva je pristigla operacija, a druga istovremena lokalna operacija. Usporedbom pozicijskih atributa operacija, transformacijska

funkcija određuje utjecaj jedne funkcije na drugu te shodno tome stvara novu transformiranu funkciju.

Ako se operacija umetanja označi s Ins(p, c), a operacija brisanja s Del(p) (gdje je *p* pozicija znaka, a *c* vrijednost znaka) algoritmi transformacijskih funkcija mogu se opisati programskim kodom iz Ispisa 3.1.

```
Tii(Ins[p1,c1], Ins[p2, c2]) {
   if p1 < p2  or (p1 = p2 and u1 > u2)
    return Ins[p1, c1];
  else
    return Ins[p1+1, c1];
     }
T id(Ins[p1,c1], Del[p2]) {
    if p1 <= p2
      return Ins[p1, c1];
  else
    return Ins[p1-1, c1];
}
Tdi(Del[p1], Ins[p2, c2]) {
  if p1 < p2
    return Del[p1];
  else
    return Del[p1 + 1];
}
Tdd(Del[p1], Del[p2]) {
  if p1 < p2
    return Del[p1];
  else if  p1 > p2
    return Del[p2];
  else
    return I;
}
```

Ispis 3.1. *Pojednostavljeni programski kod Tii, Tid, Tdi i Tdd transformacijskih funkcija*

Za bolje shvaćanje OT tehnologije koristit će se primjer gdje će biti opisani koraci suradnje u sustavu zasnovanom na OT tehnologiji. Opisani koraci praćeni su Slikom 3.4.

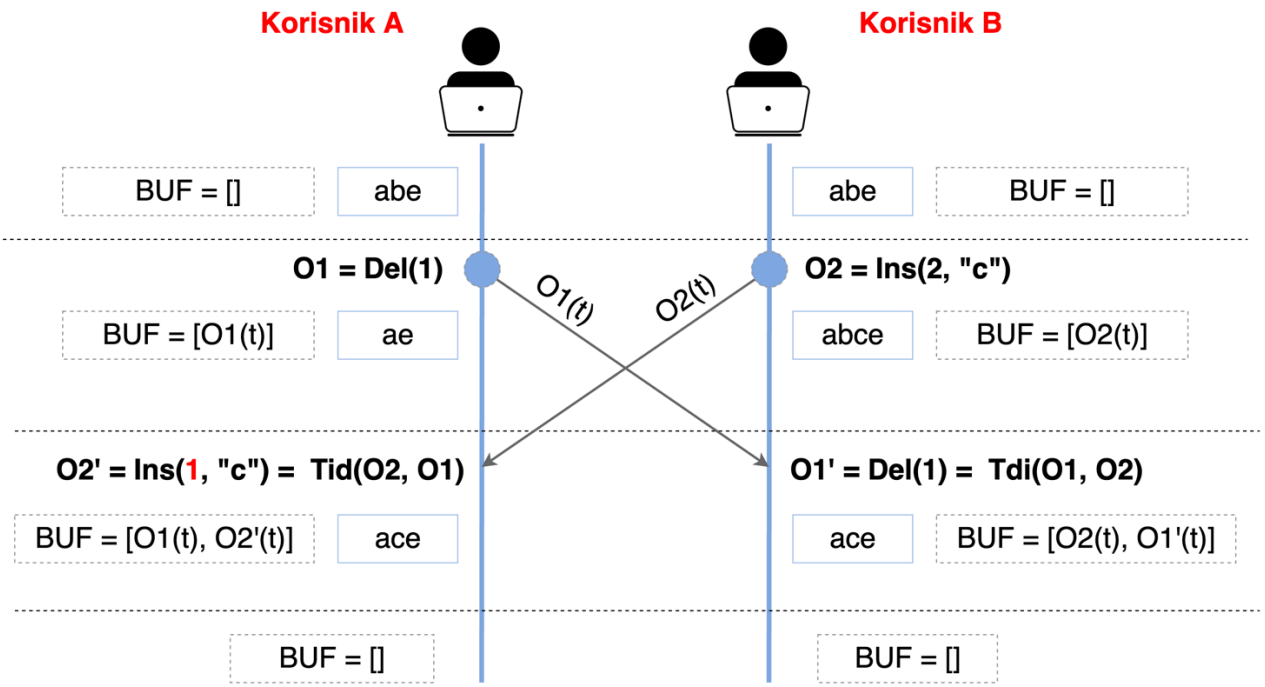
Početna pretpostavka: oba korisnika suradnju započinju s tekstom „abe“ u dokumentu i s praznim lokalnim privremenim spremnikom (na slici: *BUF*).

Manipulacija lokalnih operacija: korisnik A briše znak „b“ generirajući operaciju O1 = Del(1) koja rezultira tekstom „ae“. OT sustav kod korisnika A radi sljedeće: (1) dodaje vremensku oznaku *t* operaciji O1 $\rightarrow$ O1(t), (2) sprema operaciju O1 u privremeni spremnik $\rightarrow$ BUF = [O1(t)] i (3) propagira operaciju O1 drugoj strani. Za isto vrijeme Korisnik B u svom dokumentu umeće znak „c“ na poziciju s indeksom 2 te generira operaciju O2 = Ins(2, „c“). OT

sustav kod korisnika B radi isto kao i kod korisnika A: (1) operaciji O2 dodaje vremensku oznaku $t \rightarrow$ O2(t), (2) sprema operaciju O2 u lokalni privremeni spremnik $\rightarrow$ BUF = [O2(t)] i (3) propagira operaciju O2 drugoj strani.

Manipulacija pristiglih operacija: po dospijeću propagirane operacije O2(t) korisniku A, OT sustav operaciju transformira u O2'(t). Transformacija se vrši sljedećim koracima: transformacijski kontrolni algoritam, zbog istih vremenskih oznaka operacije O2(t) i operacije O1(t) iz lokalnog privremenog spremnika (operacije su izvršene u isto vrijeme), poziva transformacijsku funkciju te obje operacije prosljeđuje funkciji kao atribute. Kombinacija proslijeđenih operacija je umetanje - brisanje (engl. *insert - delete*) tako da se za transformaciju koristi transformacijska funkcija Tid(O2, O1). Transformacijska funkcija uspoređuje indekse pozicija operacija, u ovom slučaju kod operacija O2 = Ins(2, c) i O1 = Del(1) su to indeksi 2 i 1. Funkcija zaključuje da se operacija O2 izvodi desno od operacije O1 u linearnom stanju teksta (indeks pozicije O2 veći je od indeksa pozicije O1) i stoga prilagođava indeks pozicije operacije O2 s 2 na 1 kako bi se kompenzirao učinak operacije O1 (pomicanje pozicija ulijevo za 1). Time se operacija O2 transformirala u novu transformacijsku operaciju O2' = Ins(1, „c“). Sada se operacija O2' = Ins(1, „c“) izvršava nad dokumentom korisnika A te trenutni tekst „ae“ postaje „ace“. Operacija O2'(t) zapisuje se u lokalni privremeni spremnik.

S druge strane, kod korisnika B dolazi propagirana operacija O1(t). Proces transformacije sličan je kao kod korisnika A. Transformacijski kontrolni algoritam, zbog istih vremenskih oznaka operacije O1(t) i operacije O2(t) iz lokalnog privremenog spremnika, poziva transformacijsku funkciju te obje operacije prosljeđuje funkciji kao atribute. Ovog puta kombinacija operacija je obriši - umetni (engl. *delete - insert*) te se koristi operacijska funkcija Tdi(O1, O2). Ponovno se uspoređuju indeksi operacija. Ovog puta su to indeksi 1 i 2. Funkcija zaključuje da se u tekstu operacija O1 izvodi lijevo od operacije O2 (indeks pozicije O1 manji je od indeksa pozicije O2) te da pristigla operacija O1 nije utjecana od strane operacije O2. Zbog toga nema potrebe prilagođavati operaciju O1 za izvršavanje na tekstu korisnika B. Zato je transformirana operacija O1' jednaka operaciji O1 $\rightarrow$ O1' = O1 = Del(1). Transformirana operacija izvršava se na tekstu korisnika B te trenutni tekst „abce“ postaje „ace“. Transformirana operacija O1' se po izvršavanju zapisuje u lokalni privremeni spremnik. Nakon određenog vremena privremeni spremnik se prazni jer nema potrebe za stalnim pohranjivanjem izvršenih operacija.

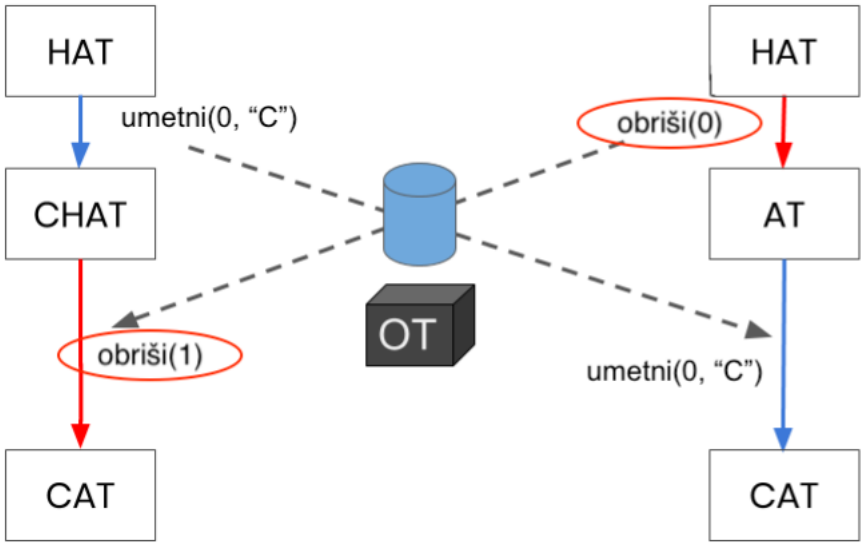


Slika 3.4. *Primjer sustava suradnje pokretanog OT tehnologijom*

Sljedeći tekst opisat će način kojim sustav suradnje temeljen na OT tehnologiji rješava slučaje suradnje iz Poglavlja 3, prikazane slikama 3.2. i 3.3.

Prvi slučaj suradnje je korištenje istovremenih operacije umetanja i brisanja. Početni tekst kod oba korisnika opet je „HAT“ i opet oba korisnika istovremeno rade operacije. Korisnik 1 operaciju umetanja znaka „C“ na početak teksta, a korisnik 2 operaciju brisanja znaka „H“. Nakon što su korisnici izvršili svoje operacije, korisnik 1 operaciju umetni(0, „C“) i korisnik 2 operaciju obriši(0), a prije izvršavanja operacije drugog suradnika, stanje njihovih tekstova isto je kao u primjeru iz Poglavlja 3. Korisnik 1 u svom dokumentu ima tekst „CHAT“, a korisnik 2 ima tekst „AT“. Međutim, u ovom trenutku sustav suradnje pokretan OT tehnologijom uspoređuje istovremene operacije te predviđa hoće li operacije uzrokovati da dokumenti ne konvergiraju. Ukoliko dokumenti ne bi konvergirali, operacije je potrebno transformirati. U ovom slučaju dokumenti ne konvergiraju stoga OT sustav zaključuje da za korisnika 1 operaciju brisanja (obriši(0)) treba transformirati prije izvršavanja. Na temelju lokalnih operacija korisnika 1 (umetanje znaka „C“ na poziciju 0), OT sustav zaključuje da su kod korisnika 1 pozicije ostalih znakova pomaknute za 1 te na temelju toga operaciju obriši(0) transformira u obriši(1). Pomaknuvši indeks pozicije operacije brisanja za 1, sada korisnik 1 više neće obrisati znak „C“ kao u primjeru iz Poglavlja 3, već će obrisati znak „H“ i time dobiti krajnji tekst „CAT“. Izvršavanjem transformirane operacije korisnika 2, korisnik 1 sada ima isto stanje teksta („CAT“) kao i korisnik 2. Njihovi dokumenti su konvergirali što znači da je

sustav suradnje uspješno riješio nedosljednosti operacija. Upravo opisani primjer suradnje korištenjem OT tehnologije uz operacije umetanja i brisanja opisan je Slikom 3.5.



