**ROMÂNIA**

**MINISTERUL APĂRĂRII NAȚIONALE**

**ACADEMIA TEHNICĂ MILITARĂ ,,FERDINAND I”**

**FACULTATEA DE SISTEME INFORMATICE ȘI SECURITATE CIBERNETICĂ**

**Specializare: Calculatoare și Sisteme Informatice pentru Apărare și Securitate Națională**



**SOLUŢIE DE SECURIZARE A SERVICIILOR INTEGRATE ŞI ANONIMIZAREA ACCESULUI LA INFRASTRUCTURI VIRTUALIZATE**

CONDUCĂTOR ȘTIINȚIFIC:

**Col. prof. univ. dr. ing. Mihai TOGAN**

ABSOLVENT:

**Student plt. Ionuţ-Alexandru PAVEL**

Conţine \_\_\_\_\_\_\_\_\_\_\_ file

Inventariat sub nr. \_\_\_\_\_\_

Poziţia din indicator: \_\_\_\_

Termen de păstrare: \_\_\_\_\_

**BUCUREŞTI**

**2022**

# ABSTRACT

Collaboration tools involve multiple types of software or services that allow people to work toghether. Collaboration tools can route work through a process, distribute pieces and tasks to involved parties, and help to coordinate activities.

This paper presents the implementation of the web-based collaborative real-time application using ReactJs and NodeJs frameworks. The application provides a secure, controlled environment for real-time document review, coauthoring and redaction. Also, it allows the creation of private or group conversation beetween users and allows text message transfer. Documents can be saved in encrypted format using server storage and personal data is stored using MySql relational database. The application also ensures communication between users by creating private or public chat channels. Communication between users is secure and allows the exchange of text messages or the transmission of media files. Another communication method provided by the application is video calling, allowing users to interact directly with each other. The software solution also provides the functionality for users with an admin role to monitor the activity of other users and restrict their access to the application.

The final goal of the project is the development and operationalization of the presented software solution based on the studied concepts. An important aspect to take into account is that the application at the moment is not a final product to be put into use, but it is a demo version that puts the theoretical concepts into practice.

# REZUMAT

Cuprins

[ABSTRACT 5](#_Toc106979747)

[REZUMAT 6](#_Toc106979748)

[LISTĂ DE ABREVIERI 9](#_Toc106979749)

[TABELĂ FIGURI 10](#_Toc106979750)

[1. Introducere 11](#_Toc106979751)

[1.1 Importanţa temei 11](#_Toc106979752)

[1.2 Scopul lucrării 11](#_Toc106979753)

[1.3 Utilitatea sistemului 11](#_Toc106979754)

[2 Stadiul actual 12](#_Toc106979755)

[2.1 Software colaborativ 12](#_Toc106979756)

[2.2 Sisteme similare 12](#_Toc106979757)

[2.2.1 Microsoft Teams 12](#_Toc106979758)

[2.2.2 Slack 13](#_Toc106979759)

[2.2.3 Discord 14](#_Toc106979760)

[2.2.4 Telegram 15](#_Toc106979761)

[2.2.5 Google Docs 16](#_Toc106979762)

[2.3 Principii de funcţionare 17](#_Toc106979763)

[2.3.1 Editarea colaborativă 18](#_Toc106979764)

[2.3.2 OT 19](#_Toc106979765)

[2.3.3 CRDT 20](#_Toc106979766)

[2.3.4 WebRTC 21](#_Toc106979767)

[2.3.5 Signal Protocol 23](#_Toc106979768)

[3 API-uri folosite 23](#_Toc106979769)

[3.1 Aplicaţia Server 24](#_Toc106979770)

[3.2 Aplicaţia Client 25](#_Toc106979771)

[3.3 SocketIo 27](#_Toc106979772)

[3.4 Quill 28](#_Toc106979773)

[4 Structura proiectului 29](#_Toc106979774)

[4.1 Cerinţele sistemului 29](#_Toc106979775)

[4.1.1 Cerinţe Non-Funcţionale 30](#_Toc106979776)

[4.1.2 Cerinţe Funcţionale 30](#_Toc106979777)

[4.2 Arhitectura sistemului 32](#_Toc106979778)

[4.2.1 Arhitectura aplicaţiei client 32](#_Toc106979779)

[4.2.2 Arhitectura aplicaţiei server 34](#_Toc106979780)

[4.2.3 Persistenţa datelor 34](#_Toc106979781)

[5 Implementarea software 35](#_Toc106979782)

[5.1 Aplicaţia client 35](#_Toc106979783)

[5.2 Aplicaţia web server 37](#_Toc106979784)

[5.2.1 Răspunderea request-urilor 37](#_Toc106979785)

[5.3 Implementarea bazei de date 37](#_Toc106979786)

[5.3.1 Diagramă 37](#_Toc106979787)

[5.4 Metode de securizare 37](#_Toc106979788)

[5.4.1 Securizarea canalului de comunicaţie 37](#_Toc106979789)

[5.4.2 Criptarea fişierelor stocate 37](#_Toc106979790)

[5.4.3 Protecţie împotriva atacurilor CSRF 37](#_Toc106979791)

[5.4.4 Controlul accesului prin firewall 37](#_Toc106979792)

[5.4.5 Alertare în cazul încercărilor neautorizate 37](#_Toc106979793)

[6 Utilizarea aplicaţiei 37](#_Toc106979794)

[6.1 Actorii sistemului 37](#_Toc106979795)

[6.2 Cazurile de utilizare 37](#_Toc106979796)

[6.3 Testarea sistemului 37](#_Toc106979797)

[6.4 Raport de testare 37](#_Toc106979798)

[6.5 Diagrame UML 37](#_Toc106979799)

[6.5.1 Diagrama cazurilor de utilizare 37](#_Toc106979800)

[6.5.2 Diagrama de activităţi 37](#_Toc106979801)

[7 Concluzii 37](#_Toc106979802)

[7.1 Analizarea rezultatelor obţinute 37](#_Toc106979803)

[7.2 Dezvoltări ulterioare 37](#_Toc106979804)

[8 Bibliografie 37](#_Toc106979805)

[9 Anexe 39](#_Toc106979806)

# LISTĂ DE ABREVIERI

|  |  |  |
| --- | --- | --- |
|  | WebRTC | Web Real-Time-Comunication |
|  | VoIP | Voice over Internet Protocol |
|  | AES | Advanced Encryption Standard |
|  | MitM | Man-in-the-Midle |
|  | ALTS | Application Layer Transport Security |
|  | TLS | Transport Layer Security |
|  | OT | Operaţional Transformation |
|  | CRDT | Conflict-free Replicated Data Type |
|  | API | Application Programming Interface |
|  | HTTP | Hypertext Transfer Protocol |
|  | ICE | Interactive Connectivity Establishment |
|  | STUN | Session Traversal Utilities for NAT |
|  | TURN | Traversal Using Relay NAT |
|  | UDP | User Datagram Protocol |
|  | NAT | Network Address Translation |
|  | DTLS | Datagram Transport Layer Security |
|  | TLS | Transport Layer Security |
|  | E2EE | End-to-End Encryption |
|  | JSON | JavaScript Object Notation |
|  | MVC | Model View Controller |
|  | DOM | Document Object Model |
|  | URL | Uniform Resource Locator |
|  | CORS | Cross-Origin Resource Sharing |
|  | CRUD | Create, Read, Update, Delete |
|  | CSS | Cascading Style Sheets |
|  | LESS | Leaner Style Sheets |
|  | JWT | Json Web Token |

# TABELĂ FIGURI

# 1. Introducere

## Importanţa temei

Progresul tehnologic şi necesitatea diminuării timpul de muncă împreună cu nevoia oameniilor de a menţine comunicaţia la distanţă au dus la dezvoltarea de aplicaţii software colaborative care să satisfacă aceste cerinţe într-o manieră uşor de gestionat pentru utilizator. Astfel aplicaţiile colaborative sunt dezvoltate cu scopul de a ajuta persoanele implicate în proiecte comune să îşi atingă obiectivele minimizând resursele necesare de spaţiu şi timp. Toate aceste acţiuni trebuie să fie desfăşurate într-un mediu care oferă siguranţă atât pentru utilizator, cât şi pentru pentru datele care sunt prelucrate.

## Scopul lucrării

Prezenta lucrare are ca obiectiv detalierea şi formarea cunoştiinţelor necesare pentru dezvoltarea şi operaţionalizarea unei soluţii software „on-premise” (implementată folosind resurse proprii) care să asigure activitatea colaborativă la nivelul unei organizaţii guvernamentale. Lucrarea va cuprinde informaţii despre realizarea arhitecturii aplicaţiei software, detalii despre tehnologiile folosite, informaţii despre mecanismele de securizare folosite şi parcursul instalării pe o infrastructură virtualizată proprie. De asemenea, în proiect vor fi prezentate cazurile de utilizare şi se va întocmi un raport de testare.

## Utilitatea sistemului

Organizaţiile guvernamentale lucrează cu date clasificate, care nu pot fi prelucrate folosind soluţii software externe din raţiuni de securitate. În acelaşi timp există riscul ca o aplicaţie pusă la dispoziţie de un dezvoltator extern să colecteze date cu privire la utilizator, precum locaţia acestuia, adresa IP sau activitatea acestuia la nivelul dispozitivului pe care este folosită. Astfel, apare necesitatea unei soluţii dezvoltate şi administrate de organizaţia în cauză, pentru a reduce riscul colectării şi interceptării datelor.

În acelaşi timp folosirea unei aplicaţii realizate în mediul intern oferă control absolut asupra fluxului de date (datele sunt stocate intern), dar şi asupra utilizatorilor, în cazul în care este detectată activitate suspectă poate fi restricţionat accesul la resursele interne. Regulile pe care le respectă organizaţiile guvernamentale nu permit folosirea de resurse Cloud externe pentru stocarea informaţiilor, deoarece furnizorii de servicii Cloud nu pot dovedi integritatea datelor pe care le administrează, dar şi din raţiuni politico-geografice. Soluţia propusă va îndeplinii criteriile de securitate specifice organizaţiilor guvernamentale, punând accent pe securizarea canalului de comunicaţie, stocarea datelor, dar şi monitorizării activităţii utilizatorilor.

# Stadiul actual

## Software colaborativ

Aplicaţiile software colaborative sunt concepute cu scopul de a optimiza munca în echipă prin accesarea unui mediu de lucru comun unde angajaţii pot împărtăşi cunoştinţe, date şi documente pentru a rezolva probleme specifice. Printre funcţionalităţile cel mai des puse la dispoziţie de aplicaţiile colaborative se numără: comunicaţia dintre utilizatori folosind canale de mesagerie instantă, managementul documentelor, împărţirea de sarcini, portabilitatea pe diferite tipuri de device-uri şi măsurile de securitate [1].