Slika 3.5. *Sustav suradnje pokretan OT tehnologijom s istovremenom operacijom umetanja i brisanja*

Drugi slučaj suradnje iz Poglavlja 3 je istovremeno brisanje prvog znaka u tekstu. Ponovi li se taj slučaj u sustavu suradnje temeljenom na OT tehnologiji ishod će biti drugačiji. Opet oba korisnika imaju namjeru obrisati prvi znak u tekstu „CAT“ ne bi li rezultat bio tekst „AT“. Nakon što su generirali i izvršili operacije brisanja u lokalnim dokumentima – obriši(0), OT sustav suradnje prepoznaje namjeru oba korisnika, da su svojim operacijama zapravo oba željela obrisati znak „C“ s početka teksta. Zbog toga sustav suradnje transformira operacije drugog korisnika kako ne bi napravile nikakve promjene u tekstu. Transformirane operacije postaju idempotententne, njihovim izvršavanjem ne mijenja se stanje teksta korisnika. Oba korisnika suradnju završavaju s tekstom „AT“ u svom dokumentu, što im je i bila namjera. OT sustav suradnje osigurao je konvergenciju dokumenata te ispunio zamisli oba korisnika.

Tehnologija *operational transformation* je bila prva široko prihvaćena tehnologija za razvoj suradničkog uređivanja. Prvi tekstualni uređivači sa sustavom suradnje koristili su OT tehnologiju za suradnju u stvarnom vremenu. Međutim, oni koji su razvijali sustave suradnje zasnovane na OT tehnologiji zaključuju da je implementacija takvih sustava izrazito zahtjevna. Joseph Gentle, razvojni inženjer uređivača *Google Wave*, o OT tehnologiji izjavio je: „Postoji milijun algoritama s različitim kompromisima, uglavnom sadržanih u akademskim radovima.

Ispravna implementacija algoritama doista je teška i dugotrajna. *Waveu* je trebalo 2 godine da ga razvije i ako bismo ga danas prepisivali, trebalo bi skoro isto toliko da ga razvijemo drugi put.”. Iz ovih razloga, kao alternativna tehnologija OT-u, razvijena je tehnologija suradnje zasnovana na strukturi *conflict-free replicated data type*. CRDT tehnologija suradnje detaljnije je opisana u sljedećem poglavlju.

3.2. *Conflict-free replicated data type*

Tehnologija suradnje *conflict-free replicated data type* nastala je pokušajem unaprjeđenja i pojednostavljenja OT tehnologije. Generalna ideja CRDT tehnologije je postizanje konzistentnih podataka između svih replika bez potrebe za koordinacijom. Postoje različite vrste CRDT tehnologije za različite namjene, međutim, u ovom radu fokus je na suradnji u uređivaču teksta.

Razlika u suradnji pomoću CRDT tehnologije u odnosu na OT tehnologiju je što se OT zasniva na temeljnoj strukturi uređivača teksta i ne mijenja ju, dok CRDT tehnologija uvodi novu strukturu zapisa teksta. OT tehnologija svakom znaku u tekstu dokumenta definira vrijednost i apsolutnu poziciju te se potpuno oslanja na algoritam za postizanje konvergencije tekstualnih dokumenata svih strana koje sudjeluju u suradnji. CRDT tehnologija koristi drugi pristup te pojedini znak ne definira njegovom vrijednošću i apsolutnom pozicijom već koristi složeniju strukturu. Strukturu teksta klasičnog uređivača reprezentira na način da svaki znak definira kao objekt s određenim svojstvima. Koristeći složeniju strukturu samog teksta potreban je jednostavniji algoritam za sinkronizaciju dokumenata između strana suradnje.

Za razliku od OT algoritama, CRDT tehnologija suradnje ne zahtijeva povijest operacija kao što je bio slučaj kod spremanja operacija u privremeni spremnik kod OT-a. Također, ne zahtijeva niti otkrivanje istovremenosti operacija kako bi se osigurala dosljednost dokumenata. Umjesto toga, zbog korištenja prilagođene strukture podataka, sve istovremene operacije u CRDT sustavu izvorno su komutativne [15].

Kao i kod OT tehnologije, postoji nekoliko algoritama na kojima se CRDT tehnologija zasniva. Prvi takav algoritam za CRDT tehnologiju je WOOT (*Without Operational Transformation*) predstavljen 2005. godine. Kod CRDT tehnologije koja koristi WOOT pristup, tekst se reprezentira nizom podatkovnih objekata. Svaki objekt ima svoj nepromjenjivi identifikator te je reprezentiran ili vidljivim znakom u tekstualnom dokumentu ili obrisanim

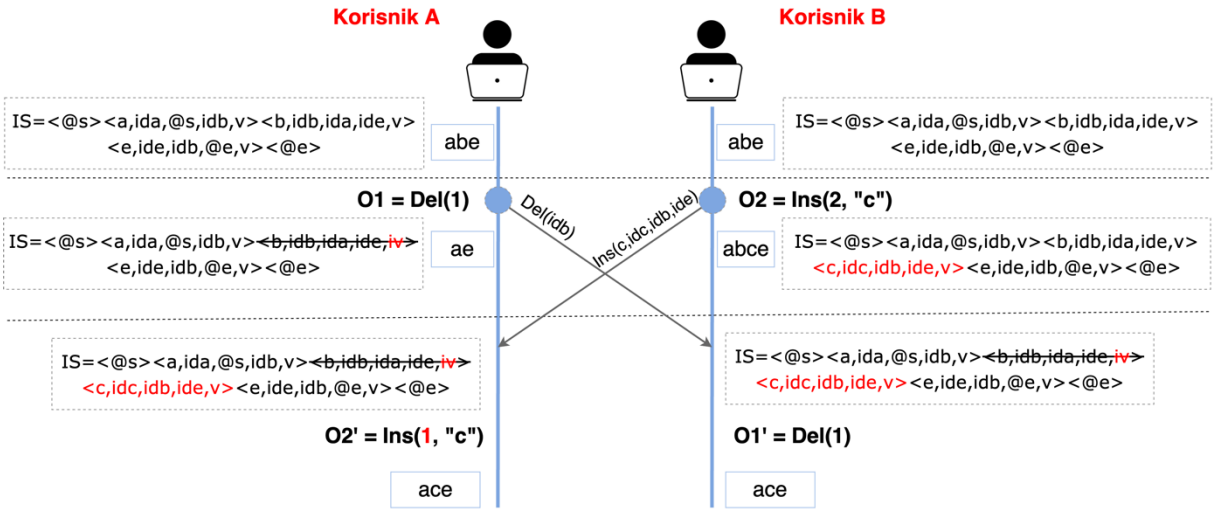
znakom. Objekt reprezentiram obrisanim znakom ne prikazuje se nikad u korisniku vidljivom tekstu. Takav objekt naziva se *tombstone*. Važno je shvatiti da CRDT tehnologija zapravo ne mijenja strukturu tekstualnog dokumenta niti operacije koje se vrše nad tekstom. U svrhu održavanja konzistentnosti, u pozadini održava unutarnje stanje dokumenta pomoću niza objekata te internih operacija. Dakle, u ovakvom sustavu suradnje postoji vanjsko i unutarnje stanje dokumenta kao i vanjske i unutarnje operacije. Vanjsko (ili vidljivo) stanje dokumenta su znakovi sa svojim vrijednostima i pozicijama kako ih korisnik vidi. Unutarnje stanje su zapravo isti ti znakovi, ali reprezentirani nizom objekata (model objekta opisan je kasnije u tekstu uz primjer korištenja CRDT tehnologije). Vanjske operacije su operacije zasnovane na poziciji (engl. *external position-based operations*) generirane od korisnika dok su unutarnje operacije iste te operacije, ali zasnovane na identifikatorima (engl. *internal identifier-based operation*).

Kao i kod OT tehnologije, podržane operacije uređivanja teksta su umetanje i brisanje znakova. Međutim, kod CRDT sustava suradnje, za umetanje znaka, unutarnja operacija mora sadržavati objekt koji se umeće u tekst (novi znak koji se umeće) te identifikatore njemu susjednih objekata koji su korisniku vidljivi u trenutku stvaranja operacije. Ako u tekstu „abc“ korisnik želi znak „x“ ubaciti na poziciju 1, generirana unutarnja operacija CRDT sustava, opisana riječima, bila bi „ubaci znak „x“ između znakova „a“ i „b“.

Životni ciklus operacije stvorene od korisnika u CRDT sustavu suradnje je sljedeći:

- Kada korisnik lokalno generira operaciju, ona se istog trena izvršava u vanjskom stanju dokumenta vidljivom korisniku. Zatim CRDT sustav tu korisnikovu vanjsku operaciju zasnovanu na pozicijama pretvara u unutarnju operaciju zasnovanu na identifikatorima. Unutarnja operacija izvršava se nad unutarnjim stanjem dokumenta, odnosno, nad nizom objekata. Posljednji korak je propagacija operacije zasnovane na identifikatorima na ostale strane koje sudjeluju u suradnji.
- Kod druge strane, po dolasku propagirane operacije, CRDT sustav unutarnju operaciju izvršava nad svojim unutarnjim stanjem dokumenta. Zatim CRDT sustav unutarnju operaciju zasnovanu na identifikatorima pretvara u vanjsku zasnovanu na pozicijama te tu operaciju primjenjuje nad vanjskim stanjem dokumenata što se prikazuje korisniku kao promjena u tekstu dokumenta.

U nastavku će biti opisan primjer korištenja CRDT sustava suradnje za uređivanje teksta, praćen Slikom 3.6.



Slika 3.6. *Primjer sustava suradnje pokretanog CRDT tehnologijom*

Početna pretpostavka: obje strane suradnje započinju uređivanje s vanjskim stanjem dokumenta koje je tekst „abe“. Unutarnje stanje dokumenta (IS – prema engleskom *internal state*), sastavljeno od niza objekata, odgovara vanjskom stanju, a izgleda ovako:

IS = <@s><a,ida,@s,idb,v><b,idb,ida,ide,v><e,ide,idb,@e,v><@e>

U navedenom formatu <@s> i <@e> su posebni objekti koji obilježavaju početak i kraj unutarnjeg stanja. Svaki od znakova vanjskog stanja dokumenta u unutarnjem stanju reprezentiran je objektom s pet atributa. Unutarnji objekt jednog znaka predstavljen je sljedećim modelom: <α, id, id_p, id_n, v> [16], gdje je α – vrijednost znaka, id – nepromjenjivi identifikator znaka, id_p – nepromjenjivi identifikator prethodnog objekta, id_n – nepromjenjivi identifikator sljedećeg objekta i v – oznaka je li znak vidljiv korisniku. Oznakom *v* označen je korisniku vidljivi element, a oznakom *iv* korisniku nevidljivi element. Takav nevidljivi element naziva se *tombstone*. Nepromjenjivi identifikator objekta (id) sastoji se od dva cijela broja (engl. *integer*). Prvi cijeli broj je identifikator strane (engl. *site id*) koja sudjeluje u suradnji – *sid*, dok je drugi broj redni broj operacije kojom je nastao taj znak, generirane od te iste strane – *seq*.