Lucrarea de faţă îşi propune crearea unei aplicaţii colaborative cu interfaţă web de tip „on premise”care să îndeplinească funcţionalităţi de comunicaţie şi de editare text în timp real, folosind doar servicii implementate intern. Spre deosebire de alte aplicaţii puse la dispoziţie pe piaţă, soluţia software propusă va oferi acces nelimitat despre modul de funcţionare, întrucât este implementată folosind resurse proprii, iar clientul va avea acces şi la codul sursă al acesteia. Marea majoritate a aplicaţiilor cu scop colaborativ oferite de dezvoltatori externi nu oferă servicii „on-premise”, datele clienţilor fiind administrate de dezvoltatorii în cauză.

## Sisteme similare

În momentul actual există o diversitate destul de mare de aplicaţii care oferă suport pentru munca colaborativă, cele mai multe dintre ele fiind implementate folosind Cloud extern. Acest aspect devine un criteriu important de selecţie atunci când organizaţia îşi doreşte o securitate ridicată şi lucrează cu informaţii clasificate care nu trebuie să părăsească mediul intern al organizaţiei. Printre aplicaţiile colaborative des întâlnite amintim: Microsoft Teams, Slack, Discord, Asana şi Google Docs.

### Microsoft Teams

Microsoft Teams este o aplicaţie colaborativă lansată la finalul anului 2016 care înglobează toate serviciile de comunicaţie (mesagerie instantă, apeluri video şi audio, gestionare de documente), oferind suport ca aplicaţie web, aplicaţie mobilă şi aplicaţie desktop. Acest utilitar se integrează cu suita de servicii Microsoft 365 şi alte servicii furnizate de Microsoft (spaţiul de stocare Cloud)[2].

Principalele avantaje ale aplicaţiei Microsoft Teams sunt portabilitatea între dispozitive, capabilitatea de a gestiona grupuri cu un număr ridicat de utilizatori şi diversitatea serviciilor puse la dispoziţie. Acest serviciu este destinat în principal companiilor sau organizaţiilor cu un număr ridicat de utilizatori, pornind de la abonamente gratuite, până la abonamente contra cost în funcţie de necesităţile şi activitatea organizaţiei. Popularitatea de care se bucură serviciul Teams este dată de suportul şi sustenabilitatea oferite de firma Microsoft, acesta ocupând marea parte a spaţiului educaţional şi antreprenorial

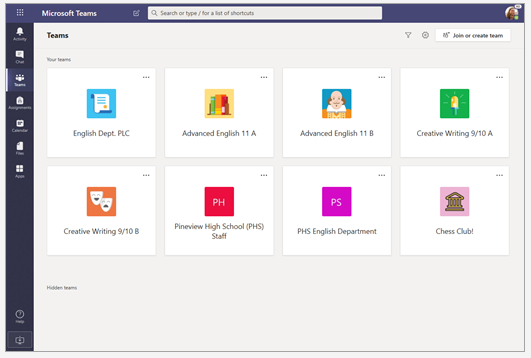


Fig. 1 - Microsoft Teams (Versiunea Desktop) – HomePage [3]

### Slack

Slack este un serviciu software de mesagerie instantanee lansat în anul 2013 şi conceput special pentru mediul de lucru din cadrul organizaţiilor. Potrivit deţinătorului companiei, numele aplicaţiei este un acronim pentru „Searchable Log of All Conversation and Knowledge”, astfel dezvăluind principala caracteristică a produsului, aceea ca toate conversaţiile şi fişierele partajate să poată fi căutate mai uşor faţă de alte dispozitive colaborative[4]. Un alt avantaj major al acestei platforme este că elimină nevoia conversaţiilor bazate pe email la nivelul organizaţiei.

Slack oferă versiuni ale aplicaţiei compatibile desktop şi mobile, dar şi o versiune de browser. Din punct de vedere al scalabilităţii, aplicaţia se remarcă prin faptul ca nu are o limită superioară al numărului de conversaţii deschise, faţă de competitorul său Teams care are o limită superioară de maxim 200 de canale la nivelul unei echipe.

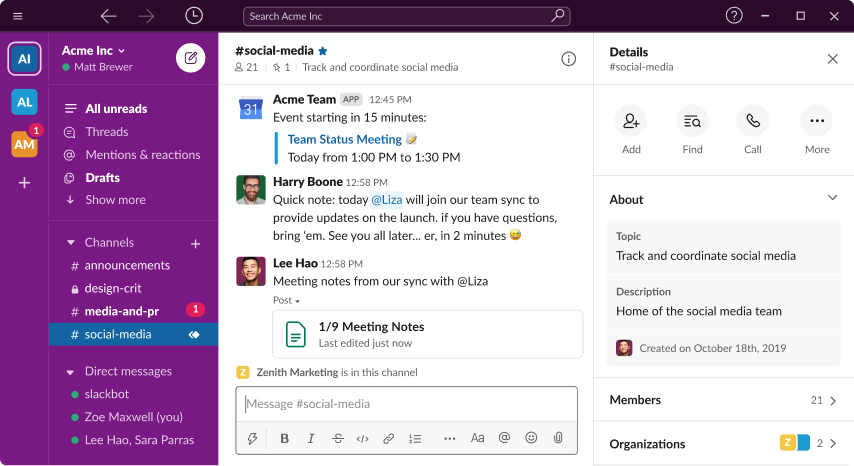


Fig. 2 – Slack (Versiunea Desktop) – HomePage [5]

### Discord

Discord este o aplicaţie cross-platform gratuită de mesagerie text, voce şi video folosită cel mai mult în scop recreativ. Popularitatea aplicaţiei este dată de faptul că este gratuită, iar limita maximă a utilizatorilor dintr-un grup poate ajunge până la 250.000, cu menţiunea că acest număr poate fi mărit. Limita de participanţi pentru un apel video este de 25 de utilizatori, iar pentru o conversaţie audio este de până la 122 de participanţi. Pentru asigurarea comunicaţiei, Discord, foloseşte tehnologia WebRTC oferită la nivelul browser-ului. Tehnologia WebRTC este disponibilă în toate browserele moderne. Pentru versiunile aplicaţiei care rulează pe alte dispozitive (desktop, mobile) este utilizat un modul media dezvoltat în C++ programat pe baza bibliotecii native WebRTC. Datorită acestui aspect unele caracteristici ale serviciului funcţionează mult mai bine pe aplicaţiile instalate decât în aplicaţia de tip browser.

Unul din dezavantajele aplicaţiei aplicaţiei sunt problemele de securitate şi incidentele petrecute în trecut precum cele de tipul phishing şi ransomware. De asemenea, aplicaţia nu eforă un serviciu de criptare end-to-end precum alte aplicaţii de chat.

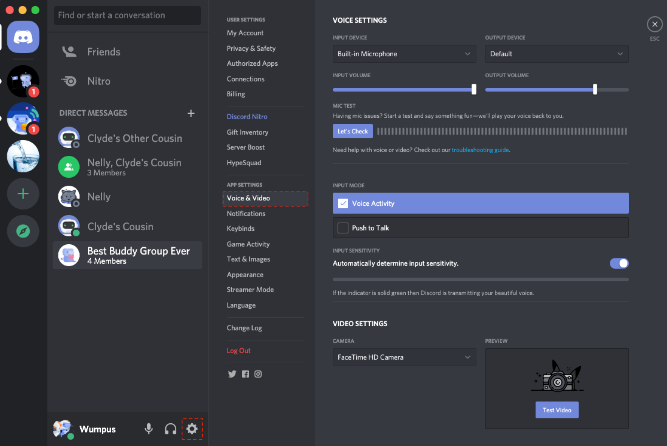


Fig. 3 – Discord (Versiunea Desktop) - Homepage[6]

### Telegram

Spre deosebire de celelalte aplicaţii prezentate până acum, Telegram este o aplicaţie colaborativă de tip open source ce prestează servicii de mesagerie text şi VoIP. Aplicaţia se remarcă prin numărul foarte mare de participanţi ce pot face parte dintr-un grup la un moment dat, în momentul de faţă acesta fiind de până la 100.000 de utilizatori. Ca şi particularităţi de interes tehnic mesajele transmise cu ajutorul aplicaţiei sunt transportate client-server criptat folosind criptare simetrică AES-256, după care sunt salvate la nivelul serverului pentru a putea fi descărcate oricând pe orice alt dispozitiv la care se conectează utilizatorul. Pentru purtarea de conversaţii sigure, aplicaţia pune la dispoziţie opţiunea de conversaţii „secrete”de tip text şi apeluri vocale criptate end-to-end doar între doi utilizatori online, dar nu şi pentru grupuri sau canale cu mai mult de doi utilizatori. Dacă utilizatorul optează pentru acest mod de conversaţie „secretă” atunci mesajele nerecepţionate nu pot fi retrimise destinatarului, iar aceste nu sunt nici salvate pentru back-up. Pe lângă aceste specificaţii aplicaţia atrage controverse prin folosirea unei scheme de criptare proprie derivată din schimbul de chei Diffie-Hellman, care a fost criticată de specialişti. Această schemă de criptare este descrisă pe site-ul celor de la Telegram, purtând numele de protocolul MTProto[7]. Acest protocol este în prezent la versiune 2.0, versiuna MTProto 1.0 fiind considerată depreciată de la sfârşitul anului 2017. Într-o lucrare publicată de Marino Miculan şi Nicola Vitacolonna este făcută o analiză teoretică asupra protocolului MTProto 2.0 şi rezultatul obţinut concluzionează că algoritmul este vulnerabil la anumite tipuri de atacuri MitM [8].