Manipulacija lokalnih operacija: korisnik A na sebi vidljivom vanjskom stanju objekta stvara operaciju zasnovanu na poziciji kojom briše znak „b“ (O1 = Del(1)), operacija se izvršava i korisnik dobiva vanjsko stanje dokumenta „ae“. CRDT sustav operaciju zasnovanu na poziciji O1 = Del(1) pretvara u operaciju zasnovanu na identifikatoru O1 = Del(id) na način da u nizu objekata unutarnjeg stanja traži objekt koji odgovara znaku na poziciji 1 u vanjskom stanju. To

čini na način da u nizu objekata broji vidljive objekte (oni koji imaju oznaku *v*) te uzima onaj na poziciji 1, što je u ovom slučaju objekt <b, idb, ida, ide, v>. CRDT sustav sada zna da je vanjska operacija brisanja znaka na poziciji 1 (Del(„1“)) jednaka unutarnjoj operaciji brisanja objekta s id-em *idb* (Del(idb)). Interna operacija Del(idb) se zatim izvršava nad unutarnjim stanjem dokumenata te se time objektu <b, idb, ida, ide, v> mijenja oznaka vidljivosti iz *v* u *iv* (postaje nevidljiv). Taj objekt sada postaje *tombstone*, ali i dalje se nalazi u unutarnjem stanju dokumenta, iako u vanjskom stanju dokumenta nije prikazan. Posljednji korak je slanje interne operacije drugim stranama suradnje, odnosno korisniku B.

U isto vrijeme korisnik B nad sebi vidljivom vanjskom stanju dokumenta stvara operaciju kojom umeće znak „c“ na poziciju 2 → O2 = Ins(2, „c“). Izvršava operaciju na vanjskom stanju dokumenta koja rezultira tekstom „abce“. CRDT sustav operaciju zasnovanu na poziciji O2 = Ins(2, „c“) pretvara u operaciju zasnovanu na identifikatoru Ins(α, id, id_p, id_n) na način da u nizu objekata unutarnjeg stanja traži objekte koji bi bili susjedni objekti znaku ubačenom na poziciju 2. Sustav susjedne objekte traži brojeći vidljive objekte u unutarnjem nizu objekata. Pronađeni prethodni susjedni objekt je <b,idb,ida,ide,v>, dok je sljedeći susjedni objekt <e,ide,idb,@e,v>. Pronalaskom susjednih objekata, sustav uspješno vanjsku operaciju pretvara u unutarnju → Ins(c, idc, idb, ide), gdje je *c* – vrijednost umetnutoga znaka, *idc* – identifikator umetnoga znaka, *idb* i *ide* – identifikatori susjednih objekata umetnog znaka. Kod operacije umetanja, susjedni objekti se koriste za pozicioniranje novog objekta. Novi objekt uvijek dolazi između susjednih objekta definiranih u unutarnjoj operaciji. Unutarnja funkcija se zatim izvršava nad unutarnjim stanjem dokumenata, što rezultira novim objektom <c,idc,idb,ide,v> umetnutim na ispravnu lokaciju u nizu objekata, između objekata s identifikatorom *idb* i *ide*. Posljednji korak je slanje unutarnje operacije korisniku A.

Manipulacija pristiglih operacija: na strani korisnika A, po dospijeću unutarnje operacije korisnika B – Ins(c, idc, idb, ide), operacija se izvršava nad nizom objekata tako da se u nizu objekata traže objekti s identifikatorima *idb* i *ide*. Iako je kod korisnika A u vanjskom stanju dokumenta znak „b“ već obrisan, njegov objekt s identifikatorom *idb* u unutarnjem stanju još uvijek postoji označen kao nevidljiv. Sustav pronalazi susjedne objekte te na poziciju između njih umeće novi objekt <c,idc,idb,ide,v>. Zatim sustav pristiglu unutarnju operaciju pretvara u vanjsku operaciju tako da broji vidljive objekte počevši od prvog u nizu pa do novo umetnutog objekta <c,idc,idb,ide,v>. Broj izbrojanih objekata odgovara poziciji na koju se znak „c“ mora umetnuti. U ovom slučaju broj izbrojanih objekata je 1, što rezultira vanjskom operacijom

umetanja znaka „c“ na poziciju 1 → Ins(1, „c“). Vanjska operacija izvršava se na vanjskom stanju dokumenta te korisnik A završava suradnju s vanjskim stanjem dokumenta „ace“.

Kod korisnika B, po dospijeću unutarnje operacije korisnika A (Del(idb)), operacija se izvršava na način da se u nizu objekata unutarnjeg stanja traži objekt s identifikatorom *idb.* Po pronalasku objekta, izvršava se unutarnja operacija brisanja. Operacija brisanja objektu mijenja oznaku vidljivosti iz *vidljiv* u *nevidljiv* (*v* → *iv*) te on postaje *tombstone*. Zatim se unutarnja operacija zasnovana na identifikatoru pretvara u vanjsku operaciju zasnovanu na poziciji. To se događa na način da se u unutarnjem stanju broje vidljivi objekti počevši od prvog pa do upravo izmijenjenog objekta (<b, idb, ida, ide, vi>). Broj izbrojanih objekata u ovom slučaju je 1, što znači da će vanjska operacija zasnovana na poziciji izgledati ovako: Del(1). Vanjska operacija brisanja izvršava se na vanjskom stanju dokumenta te je sada tekst vidljiv korisniku B „ace“.

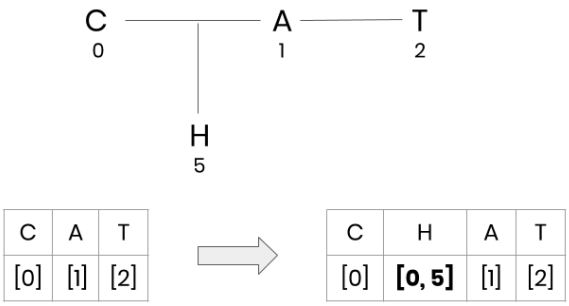
Ovim primjerom pokazano je da su dokumenti oba korisnika konvergirali u isto stanje. Krajnje vanjsko stanje dokumenta vidljivo korisnicima je „ace“, a unutarnje stanje dokumenta reprezentirano nizom objekata je:

IS = <@s><a,ida,@s,idb,v><b,idb,ida,ide,iv><c,idc,idb,ide,v><e,ide,idb,@e,v><@e>

Može se vidjeti da u unutarnjem stanju postoji objekt <b,idb,ida,ide,iv> koji nije vidljiv korisniku, takozvani *tombstone*. Primjerom je pokazano da se prilikom istovremenih uređivanja različitih korisnika, *tombstone* koristi za pronalazak pozicije kod unutarnjih operacija. Međutim, nije potrebno da se takvi objekti spremaju zauvijek, naprotiv, čišćenje takvih objekata smanjuje memoriju potrebnu za održavanje stanja dokumenta. WOOT algoritam *tombstone* objekte sprema zauvijek dok neki drugi CRDT algoritmi (primjerice RGA) imaju sustav za čišćenje takvih objekata, takozvani *garbage collection scheme*.

U prethodnom primjeru korišten je CRDT sustav suradnje zasnovan na WOOT algoritmu, međutim, iako je glavna ideja svih algoritama ista (složena struktura podataka), ne koriste svi algoritmi isti pristup upravljanju operacijama u međukorisničkoj suradnji. Razvojni inženjeri tekstualnog uređivača *Conclave* u svom istraživanju [17] predstavljaju i opisuju svoj algoritam korišten za CRDT suradnju. Kod ovog pristupa važno je istaknuti dva uvjeta koji garantiraju ispravnost suradnje. Prvi uvjet je da su znakovi u tekstu globalno jedinstveni, što znači da svaki znak u tekstu ima svoj nepromjenjivi jedinstveni identifikator koji je isti kod svih strana suradnje. Drugi uvjet su globalno poredani znakovi. Ovime se zahtjeva da su svi znakovi u tekstu jednako poredani te da poredak znakova ostaje uvijek konzistentan kod svih strana suradnje. Ovaj pristup za poziciju znakova ne koristi numeričke već frakcijske indekse (engl.

fractional indeces). Prednost koju donose frakcijski indeksi je taj što se operacijama umetanja ili brisanja znakova ne mijenja indeks pozicije ostalih znakova. Primjerice, u tekst „CAT“, koji se sastoji od znakova s indeksima: C → 0, A → 1, T → 2, želimo ubaciti znak „H“ između znaka „C“ i „A“. Frakcijski indeks znaka „H“ biti će 0.5, što pokazuje da adresira poziciju između znakova s indeksom 0 i 1. Za primijetiti je da se ostalim znakovima pozicijski indeks ne mijenja. Slika 3.7. opisuje primjer ubacivanja znaka „H“ u tekst „CAT“ te frakcijske indekse znakova.



Slika 3.7. *Frakcijski indeksi znakova*

U sljedećem dijelu rada pokazat će se kako se pomoću algoritma *Conclave*-a pristupa problematici predstavljenoj u Poglavlju 3. Koristit će se isti slučajevi istovremenog uređivanja. Prvi slučaj kod kojega oba korisnika u isto vrijeme brišu isti znak te drugi slučaj kod kojega jedan korisnik umeće znak, dok drugi u isto vrijeme briše znak.

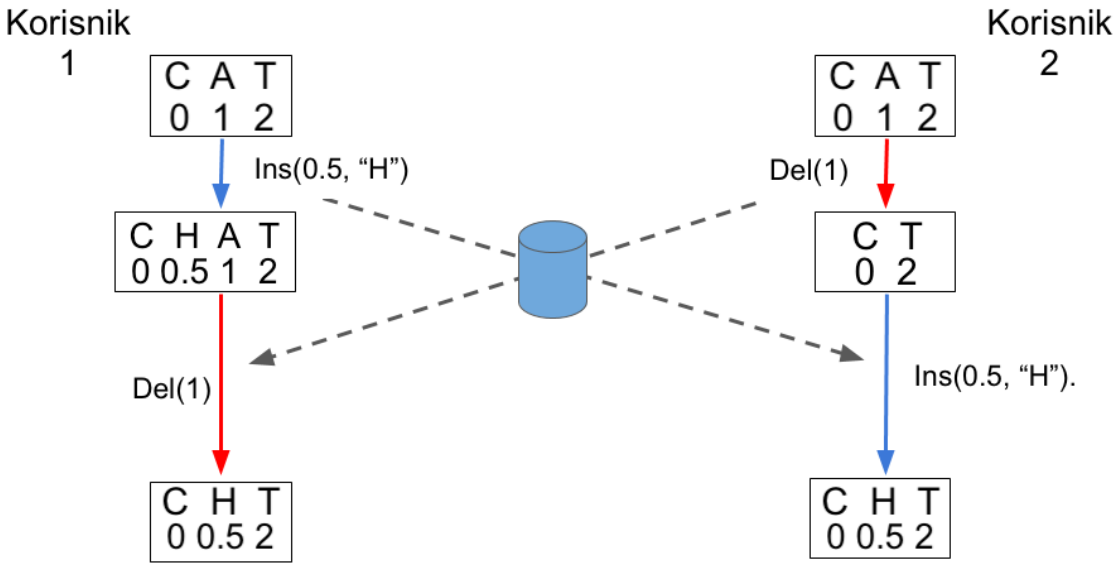
U prvom slučaju oba korisnika započinju suradnju s tekstom „HAT“, gdje su identifikatori znakova slijedno 1, 2, 3. Prvi korisnik želi obrisati znak „H“ te radi operaciju brisanja znaka s identifikatorom 1. Drugi korisnik također želi obrisati znak „H“ te u svom uređivaču u isto vrijeme radi istu operaciju brisanja znaka s identifikatorom 1 → Del(1). Nakon izvršenih lokalnih operacija, oba korisnika ostaju s tekstom „AT“, gdje su sada identifikatori znakova slijedno 2, 3. Nakon propagacije lokalnih operacija drugim stranama suradnje, prvi korisnik prima operaciju brisanja drugog korisnika. Međutim, on ju više ne može izvršiti jer u svom tekstu nema znak s identifikatorom 1. Isto se događa i kod drugog korisnika, ni on ne može obrisati znak s identifikatorom 1.

Ovime se pokazalo kako se ostvarila idempotentnost operacija; nije se dogodilo dvostruko brisanje. Također, oba korisnika su završila suradnju s istim tekstom „AT“, tj. njihovi dokumenti su konvergirali u isto stanje.