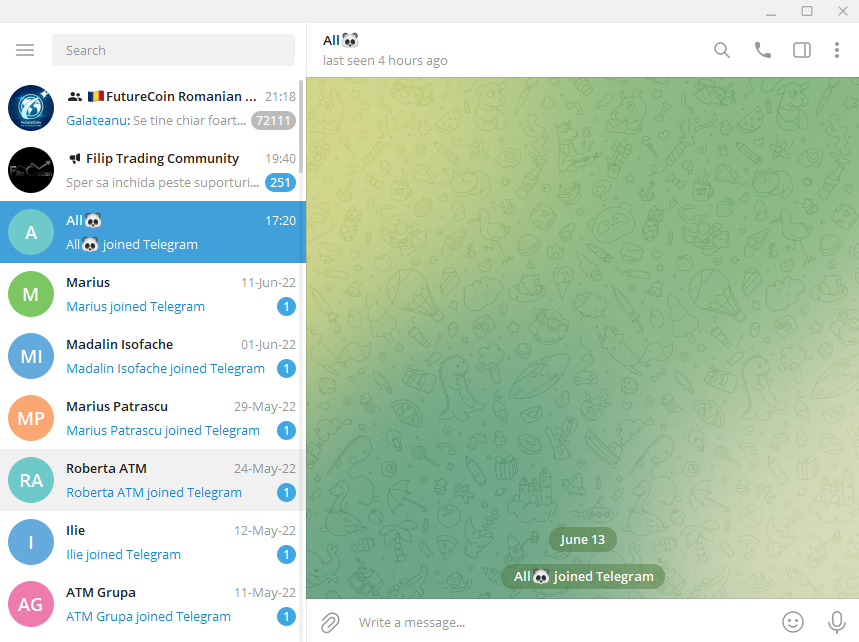


Fig. 4 – Telegram (Versiunea Desktop) Homepage

### Google Docs

Google Docs este face parte din suita de aplicaţii şi servicii puse la dispoziţie de către Google. Acest serviciu este accesibil prin intermediul browser-ului de internet ca aplicaţie web, dar este disponibil şi ca aplicaţie mobilă pentru Android, iOS sau desktop doar pe sistemul de operare Chrome OS dezvoltat de Google. Aplicaţia, Google Docs, este destinată pentru realizarea şi editarea de documente online de către mai mulţi utilizatori simultan. Modificările aduse de fiecare utilizator sunt salvate într-un istoric propriu folosit pentru revizuire. Poziţia cursorului specifică fiecărui utilizator este evidenţiată cu ajutorul unei culori, iar un sistem de permisiuni reglementează ceea ce pot face utilizatorii. În plus aplicaţia, dispune de o secţiune specială pentru distribuirea de sarcini către utilizatori, iar modificările aduse fişierelor sunt transmise serverului în mod automat.

Serviciile Google folosesc pentru autentificare şi transport un protocol criptografic numit ALTS, dezvoltat special de Google pentru securizarea servicilor sale, în detrimentul protocolului standard TLS. Ca principale avantaje ale folosirii protocolul ALTS se remarcă utilizarea protocolului de tip buffer (format de date cross-platform gratuit şi open-source destinat serializării datelor) pentru a serializa certificatele şi mesajele de protocol, în timp ce TLS utilizează certificate X.509 codificate cu ASN.1[9].