Kod drugog slučaja, praćenog Slikom 3.8., oba korisnika započinju uređivanje s tekstom „CAT“. Radi pojednostavljenja primjera u ovom slučaju su kao identifikatori znakova korišteni

njihovi frakcijski indeksi pozicije. Identifikatori znakova teksta slijedno su 0, 1, 2. Prvi korisnik želi ubaciti znak „H“ između znakova „C“ i „A“, te radi operaciju ubacivanja znaka „C“ na poziciju s frakcijskim indeksom 0.5 (između frakcijskih indeksa susjednih znakova: 0 < **0.5** < 1) → Ins(0.5, „C“). Nakon lokalnih promjena u dokumentu sada ima tekst „CHAT“ s identifikatorima znakova: 0, 0.5, 1, 2. Zatim svoju operaciju umetanja znaka propagira drugom korisniku. Za isto vrijeme drugi korisnik želi obrisati znak „A“ iz teksta te radi operaciju brisanja → Del(1). Nakon lokalnih promjena ostaje mu tekst „CT“ s identifikatorima znakova: 0, 2. Zatim svoju operaciju brisanja znaka propagira prvom korisniku. Prvi korisnik prima operaciju brisanja, u svom tekstu nalazi znak s identifikatorom 1 te izvršava operaciju brisanja. Tekst koji mu ostaje u dokumentu je „CHT“. Drugi korisnik prima operaciju umetanja znaka „H“. Zbog frakcijskog indeksa pozicije zna gdje umetnuti znak (0 < **0.5** < 2) te izvršava operaciju umetanja znaka. Suradnju završava s tekstom „CHT“.

Ovim primjerom je pokazano da su oba korisnika suradnju završila s istim tekstom, tj. tekst je konvergirao. Operacije su komutativne jer su kod korisnika izvršene različitim redoslijedom, a krajnje stanje njihovih tekstova je isto.



Slika 3.8. *Sustav suradnje pokretan CRDT tehnologijom s istovremenom operacijom umetanja i brisanja*

3.3. Razlike u složenosti OT i CRDT tehnologija suradnje

Iako je cilj obje navedene tehnologije omogućavanje suradnje u stvarnom vremenu, pristup koji koriste za postizanje toga cilja značajno se razlikuje. Svaka od tehnologija usvojila je različitu strategiju za rukovanje istovremenim operacijama. Kod OT sustava suradnje to je privremeni spremnik, a kod CRDT sustava suradnje je to unutarnje stanje tekstualnog dokumenta. Upravo je dizajnom tih komponenata tehnologije definirana njezina složenost korištenja [18].

Vremenska i memorijska složenost (engl. *time and space complexity*) kod OT sustava suradnje ovisi najviše o broju operacija pohranjenih u privremenom spremniku. Varijablom *c* označava se broj operacija pohranjenih u privremenom spremniku. Kod OT sustava, vrijednost varijable *c* vezana je za istovremenost suradnje te nije vezana za sadržaj tekstualnog dokumenta. Vrijednost varijable *c* prilikom neke normalne sjednice suradnje (tek nekoliko suradnika) kreće se između 0 i 10. Kod CRDT tehnologije, vremenska i memorijska složenost ovise o broju objekata unutarnjeg stanja tekstualnog dokumenta. Varijablom *C* označava se broj svih objekata unutarnjeg stanja nekog tekstualnog dokumenta. Vrijednost varijable *C* kod ove tehnologije nije vezana za istovremenost suradnje već za sadržaj tekstualnog dokumenta. Za tekst veličine od tisuću do milijun znakova, vrijednost varijable C kreće se između 10^3 i 10^6. U normalnim slučajevima suradnje, vrijednost varijable *c* znatno je manja od vrijednosti varijable *C*. To znači da je vremenska i memorijska složenost korištenja CRDT tehnologije znatno veća nego kod OT tehnologije.

Kod inicijalizacije suradničke sjednice, manja vremenska i memorijska složenost na strani je OT tehnologije. Prilikom pokretanja sjednice, privremeni spremnik OT tehnologije je prazan (nije se još izvršila nijedna operacija). To znači da kod OT tehnologije ne postoji vremenski i memorijski trošak (engl. *time and space cost*) inicijalizacije sjednice. S druge strane, za inicijalizaciju sjednice kod CRDT tehnologije nužno je da se za postojeći tekst dokumenta izgradi unutarnje stanje dokumenta koje odgovara početnom tekstu dokumenta. Izgradnja unutarnjeg stanja rezultira vremenskim i memorijskim troškom. Postojanje vremenskog i memorijskog troška rezultirat će sporijim odazivom sustava suradnje, a time i manje responzivnim uređivačem teksta.

Ako se govori o slijednim (operacije koje se izvršavaju jedna nakon druge) i istovremenim operacijama, kod OT sustava suradnje za slijedne operacije ne postoji vremenski i memorijski trošak. Transformacija operacije nije potrebna ako ne postoji još jedna

istovremena operacija. CRDT sustav suradnje svaku operaciju, neovisno bila ona slijedna ili istovremena, mora primijeniti na unutarnje stanje dokumenta. To znači da za svaku operaciju postoji vremenski i memorijski trošak, koji je otprilike podjednak za slijedne i istovremene operacije. Isto se događa i kod lokalnih i pristiglih operacija. CRDT sustav suradnje, kako lokalne, tako i pristigle operacije mora izvršiti nad unutarnjim stanjem tekstualnog dokumenta, što opet rezultira postojanjem vremenskog i memorijskog troška. Za izvršavanje lokalnih operacija u OT sustavu suradnje ne postoji trošak zbog toga što se lokalne operacije ne mogu dogoditi u istom vremenu. Kod pristiglih operacija trošak postoji samo ako je pristigla operacija istovremena s nekom iz privremenog spremnika.

4. RAZVOJ UREĐIVAČA TEKSTA SA SURADNJOM U STVARNOM VREMENU

Dio zadatka ovog diplomskog rada bio je razvoj vlastitog uređivača teksta sa suradnjom u stvarnom vremenu. Ovo poglavlje daje uvid u način kako je isti napravljen te korištene alate koji olakšavaju razvoj uređivača teksta sa suradnjom u stvarnom vremenu. Opisat će se na koji način su ostvarene pojedine funkcionalnosti uređivača i koji su alati korišteni za njih. Prije početka opisa razvoja daje se kratki opis razvijene aplikacije za bolje shvaćanje pojedinih komponenata i funkcionalnosti iste.

Ogledna aplikacija razvijena u sklopu ovog rada zamišljena je kao jednostavni upravitelj dokumentima. Korisnici na početnoj stranici mogu dodavati i brisati dokumente. Klikom na dokument korisnicima se otvara uređivač teksta za uređivanje istog. Funkcionalnosti koje uređivač nudi su suradnja u stvarnom vremenu te sustav upravljanja inačicama dokumenata.

Razvoj aplikacije započinje izradom klasične web aplikacije. Za razvoj pristupnog dijela (engl. *frontend*) korištene su dobro znane tehnologije temeljene na programskim jezicima HTML, CSS i *Javascript*. Također, korišten je i CSS radni okvir (engl. *framework*) *Bootstrap* za postizanje željenog izgleda aplikacije te postizanje responzivnog dizajna, kako na većim zaslonima računala tako i na manjim zaslonima mobilnih uređaja. Korištenjem *manifest.json* datoteke, za mobilne uređaje omogućeno je dodavanje aplikacije na početni zaslon uređaja. Dodavanjem aplikacije na zaslon mobilnog uređaja dobiva se dojam korištenja nativne mobilne aplikacije. Kao pozadinski servis (engl. *backend*) za upravljanje te pohranjivanje svih podataka korišten je *Google*-ov servis *Firebase*. Glavni alati na koje se ova aplikacija oslanja su *Realtime Database, Quill* i *Yjs*. Svi podaci koje aplikacija koristi (dokumenti i inačice dokumenata) pohranjuju se i dohvaćaju koristeći *Firebase*-ovu bazu podataka *Realtime Database*. Važno je napomenuti kako *Realtime Database* baza podataka nema veze s istovremenom suradnjom kod uređivanja teksta. *Realtime Database* se koristi isključivo zbog prigodne arhitekture za pohranjivanje podataka te zbog podrške za prikazivanje izmjena u stvarnom vremenu na ostalim podacima, kao što su naslov dokumenata, inačice i ostalo. Alat *Quill* se koristi za razvoj samog uređivača teksta. Posljednji je alat *Yjs*, koji se koristi za omogućavanje uređivanja teksta u stvarnom vremenu između više korisnika. Većina korištenih alata instalirana je koristeći paketni upravitelj (engl. *package manager*) *npm*. Prednosti korištenja *npm*-a su jednostavnost dodavanja raznih paketa u aplikaciju te pregled instaliranih paketa s njihovim verzijama u

package.json datoteci. Kroz daljnji tekst opisat će se razvoj glavnih funkcionalnosti aplikacije te alata korištenih za implementaciju istih.

4.1. Implementacija uređivača teksta

Quill je alat otvorenog koda namijenjen za izgradnju bogatog uređivača teksta [19]. Uređivač teksta podržan od alata *Quill* zasniva se na principu WYSIWYG („what you see is what you get“), što u prijevodu s engleskog jezika znači „ono što vidite je ono što dobivate“. Ovakav pristup korisniku u uređivaču omogućuje direktno uređivanje i manipuliranje izgledom sadržaja. Zbog modularne arhitekture uređivača teksta nudi se velik broj modula kojima se funkcionalnosti uređivača teksta mogu proširivati. Kako je Quill alat otvorenog koda svaki od modula moguće je dodatno prilagoditi vlastitim promjenama. Također, alat Quill nudi bogat API za upravljanje uređivačem teksta, kao što su izmjena, dohvaćanje te postavljanje teksta. Osim manipulacije tekstom, API ima podršku za osluškivanje pojedinih događaja (engl. *events*) nastalih tijekom uređivanja teksta. Format teksta s kojim ovi uređivači rade naziva se Delta. Delta je JSON format koja omogućava zapisivanje teksta i njegovih atributa na ljudima i računalima čitljiv način. Delta struktura iz Ispisa 4.1. u uređivaču teksta reprezentirala bi se ovako: „**Nebo** je plavo!“.

```
{
  ops: [
    { insert: 'Nebo', attributes: { bold: true } },
    { insert: ' je ' },
    { insert: 'plavo!', attributes: { color: '#0000ff' } }
  ]
}
```

Ispis 4.1. *Primjer Delta strukture*

U oglednoj aplikaciji uređivač teksta alata *Quill* korišten je s osnovnim funkcijama stiliziranja teksta. Programski kod Ispisa 4.2. pokazuje inicijalizaciju tekstualnog uređivača sa modulima za stiliziranje teksta. Jednom inicijaliziran uređivač teksta u aplikaciji se prikazuje bez ikakvog teksta, zbog čega je potrebno iz baze podataka dohvatiti tekst dokumenta.

```
const textEditor = new Quill(editorContainer, {
  modules: {
    toolbar: stylingModules,
    history: { userOnly: true },
    cursors: true
  },
  placeholder: 'Start typing...',
  theme: 'snow'
});
```

Ispis 4.2. *Inicijalizacija uređivača teksta pomoću alata Quill*

Tekst dokumenta zapisan je u bazi podataka u *document* modelu, a dohvaća se po učitavanju stranice koristeći sljedeći *event listener*: `window.addEventListener('load', (event) => { … });`. Pomoću *Firebase* servisa radi se spajanje na bazu podataka gdje se pomoću identifikatora dokumenta i njegove reference dohvaća njegov tekst. Tekst iz uređivača pohranjuje se u bazu podataka koristeći funkciju *Quill* API-a, odnosno koristeći *event listener* koji se odaziva na svaku promjenu u tekstu uređivača. Kako je zbog suradnje moguće da promjenu u tekstu radi neka druga strana suradnje, provjerava se je li promjenu napravio sam korisnik. Ukoliko je, tekst dokumenta se zapisuje u bazu podataka. Logika za pohranjivanje teksta prezentirana je u Ispisu 4.3.

```
textEditor.on('text-change', function(delta, oldDelta, source) {
   if (source == 'user') {
       updateDocContent(textEditor.getContents());
   }
});
```

Ispis 4.3. *Pohranjivanje teksta dokumenta*

4.2. Implementacija sustava suradnje u stvarnom vremenu

Mogućnost uređivanja teksta u stvarnom vremenu osigurana je koristeći *Yjs* razvojni okvir za suradnju u stvarnom vremenu zasnovanu na CRDT tehnologiji. Za sinkronizaciju podataka *Yjs* koristi dijeljene tipove (engl. *shared types*). *Shared types* su slični običnim tipovima podataka kao što su polja, skupovi i parovi, osim što se oni automatski sinkroniziraju kod različitih strana suradnje. Za sinkronizaciju teksta koristi se *shared type text*. Iako je primarno razvijen za sinkronizaciju teksta, može se koristiti i za sinkronizaciju ostalih objekata

koristeći pritom odgovarajuće *shared type*-ove. Podrške koje *Yjs* nudi, osim suradnje u stvarnom vremenu, još su uređivanje u uvjetima bez mrežne povezanosti, *undo/redo* funkcije uređivanja, upravljanje inačicama i dijeljenje pokazivača. Ogledna aplikacija od mogućnosti *Yjs*-a, uz suradnju u stvarnom vremenu, koristi još i dijeljenje pokazivača. *Yjs* je pogodan za suradnju velikog broja korisnika te dokumente s velikom količinom teksta [20].