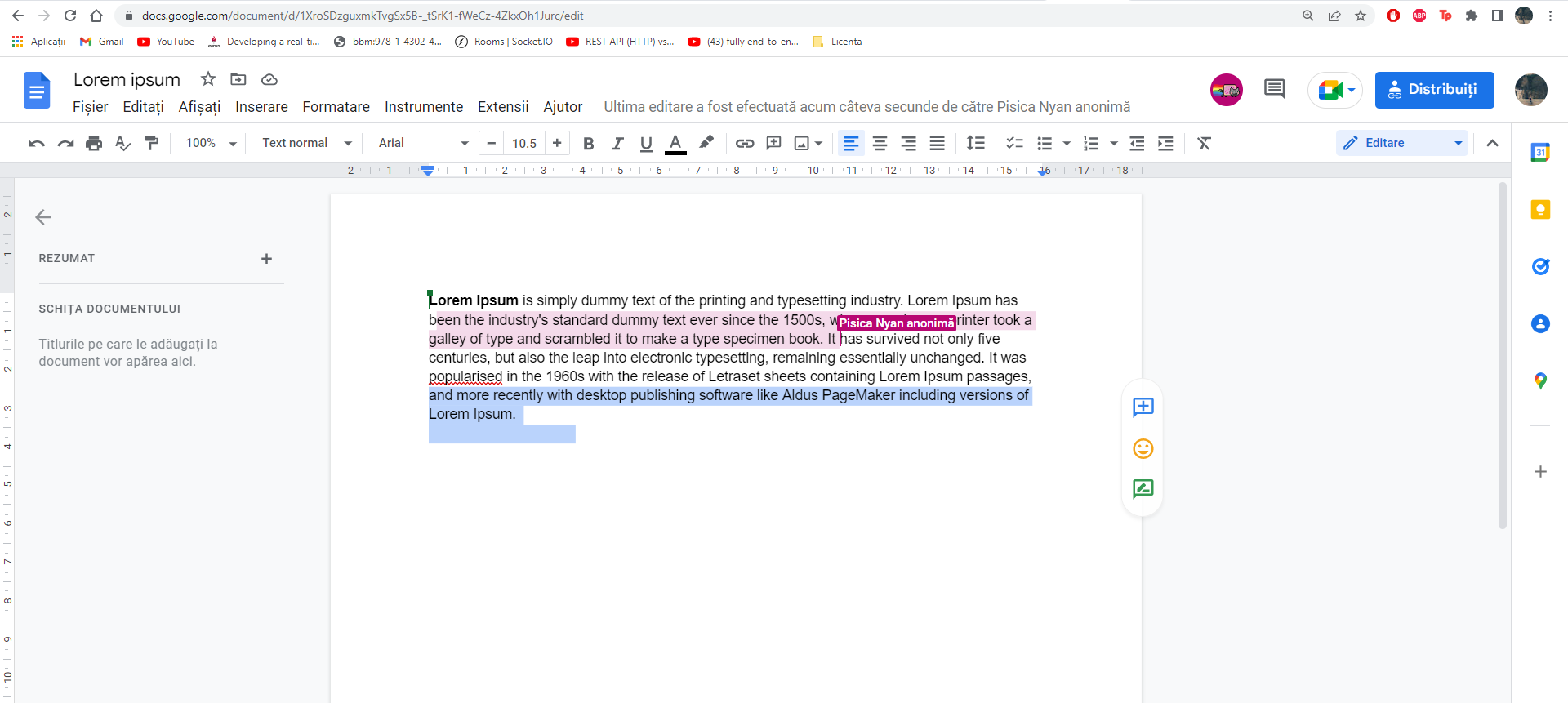


Fig. 5 - Google Docs - Editare colaborativă

## Principii de funcţionare

Aplicaţiile software colaborative sunt sisteme informatice care sprijină persoanele angajate în sarcini de interes comun prin oferirea de aplicaţii cu interfaţă partajată. Sistemele colaborative sunt instrumente utilizate pentru a facilita punerea în aplicare a muncii în grup. Aceste intrumente trebuie să respecte anumite criterii de funcţionare pentru a furniza utilizatorilor forme de interacţiune la un nivel optim, facilitând controlul, coordonarea şi colaborarea la nivelul echipei.

Potrivit articolului „Collaborative Systems: Characteristics and Feature” apărut în anul 2012[10], sistemele colaborative pot fi clasificate pe baza interacţiunii şi comunicaţiei cu utilizatorului. Prin urmare, instrumentele colaborative pot fi clasificate pe baza locului unde se desfăşoară interacţiunea (ambii utilizatori sunt prezenţi în acelaşi loc sau de la distanţă) şi pe baza timpului (dacă activitatea efectuată de utilizatori este sincronă sau asincronă):

* Sisteme sincrone. Sunt instrumente de colaborare care necesită un timp de răspuns cât mai scăzut precum serviciile de mesagerie sau teleconferinţă;
* Sisteme asincrone. Aceste instrumente nu necesită un timp de răspuns aproape instant. Exemple pentru această categorie de instrumente asincrone sunt: transmiterea de email-uri, calendarele de activităţi sau aplicaţiile de tip forum.

În continuare, voi descrie o parte din conceptele care stau la baza sitemelor colaborative din cadrul editării de text, partajării de flux video şi asigurarea confidenţialitate.

### Editarea colaborativă

Editorul de text colaborativ permite mai multor utilizatori să vizualizeze şi să editeze simultan un document partajat. Dezvoltarea unui editor colaborativ necesită luarea în considerare a următoarelor cerinţe de sistem:

* Timpul de răspuns trebuie să fie adecvat operaţiunilor de editare simultane;
* Asigurarea coerenţei textului în timpul actualizărilor simultane;
* Posibilitatea fiecărui utlizator de a se întoarce la o stare precedentă;
* Evidenţierea şi indicarea prezenţei celorlalţi utilizatori la nivelul documentului.

Principalul aspect ce trebuie luat în considerare la dezvoltarea unei aplicaţii colaborative bazate pe editare de text este că atunci când se doreşte editarea simultan de către mai mulţi utilizatori, apar probleme de control concurenţial. Pentru detalierea acestui caz voi reluarea exemplul descris în capitolul „Group Editors” din lucrarea „Computer Supported Co-operative Work” publicată de către profesorul Michel Beaudouin-Lafon.

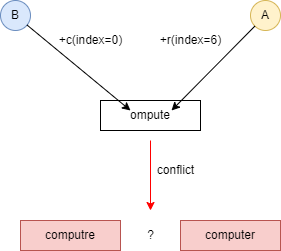


Fig. 6 - Creare de conflict prin accesare concurenţială

Un document conţine un şir de caractere „ompute”. Să presupunem că utilizatorul A încearcă să insereze caracterul „r” după caracterul „e”. Majoritatea editoarelor text folosesc funcţii de inserare care utilizează un parametru pentru indexarea poziţiei din document, mai exact o funcţie care va adăuga litera „r” la poziţia 6. Dar să presupunem că între momentul în care operaţia este generată de datele de intrare ale lui A şi momentul în care este executată, un alt utilizator, B, încearcă să adauge litera „c” înaintea literei „o”. Dacă este executată operaţia lui B înaintea operaţiei lui A, şirul afişat ar fi „computre”, în loc de rezultatul dorit, „computer”[11].

Autorul lucrării propune diferite scheme de control al accesului pentru rezolvarea conflictelor printre care: controlul concurenţial pesimist, controlul concurenţial pozitiv sau posibilitatea ca modificările participanţilor să comute între ele. În cazul pesimist restricţionează accesul utilizatorilor la editare până ce documentul este reactualizat pentru fiecare utilizator, acest algoritm fiind nefezabil din cauza generării timpilor mari de aşteptare ca rezultat al latenţei reţelei. Pentru accesul concurenţial pozitiv, o operație este executată imediat pe copia locală și apoi este transmisă spre restul utilizatorilor pentru a fi executată. Toate operațiile de actualizare sunt mai întâi marcate în timp, astfel încât două operații oarecare să fie consecvent reordonate în toate copiile, chiar dacă sunt primite în aceeaşi ordine sau nu. Pentru reordonare, fiecare editor trebuie să mențină o listă de istoric a operaţiilor ce au fost făcute asupra documentului. În continuare urmează descrierea a doi algoritmi dezvoltaţi pentru rezolvarea controlului concurenţial la nivelul editorului.