Za instalaciju *Yjs* paketa korišten je paketni upravitelj npm. Za korištenje *Yjs* sustava suradnje potrebno je definirati vrstu dijeljenih podataka (*shared type*), povezati ih s uređivačem teksta te definirati protokol za komunikaciju između korisnika. Implementacija sustava suradnje započinje inicijalizacijom *Yjs* dokumenta. *Yjs* dokument je osnovna jedinica *Yjs* sustava koja se sinkronizira između korisnika. *Yjs* dokumentu se definira vrsta dijeljenog podataka, što je u ovom slučaju tekst. Zatim je potrebno dijeljeni tekst povezati s uređivačem. Kako *Yjs* sustav već ima podršku za neke uređivače teksta korišten je poveznik za *Quill* uređivač teksta (Ispis 4.4.).

```
const yjsDocument = new Y.Doc();
const sharedText = yjsDocument.getText('quill');
const binding = new QuillBinding(
    sharedText, textEditor, provider.awareness
);
```

Ispis 4.4. *Inicijalizacija Yjs dokumenta i povezivanje s uređivačem*

Protokol korišten za komunikaciju između korisnika je *WebSocket*. Korištenjem konvencionalnog klijent–poslužitelj modela, promjene u tekstu distribuiraju se putem poslužitelja klijentima koji su zapravo korisnici koji sudjeluju u suradnji. Pomoću *Yjs*-a komunikacijski pružatelj usluge (engl. *provider*) se definira koristeći URL poslužitelja, naziv sobe (engl. *room*) i *Yjs* dokument (Ispis 4.5.). Kao poslužitelj korišten je demonstracijski poslužitelj alata *Yjs*. Soba funkcionira kao *port* te je važno da svaki dokument za sinkronizaciju koristi „svoju“ sobu. To znači da će se u jednoj sobi uvijek izmjenjivati promjene korisnika za isti dokument. Inače, ako bi se uvijek koristila ista soba promjene bi se razmjenjivale između različitih dokumenata, a to bi u najgorem slučaju dovelo do gubitka teksta dokumenta. Zato se naziv sobe generira ovisno o identifikatoru tekstualnog dokumenta i takav koristi.

```
const documentRoom = `editor-doc-${docId}`;
const provider = new WebsocketProvider(
    'wss://demos.yjs.dev', documentRoom, yjsDocument
);
```

Ispis 4.5. *Inicijalizacija dijeljenog teksta i povezivanja s uređivačem*

Za prikazivanje pokazivača ostalih korisnika u uređivaču teksta koristi se funkcija svijesti (engl. *awareness*) *Yjs*-a. Funkcija svijesti omogućava da se informacije o korisniku prenose ostalim korisnicima te se veže na pružatelj putem kojeg se te informacije prenose između korisnika. Uz poziciju pokazivača, dodatne informacije o korisniku u ovoj aplikaciji su korisničko ime i boja pokazivača (Ispis 4.6.). Ime korisnika dohvaća se iz lokalne memorije web preglednika ako ga je korisnik postavio, a ako nije koristi se ime „Unnamed user“. Boja pokazivača korisnika se postavlja nasumice iz predefiniranog skupa boja.

```
const username = localStorage.username || 'Unnamed user';
provider.awareness.setLocalStateField('user', {
    name: username,
    color: getRandomColor();
});
```

Ispis 4.6. *Postavljanje korisničkog imena i boje pokazivača korisnika*

4.3. Implementacija sustava za upravljanje inačicama

Sustav za upravljanje inačicama implementiran je oslanjajući se na *Realtime Database* bazu podataka te alat *Quill*. U Poglavlju 4.1. rečeno je kako uređivač teksta sadržaj reprezentira kroz Delta strukturu. Ako bi se u inačicu teksta pohranjivao samo čisti tekst, izgubilo bi se stilsko oblikovanje teksta; zato je važno da inačica pohranjuje Delta strukturu teksta. Klikom na gumb za spremanje inačice dohvaća se ime inačice ukoliko je postavljeno, Delta struktura trenutnog teksta iz uređivača teksta i korisničko ime korisnika koji radi inačicu te se zatim koriste u funkciji koja sve navedeno sprema u bazu podataka (Ispis 4.7.)

```
createVersionBtn.addEventListener('click', () => {
    let editorContent = editor.getContents();
    let versionName = versionNameInput.value || 'Unnamed version';
    createVersion(editorContent, versionName);
})
```

Ispis 4.7. *Dohvaćanje podataka i poziv funkcije za stvaranje inačice*

Prikaz inačica vrši se dohvaćanjem inačica iz baze podataka koristeći identifikator dokumenta na kojeg su inačice vezane. Dohvaćeni podaci o inačicama prikazuju se umećući prigodne HTML elemente koji tvore karticu inačice. Za prikaz teksta inačice koristi se isti *Quill* uređivač teksta kao za uređivanje dokumenta, ali izmijenjen na način da služi samo kao preglednik teksta, bez mogućnosti uređivanja. Klikom na gumb za vraćanje teksta inačice dohvaća se Delta iz preglednika inačica te se umeće u uređivač teksta (Ispis 4.8).

```
const versionViewer = new Quill(viewerContainer, {
    placeholder: 'Version viewer...',
});
versionViewer.enable(false);

restoreVersionBtn.addEventListener('click', () => {
    let versionContent = versionViewer.getContents()
    textEditor.setContents(versionContent)
});
```

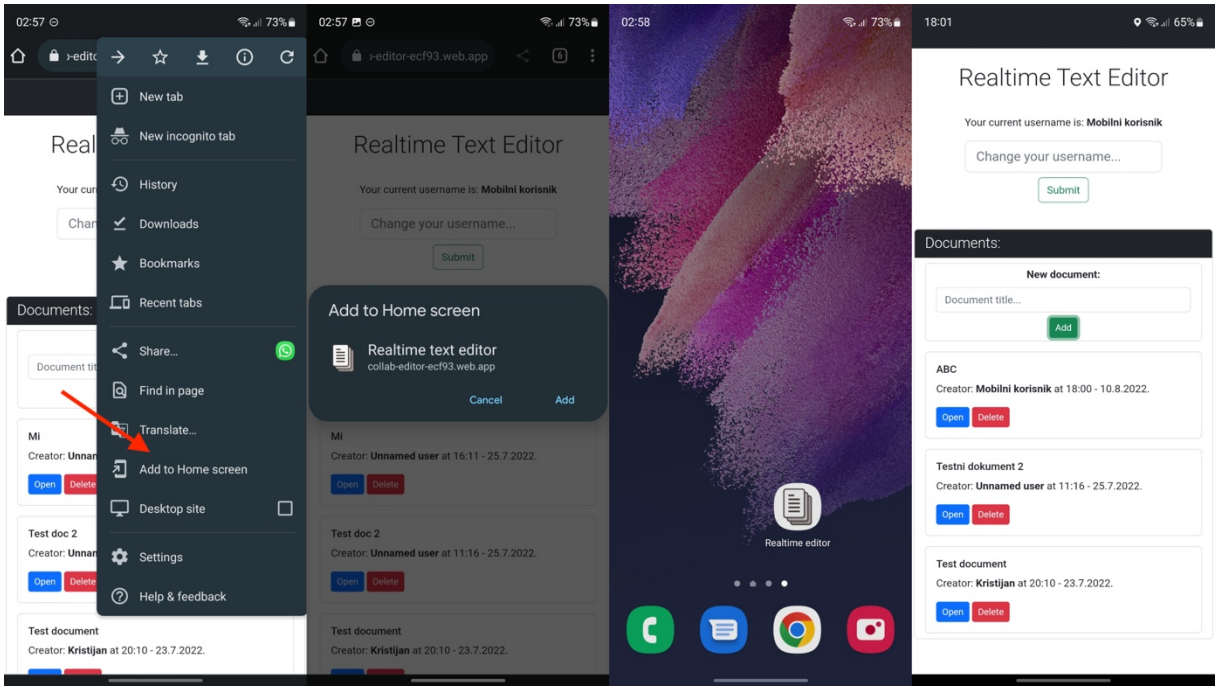
Ispis 4.8. *Inicijalizacija preglednika inačica i vraćanje teksta inačice*

5. IZVORNI UREĐIVAČ TEKSTA S MOGUĆNOŠĆU SURADNJE U STVARNOM VREMENU

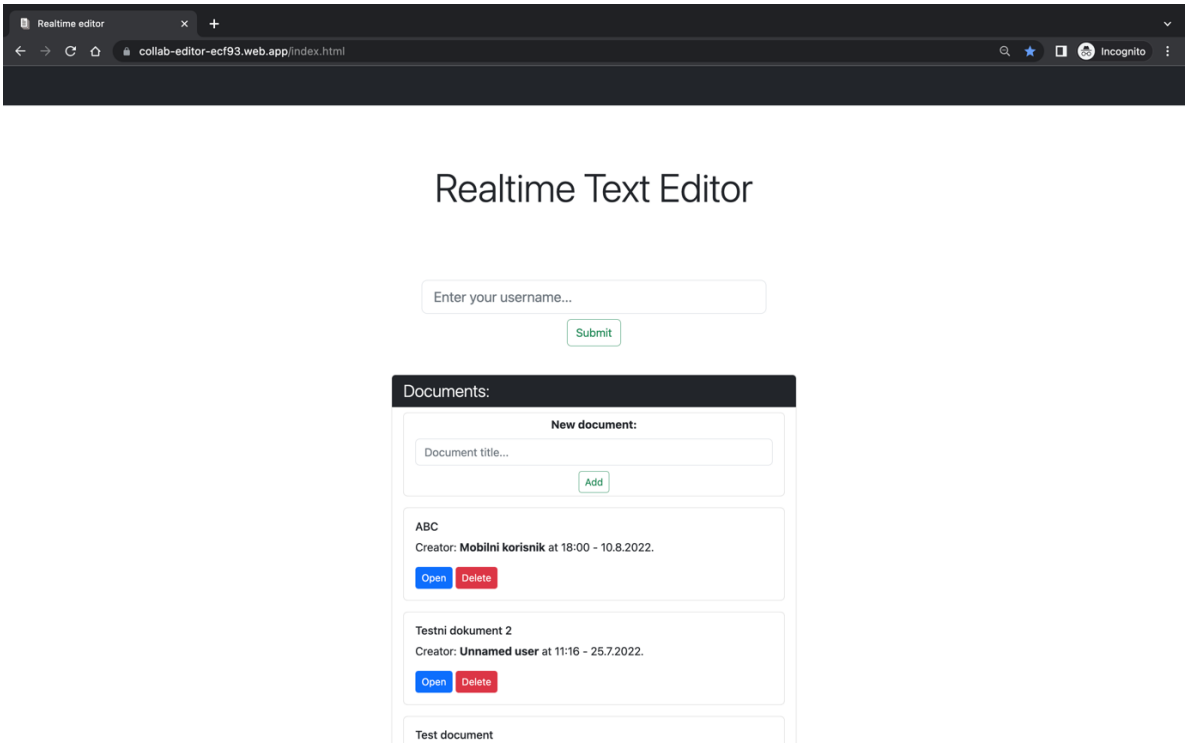
U ovom poglavlju opisat će se razvijena aplikacija. Objasnit će se princip korištenja aplikacije, upravljanje dokumentima, uređivač teksta, suradnja između korisnika i upravljanje inačicama. U svrhu preciznijeg pojašnjenja aplikacije, uz opis će se koristiti i zaslonske snimke. Nakon završenog razvoja, aplikacija je poslužena na *Firebase Hosting* usluzi gdje je javno dostupna za korištenje i testiranje.

Za pokretanje aplikacije nužno je biti povezan na internetsku mrežu te imati instaliran neki od web preglednika. Upisivanjem web adrese na kojoj se poslužuje aplikacija u preglednik, korisniku se otvara aplikacija. Korisnici mobilnih uređaja imaju mogućnost aplikaciju dodati na zaslon svog uređaja. Dovoljno je aplikaciju otvoriti u web pregledniku te u postavkama odabrati *Add to home page*. Aplikacija se tada nalazi u listi svih aplikacija na uređaju te se pokretanjem takve aplikacije dobiva iskustvo korištenja nativne mobilne aplikacije (ne prikazuje se alatna traka kao kad se aplikacija koristi u web pregledniku). Slika 5.1. prikazuje proces dodavanja aplikacije na zaslon mobilnog uređaja.

Dolaskom na početnu stranicu aplikacije korisnik dobiva uvid u svoje korisničko ime te listu postojećih dokumenata. Korisnik ima mogućnost postaviti svoje korisničko ime te, ako ga je već prije postavio, izmijeniti ga. Svi dokumenti u aplikaciji javno su dijeljeni te će svatko tko pokrene aplikaciju vidjeti iste dokumente. Osim pregleda, postoji i mogućnost dodavanja novih dokumenata na način da se upiše naslov dokumenta i pritisne gumb *Add.* Svaki dokument reprezentiran je karticom s podacima o dokumentu. To su naslov, autor dokumenta i vrijeme kada je dokument stvoren. Na svakoj kartici dokumenta su dva gumba *Open* i *Delete. Delete* gumb briše dokument iz baze podataka, dok gumb *Open* vodi na drugu stranicu gdje se otvara tekstualni uređivač za uređivanje teksta dokumenta. Promjene koje korisnici rade nad listom dokumenata (dodaju i brišu dokumente) u stvarnom vremenu se sinkroniziraju kod svih korisnika. Početna stranica prikazana na zaslonu računala i zaslonu mobilnog uređaja prikazana je slikama 5.1. i 5.2.



Slika 5.1. *Dodavanje aplikacije na početni zaslon mobilnog uređaja te početni zaslon aplikacije*

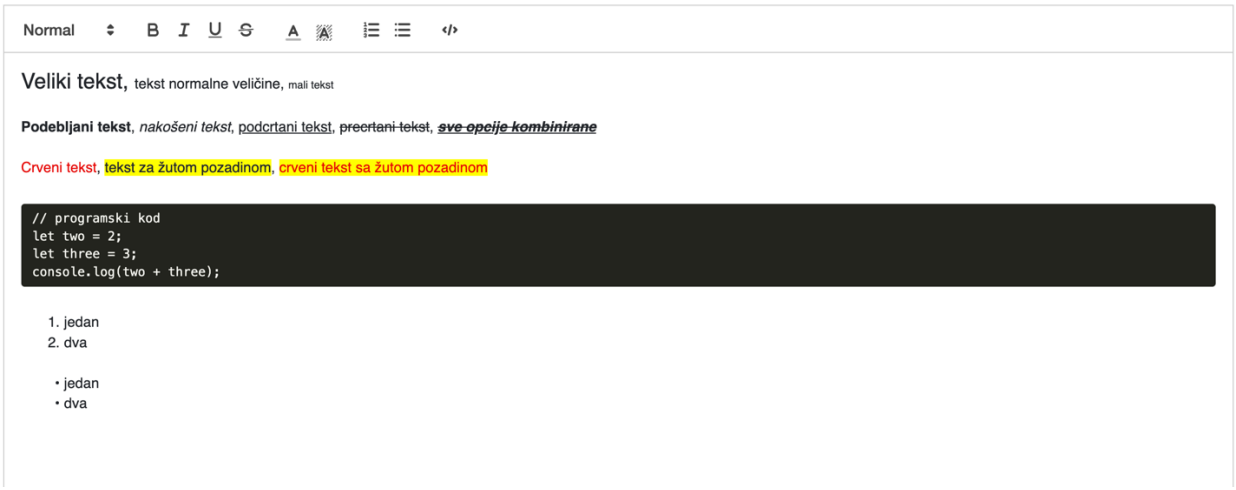


Slika 5.2. *Zaslonska snimka početne stranice na zaslonu računala*

Kako je spomenuto u prethodnom tekstu, pritiskom gumba *Open* na kartici dokumenta učitava se druga stranica gdje korisnici imaju mogućnost uređivati tekst dokumenta te upravljati inačicama. U navigacijskoj traci pri vrhu stranice nalazi se strelica za povratak na početnu stranicu te korisničko ime koje je korisnik postavio. Pri vrhu stranice prikazuje se naslov dokumenta. Klikom na naslov prikazuje se tekstualno polje gdje se naslov može izmjenjivati. Po završenoj izmjeni naslova, novi naslov se trenutno prikazuje i kod svih ostalih korisnika koji imaju „otvoren“ taj dokument te u listi dokumenata s početne stranice.

5.1. Uređivanje teksta

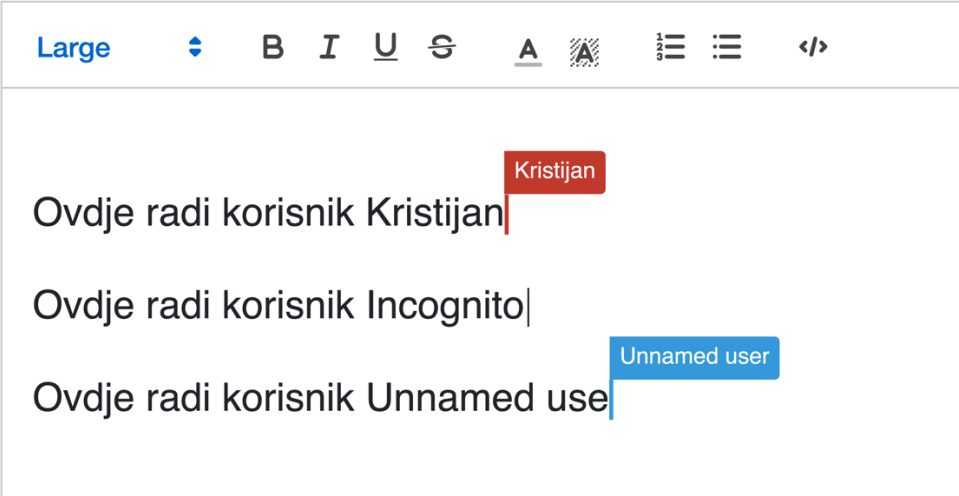
Ispod naslova dokumenta nalazi se bogati uređivač teksta za uređivanje teksta dokumenta. Opcije stiliziranja teksta redom su sljedeće: odabir veličine teksta, podebljanje, kurziv, podcrtavanje, precrtavanje, boja teksta, boja pozadine teksta, popis s rednim brojevima, popis s grafičkim oznakama i posljednje, stil programskog koda. Korisnik unutar uređivača može koristiti sve od navedenih opcija kako bi postigao željeno oblikovanje teksta. Također, većinu opcija stiliziranja može kombinirati. Primjerice, tekst male veličine, crvenu boju teksta, žutu boju pozadine i podebljanje. U uređivaču teksta, na označenom tekstu mogu se koristiti već poznati prečaci kod stiliziranja kao što su podebljanje (*ctrl + B*), kurziv (*ctrl + I*) ili podcrtavanje (*ctrl + U*). Osim klasičnih prečaca stiliziranja teksta ovaj uređivač ima svoj prilagođeni prečac za uklanjanje svih stilova s teksta. Kada je označen određen stilizirani tekst potrebno je pritisnuti tipke *ctrl + K* da bi se tekst „očistio“. Slika 5.3. pokazuje neke od opcija stiliziranja teksta u razvijenom uređivaču teksta.



Slika 5.3. *Opcije stiliziranja teksta*

5.2. Suradno uređivanje u stvarnom vremenu

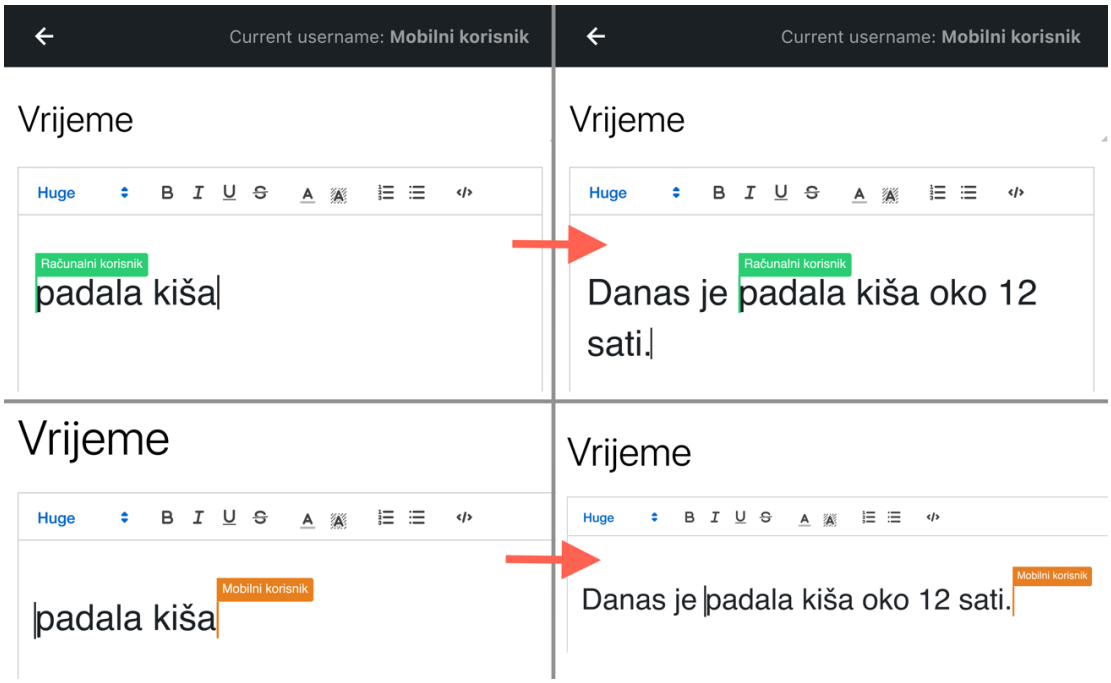
Kako bi više korisnika u isto vrijeme moglo sudjelovati u suradnom uređivanju, dovoljno je da svaki korisnik na svom uređaju pokrene aplikaciju te izabere isti dokument kao i ostali. U uređivaču teksta svi korisnici uvijek će imati isti tekst. Kako bi pojedini korisnik znao da u nekom trenutku nije jedini koji uređuje tekst, prikazuju se pokazivači ostalih aktivnih korisnika. Unutar teksta prikazuju se pokazivači na pozicijama gdje ostali korisnici uređuju tekst. Prijelazom miša preko pokazivača ostalih korisnika pokazuje se korisničko ime toga korisnika koje je sam postavio. Ako korisničko ime nije postavio, uz pokazivač tog korisnika pisat će „Unnamed user“. Osim prikaza imena uz pokazivač, pokazivač svakog korisnika vizualiziran je nekom bojom. Korisniku se njegova boja dodjeljuje otvaranjem dokumenta te je ta boja njegovog pokazivača ista kod svih ostalih korisnika. Slika 5.4. vizualizira prikaz pokazivača kako ih vidi korisnik sa korisničkim imenom „Incognito“.