### OT

OT (Operational Transformation) este o tehnică de menținere a consistenței pentru sistemele de editare colaborativă folosită de aplicații distribuite pentru sprijinirea interacțiunii om-calculator și a colaborării prin rețele de comunicații. Teoria cauzalității a stat la baza tuturor sistemelor OT anterioare, dar este inadecvată pentru a satisface cerințele esențiale ale OT în funcționalitate și corectitudine. Capacitățile sale au fost extinse, iar aplicabilităţile sale au fost extinse pentru a include anularea în grup, blocarea, rezolvarea conflictelor, notificarea și comprimarea operațiilor, conștientizarea grupului, partajarea aplicațiilor și instrumente de proiectare media asistată de calculator în colaborare[12]. Această tehnologie este folosită de către editorul Google Docs pentru rezolvarea conflictelor.

Sistemele de colaborare care utilizează OT folosesc de obicei stocarea replicată a documentelor, astfle fiecare aplicaţie client are propria copie a documentului. Modificările de text sunt făcute de utilizatori pe copiile lor locale într-o manieră cursivă, fără blocaje, iar modificările sunt apoi propagate la restul clienților. Acest mecanism asigură o capacitate de reacție ridicată a clientului în medii cu latență ridicată, precum internetul. Atunci când un client primește modificările propagate de la un alt client, acesta transformă modificările în funcţie de copia sa locală, înainte de a le executa. Transformarea asigură menținerea criteriilor de coerență dependente de aplicație de către toate site-urile. Acest mod de funcționare are ca rezultat un sistem potrivit pentru implementarea funcțiilor de colaborare, cum ar fi editarea colaborativă a documentelor.

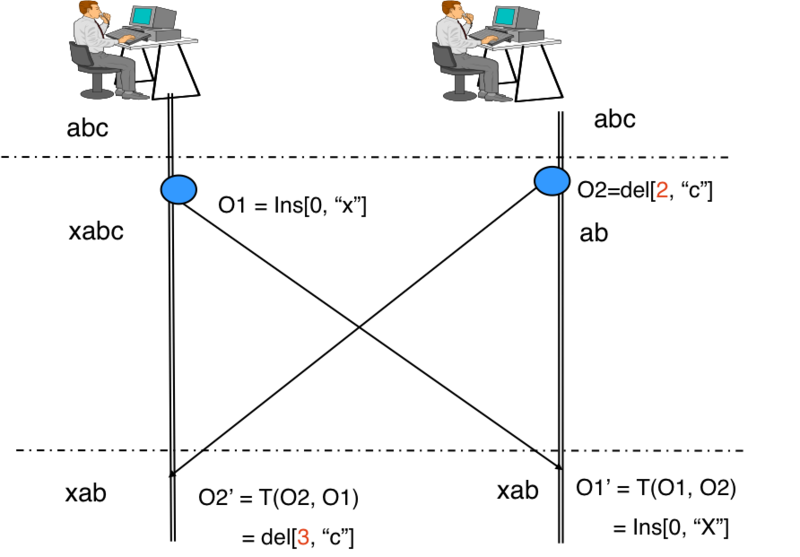


Fig. 7 - Mecanism Transformare Operaţională[12]

Mecanismul de funcţionare al transformării operaţionale este reprezentat în figura de mai sus folosind un scenariu simplu: fiecare utilizator este conectat la sesiunea de editare ce conţione textul „abc”. Primul utilizator introduce caracterul „x” la poziţia 0 a şirului, definind astfel operaţia O1 = insert[0,’x’]. Cel de al doilea utilizator doreşte să şteargă caracterul „c” de la poziţia 2, definind operaţia O2 = insert[2,’c’].

Să pleacă de la premiza că cele două operații sunt executate în ordinea O1 și O2, pentru primul utilizator. După executarea lui O1, textul formatat devine "xabc". Pentru a putea fi executată O2 după ce a fost executată O1, mai întâi trebuie ca O2 să fie transformată față de O1 pentru a deveni: O2' = Delete[3, ‘c’], al cărui parametru pozițional este incrementat cu o unitate datorită inserării caracterului „x” de către operaţia O1. În final, după O2' pe „xabc” se șterge caracterul corect „c” și documentul devine „xab”. Cu toate acestea, dacă O2 este executată fără transformare, aceasta șterge în mod incorect caracterul „b” în loc de caracterul „c”. Ideea de bază a transformării operaţionale este transformarea parametrilor operaților de editare în funcție de efectele operațiilor concurente executate anterior, astfel încât operația transformată să poată obține efectul corect și să fie menținută coerența documentului.

### CRDT

În calcul distribuit, CRDT (Conflict-free replicated data type) este o structură de date care poate fi replicată pentru mai mulţi utilizatori, replicile putând fi actualizate independent și simultan fără coordonare între ele, iar inconsecvenţele care pot apărea pot fi mereu rezolvate din punct de vedere matematic. Conceptul CRDT a fost definit în mod oficial în 2011, iar dezvoltarea a fost motivată inițial de editarea colaborativă de text și de informatica mobilă. Sistemele CRDT au fost utilizate, de asemenea, în sistemele de mesagerie online şi în jocurile de noroc online[13] Există două abordări pentru sisteme CRDT, ambele putând asigura o coerență puternică eventuală: sisteme CRDT bazate pe operații şi sisteme CRDT bazate pe stări. Cele două alternative sunt echivalente din punct de vedere teoretic, deoarece una o poate emula pe cealaltă Cu toate acestea, există diferențe practice.