Slika 5.4. *Prikaz vlastitog pokazivača i pokazivača ostalih korisnika*

Kako bi korisnici mogli uspješno surađivati i uređivati tekst, promjene pojedinog korisnika u stvarnom vremenu prikazuju se kod svih korisnika. Za to je zaslužan sustav suradnje u stvarnom vremenu. To znači da jedan korisnik može raditi na jednom dijelu dokumenta, dok drugi radi na drugom te će obojica vidjeti promjene koje radi onaj drugi. Suradnja između korisnika je ispravna neovisno koriste li računalo, mobilni uređaj ili neku drugu vrstu uređaja. To je moguće provjeriti praktičnim primjerom. Neka jedan korisnik koristi stolno računalo za uređivanje teksta, a drugi mobilni uređaj. Tekst koji uređuju je „padala kiša“. Korisnik stolnog računala će na početak teksta dodati „Danas je “, dok će korisnik mobilnog uređaja na kraj

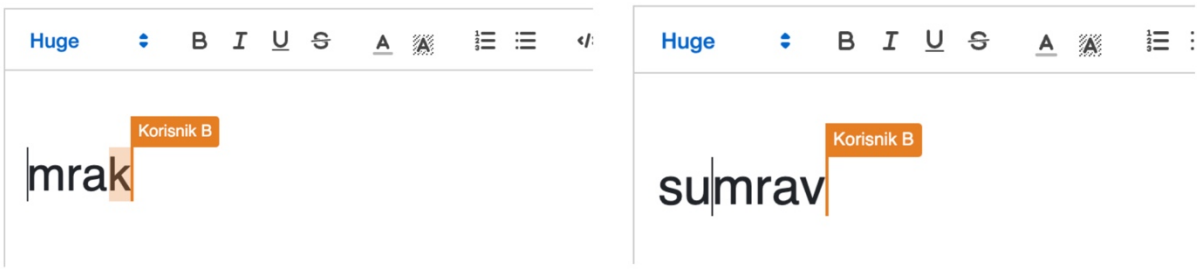
teksta dodati „ oko 12 sati.“. Nakon što oba, u isto vrijeme dodaju svoj tekst, rezultat će biti „Danas je padala kiša oko 12 sati.“. Slika 5.5. pokazuje stanje teksta prije i nakon uređivanja kako to vidi korisnik mobilnog uređaja (gornje dvije zaslonske snimke) i korisnik stolnog računala (donje dvije zaslonske snimke).



Slika 5.5. *Suradno uređivanje između mobilnog uređaja (gore) i stolnog računala (dolje)*

Tijekom suradnje više korisnika može se dogoditi da oni uređuju isti dio teksta s različitim namjerama. Tada nastaju konflikti. Sljedeća dva primjera pokazat će što se s tekstom događa kada dva korisnika naprave isključivi i neisključivi konflikt. U Poglavlju 2.2. dani su primjeri isključivog i neisključivog konflikta, a isti će se reproducirati i u produkcijskoj aplikaciji.

Primjer neisključivog konflikta bio je uređivanje teksta „mrak“. Prvi korisnik dodavao je riječi „su“ na početak teksta, dok je drugi znak „k“ zamijenio s „v“. Isti slučaj reproducirat će se u oglednom uređivaču teksta. Ispitivanje se izvelo koristeći dva računala, svako za jednog korisnika, gdje su oba korisnika u isto vrijeme napravila svoje promjene te je rezultat promjena tekst „sumrav“. Slika 5.6. prikazuje stanje teksta prije i nakon promjena, kako ga vidi korisnik A. Kao što je opisano u Poglavlju 2.2., vidi se da su promjene oba korisnika uspješno primijenjene na početni tekst. Međutim, kako su korisnici imali različite namjere uređivanja, a to su mogli vidjeti zbog podržane suradnje u stvarnom vremenu, ostaje na njima da dogovorom postignu rješenje prihvatljivo obojici.

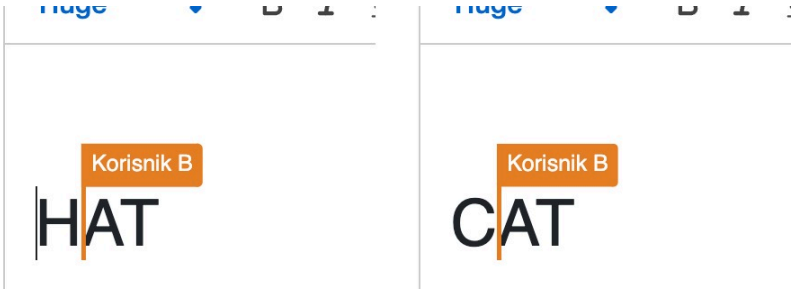


Slika 5.6. *Primjer neisključivog konflikta u oglednom uređivaču teksta*

Primjer isključivog konflikta u Poglavlju 2.2. bio je promjena veličine teksta. Isto će se ispitati i u oglednoj aplikaciji. Kao početno stanje koristit će se tekst normalne veličine te će jedan korisnik nad tim tekstom postavljati veliku veličinu teksta, a drugi malu veličinu teksta. U ispitivanju se promjene oba korisnika nastoje izvesti istovremeno. Ispitivanje se izvelo nekoliko puta te je rezultat uvijek bio primjena samo jedne od promjena korisnika, nekad je tekstu bila postavljena mala veličina, nekad velika. To potvrđuje tvrdnju iz Poglavlja 2.2., a to je da sustavi suradnje u stvarnom vremenu istovremene konflikte rješavaju na način da se prethodne promjene uvijek prepišu onom zadnjom. Iako su tekstovi oba korisnika konvergirali u isto stanje, problem ostaje što jedan korisnik nije svjestan namjere uređivanja drugog korisnika.

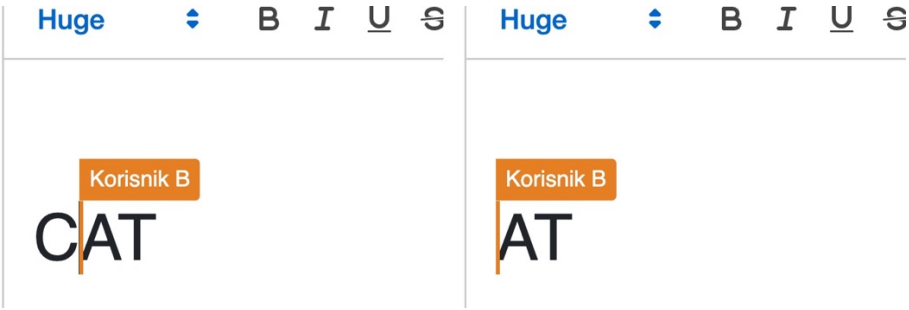
Sljedeći primjeri pomoću kojih je ispitana ispravnost sustava suradnje u stvarnom vremenu su slučajevi suradnje iz Poglavlja 3 prikazani slikama 3.2. i 3.3. U tim slučajevima suradnje važnost se pridaje komutativnosti i idempotenciji operacija. Koristeći ogledni uređivač teksta reproducirat će se prvo slučaj gdje jedan korisnik ubacuje znak, dok drugi briše znak. Nakon toga će se reproducirati slučaj kada oba korisnika brišu isti znak. Početni tekst i izvršene operacije biti će iste kao u Poglavlju 3.

Početna postavka prvog ispitivanja je tekst „HAT“ kod oba korisnika. Prvi korisnik radit će operaciju umetanja slova „C“ na poziciju 0, dok će drugi korisnik brisati znak „H“ s trenutne pozicije 0. Ispitivanje je izvedeno nekoliko puta s nastojanjem izvršavanja operacija u istom vremenu. Nakon svakog ispitivanja krajnji tekst kod oba korisnika uvijek je bio „CAT“ (Slika 5.7.). Operacije su u tekstu korisnika primijenjene drugim redoslijedom (prvo lokalna operacija korisnika te zatim operacija onog drugog korisnika), ali tekst je svejedno konvergirao. Ovime se pokazalo da je korišteni sustav suradnje uspješno osigurao komutativnost operacija oba korisnika.



Slika 5.7. *Tekst prije (lijevo) i nakon (desno) operacija umetanja i brisanja*

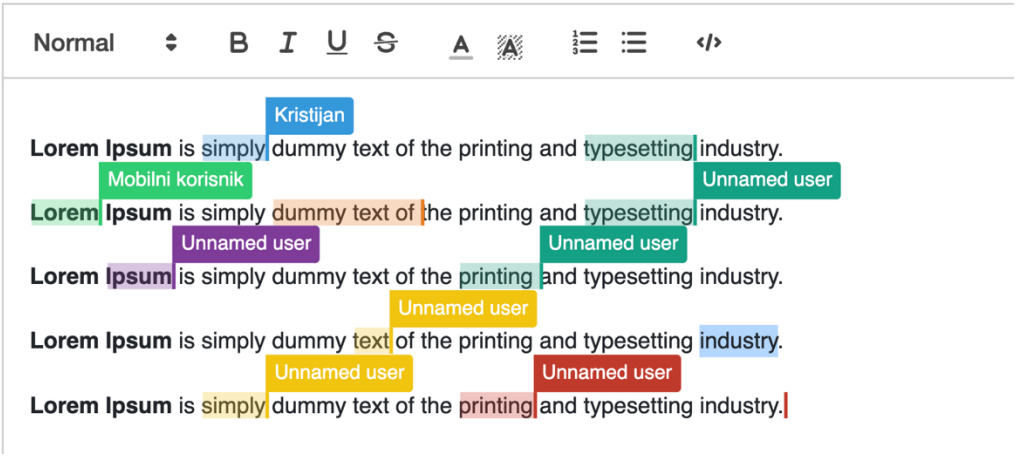
Početna postavka drugog ispitivanja opet je tekst „HAT“. Ovog puta oba korisnika radit će operaciju brisanja znaka „H“ iz teksta. Kao i u prethodnim slučajevima, ispitivanje će se ponoviti nekoliko puta s nastojanjem izvršavanja operacija u istom vremenu kod oba korisnika. Nakon svakog ispitivanja krajnji tekst kod oba korisnika bio je „AT“ (Slika 5.8.). Ispunjene su namjere oba korisnika te je tekst konvergirao. Oba korisnika izvršila su operaciju brisanja nad lokalnim tekstom, ali kada su se promjene propagirale onom drugom nije se dogodilo ponovno brisanje znaka. Ovime testnim slučajem se dokazalo da sustav suradnje osigurava idempotentnost operacija korisnika.



Slika 5.8. *Tekst prije (lijevo) i nakon (desno) operacija brisanja*

Sustav suradnje također podržava veći broj korisnika. Koristeći nekoliko stolnih računala i nekoliko mobilnih uređaja simulirano je uređivanje teksta s većim brojem suradnika. Slika 5.9. pokazuje kako izgleda uređivač teksta kada surađuje 12 korisnika. Svaki označeni dio teksta ili pokazivač predstavlja jednog korisnika. Kada istovremeno veći broj korisnika uređuje dokument, primjećuje se kratko vremensko kašnjenje prikazivanja tih promjena na dokumentu nekog drugog korisnika. To inače nije slučaj kod suradnog uređivanja manjeg broja korisnika. Međutim, trajanje tog vremenskog kašnjenja nije takve prirode da bi onemogućilo ili omelo suradnju korisnika u uređivanju teksta.

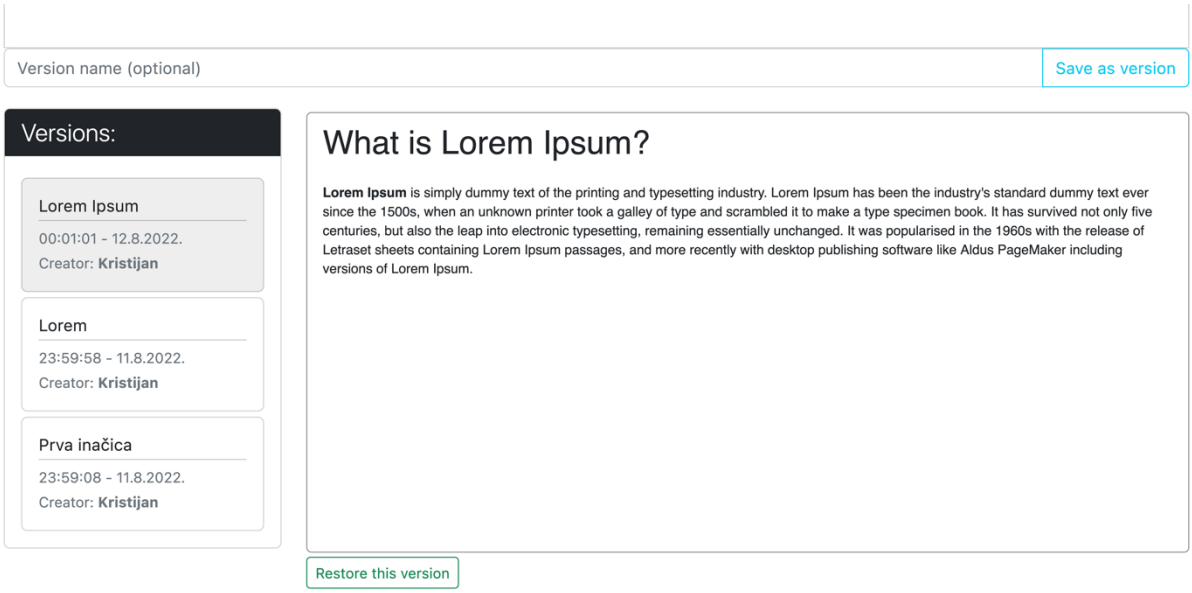
Ovaj, kao i sve navedene slučajeve u Poglavlju 5.2., svaki korisnik, neovisno o vrsti uređaja, može simulirati kako bi provjerio ispravnost suradnje u stvarnom vremenu.



Slika 5.9. *Suradnja većeg broja korisnika*

5.3. Upravljanje inačicama

Kada korisnik prilikom uređivanja teksta ima potrebu spremiti trenutni tekst to može napraviti stvaranjem inačice. Ogledna aplikacija ima sustav za upravljanje inačicama teksta dokumenta. U svakom trenutku korisnik može napraviti novu inačicu, pogledati neku od postojećih inačica ili vratiti tekst neke postojeće inačice u uređivač teksta. Korisnicima se ispod uređivača teksta nudi opcionalno postavljanje naziva inačice te gumb za stvaranje inačice. Ukoliko korisnik ne postavi naziv inačice, inačica dobiva naziv „Unnamed version“. Klikom na gumb *Save as version* stvara se nova inačica s trenutnim tekstom iz uređivača. Sve postojeće inačice dokumenta prikazuju se kao kartice u listi inačica. Svaka kartica predstavlja jednu inačicu te prikazuje naziv inačice, autora i vrijeme kad je napravljena. Kada korisnik klikne na neku od inačica u pregledniku inačica, pored liste inačice u pregledniku inačica prikazuje se spremljeni tekst. Korisnik iz preglednika može kopirati dio teksta koji mu je potreban ili klikom na gumb *Restore this version* trenutni tekst u uređivaču zamijeniti onim iz inačice. Slika 5.10. prikazuje upravo opisane opcije za upravljanje inačicama.



Slika 5.10. *Inačice tekstualnog dokumenta*

6. ZAKLJUČAK

Sustavi suradnje u stvarnom vremenu važna su odrednica suvremenih komunikacijskih i suradničkih aktivnosti. Svojim karakteristikama olakšavaju mnogim korisnicima i kompanijama postizanje željenih rezultata i ciljeva.

Današnji sustavi suradnje uglavnom se temelje na tehnologiji *operational transformation* ili tehnologiji korištenjem strukture podataka *conflict-free replicated data type*. Oba suradnju u stvarnom vremenom postižu korištenjem različitih pristupa. Za postizanje mogućnosti suradnje, OT sustav suradnje oslanja se isključivo na algoritam, koji je vrlo često zahtjevan za implementaciju. S druge strane CRDT sustav suradnje, kako bi izbjegao korištenje složenog algoritma, uvodi složeniju strukturu zapisa teksta. Složenijom strukturom teksta, operacije se više ne moraju transformirati kao kod OT sustava suradnje.

U sklopu ovog diplomskog rada razvijena je aplikacija s uređivačem teksta ove vrste. Zbog složenih implementacijskih rješenja suradnje u stvarnom vremenu, za razvoj uređivača koriste se alati koji olakšavaju izradu uređivača. Da je uređivač teksta uspješno implementiran pokazuje ispitivanje ispravnosti rješenja simulacijom specifičnih slučajeva suradnje. Stvaranjem i ponovnim vraćanjem inačica u razvijenom uređivaču, potvrđuje se ispravnost implementiranog sustava za upravljanje inačicama.

Implementirani uređivač teksta u budućnosti je moguće nadograditi sustavom komentiranja. U tom slučaju, označavanjem određenog dijela teksta te dodavanjem komentara na njega, suradnicima bi se omogućilo vođenje diskusije o tome što taj dio teksta predstavlja. Ovakvim sustavom dodatno bi se poboljšala suradnja između korisnika. Ostala dodatna moguća proširenja uređivača mogla bi uključivati i provjeru pravopisa, unos teksta govorom, spominjanje ostalih korisnika te sustav slanja obavijesti.

Ovim radom pokazalo se da se unatoč složenosti implementacije uređivača teksta sa suradnjom u stvarnom vremenu, korištenjem postojećih alata znatno može olakšati razvoj svih funkcija potrebnih za njegovo kvalitetno funkcioniranje u praksi.

7. LITERATURA

[1] Ahmed-Nacer, M. i dr.: „Evaluating CRDTs for Real-time Document Editing“, ACM, 11th ACM Symposium on Document Engineering, pp.103–112, California, 2011.

[2] Attiya, H. i dr.: „Specification and Complexity of Collaborative Text Editing“, PODC'16, Chicago, 2016.

[3] Flynn, P.: „Human interfaces to structured documents - The usability of software for authoring and editing“, National University of Ireland, Cork, 2014.

[4] Citro, S.; Ryan, C.; Mcgovern, J.: „Conflict Management For Real-Time Collaborative Editing in Mobile Replicated Architectures“, School of Computer Science and Information Technology, RMIT University, Melburne, 2007.

[5] Harrin, E.: „Collaboration Tools for Project Managers: How to Choose, Get Started and Collaborate with Technology“, Project Management Institute, Pennsylvania, 2016.

[6] Zagorskii, A.: „Operational Transformations as an algorithm for automatic conflict resolution“, s interneta, https://medium.com/coinmonks/operational-transformations-as-an-algorithm-for-automatic-conflict-resolution-3bf8920ea447, 06.09.2022.

[7] Google Docs, s interneta , https://docs.google.com/, 04.09.2022.

[8] Quang-Vinh, D.; Claudia-Lavinia, I.: „Performance of real-time collaborative editors at large scale: User perspective“, IFIP Networking, pp. 548-553., Lorraine, 2016.

[9] Github: Etherpad, s interneta , https://github.com/ether/etherpad-lite, 04.09.2022.

[10] Etherpad, s interneta , https://etherpad.org/, 04.09.2022.

[11] Ciocca, S.; Sisson, J.: „We Built Collaborative Editing for Our Newsroom's CMS.“, s interneta, https://open.nytimes.com/we-built-collaborative-editing-for-our-newsrooms-cms-here-s-how-415618a3ec49, 02.09.2022.

[12] Prosemirror, s interneta , https://prosemirror.net/, 04.09.2022.

[13] Kumawat, S.; Ajay, K.: „A Survey on Operational Transformation Algorithms: Challenges, Issues and Achievements“, International Journal of Computer Applications, pp. 787-1115., Jaipur, 2010.

[14] Ellis, C.A. i Gibbs, S.J.: „Concurrency Control in Groupware Systems“, MCC, Texas, 1989.

[15] Ahmed Nacer, M. i dr.: „Evaluating CRDTs for Real-time Document Editing“, Proceedings of the 11th ACM Symposium on Document Engineering, California, 2011.

[16] Gérald, O. i dr.: „Real time group editors without Operational transformation“, Research Report, RR-5580, INRIA, Villers-Lès-Nancy, 2005.

[17] Conclave, s interneta , https://conclave-team.github.io/conclave-site/, 07.09.2022.

[18] Sun, G. i dr.: „Real Differences between OT and CRDT under a General Transformation Framework for Consistency Maintenance in Co-Editors“, Nanyang Technological University, Singapur, 2020.

[19] Github: Quill, s interneta , https://github.com/quilljs/quill/, 08.09.2022.

[20] Github: Yjs, s interneta , https://github.com/yjs/yjs, 08.09.2022.

8. POPIS KRATICA

OT - Operational transformation

CRDT - Conflict-free replicated data type

WOOT - Without operational transformation

HTML - Hypertext Markup Language

CSS - Cascading Style Sheets

API - Application Programming Interface

JSON - JavaScript Object Notation

URL - Uniform Resource Locator

SAŽETAK

Uređivači teksta s mogućnošću suradnje u stvarnom vremenu omogućavaju većem broju korisnika istovremeno uređivanje i pregled svih promjena. Kako bi se omogućio takav kontekst rada, nužno je primijeniti neki od postojećih sustava suradnje. U ovome radu prezentirana je problematika suradnje u stvarnom vremenu te dva najčešće korištena principa za izgradnju sustava suradnje koja čine osnovu modernih uređivača teksta: transformacija operacija (engl. Operational Transformation, OT) i beskonfliktni replicirani tip podataka (engl. Conflict-free Replicated Data Type, CRDT). U sklopu rada implementiran je ogledni uređivač teksta s mogućnošću suradnje u stvarnom vremenu. Detaljno je opisan postupak razvoja te važni programski resursi korišteni za razvoj uređivača teksta. Funkcionalnost implementiranog rješenja ispitana je u karakterističnim slučajevima korištenja, kako u stolnoj tako i u mobilnoj domeni. Dodatno, uređivač teksta nadograđen je sustavom za upravljanje inačicama tekstualnih dokumenata.

Ključne riječi: uređivač teksta, suradnja u stvarnom vremenu, upravljanje inačicama, transformacija operacija (OT), beskonfliktni replicirani tip podataka (CRDT)

ABSTRACT

Real-time collaboration text editors enable a large number of users to simultaneously edit and review all text changes. In order to enable such a context, it is necessary to apply some of the existing collaboration systems. This thesis presents the issue of collaboration in real time and the two most commonly used principles for building collaboration systems that form the basis of modern text editors: Operational Transformation (OT) and Conflict-free Replicated Data Type (CRDT). As part of the work, a demonstrational real-time collaboration text editor was implemented. The development process and important software resources utilized for the development of the text editor are described in detail. The functionality of the implemented solution was tested in typical use cases, both in the desktop and mobile domains. Additionally, the text editor has been upgraded with a text document versioning system.

Keywords: text editor, real-time collaboration, versioning, Operational Transformation (OT), Conflict-free Replicated Data Type (CRDT)