Sistemele CRDT bazate pe stare sunt adesea mai simplu de proiectat și de implementat; singura lor cerință de la substratul de comunicaţie fiind mesajele despre starea celorlalte sisteme replicate. Dezavantajul lor este că întreaga stare a fiecărei CRDT trebuie transmisă în cele din urmă fiecărei alte replici, ceea ce poate fi costisitor. În schimb, sistemele CRDT bazate pe operații transmit doar operațiile de actualizare, care sunt de obicei mici. Cu toate acestea, sistemele CRDT bazate pe operații necesită garanții din partea protocolului de transmisie că operațiile nu sunt abandonate sau duplicate atunci când sunt transmise către celelalte replici și că sunt livrate în ordine cauzală[13].

Sistemele CRDT bazate pe operații sunt, de asemenea, numite tipuri de date replicate comutative. Replicile CRDT de tip comutativ propagă starea prin transmiterea doar a operației de actualizare. Cu toate acestea, ele nu sunt neapărat neschimbate. Prin urmare, infrastructura de comunicații trebuie să se asigure că toate operațiile care aparţin unei replici sunt transmise celorlalte replici, fără duplicate, dar în orice ordine. Sistemele CRDT bazate pe stare se numesc tipuri de date replicate convergente. Acestea, spre deosebire de celelalte, trimit starea lor locală completă către alte replici, unde stările sunt fuzionate utilizând o funcție care trebuie să fie comutativă, asociativă și imutabilă.

### WebRTC

Web Real-Time Communication (WebRTC) este o colecţie de standarde, protocoale şi API-uri JavaScript care face posibilă partajarea de date audio şi video între browsere, în mod peer-to-peer. Semantica clasică a arhitecturii web se bazează pe o paradigmă client-server, în care browserele trimit o cerere de conținut HTTP către serverul web, care răspunde cu un răspuns conținând informațiile solicitate.

WebRTC extinde semantica client-server prin introducerea unei paradigme de comunicare peer-to-peer între browsere. În cazul aplicaţiilor colaborative acest standard este foarte popular datorită modului facil de implementare al canalului de comunicaţie. Înainte de stabilirea conexiunii dintre cele două browsere implicate în conexiune, este necesar trimiterea de mesaje intermediare pentru schimbul de adrese ale clienţilor. Aceste mesaje se mai numesc şi mesaje de semnalizare. Acestea sunt transportate prin protocolul HTTP sau WebSocket prin intermediul serverelor web care le pot modifica, traduce sau gestiona în funcție de necesități. Este de remarcat faptul că semnalizarea dintre browser și server nu este standardizată în WebRTC, deoarece este considerată ca făcând parte din aplicație. În ceea ce privește calea de transmitere a datelor, conexiunea peer-to-peer permite ca mediile să circule direct între browsere, fără intervenția serverului, reprezentare în Fig. ????.

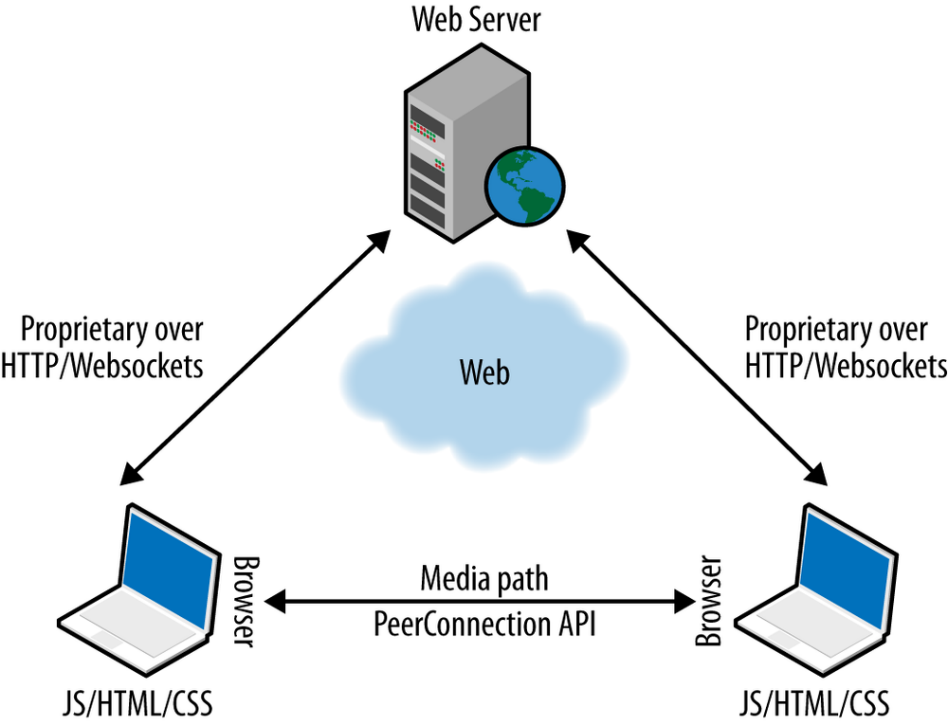


Fig. 8 - WebRTC - Transmisia datelor[14]

Odată stabilită o conexiune între omologi, fluxurile media (asociate local cu obiecte MediaStream definite ad-hoc) pot fi trimise direct către browserul de la distanță. Conxiunea peer-to-perr utilizează protocolul ICE împreună cu serverele STUN și TURN pentru a permite fluxurilor media bazate pe UDP să traverseze sistemele NAT și sistemele firewall.Protocolul ICE permite browserelor să descopere suficiente informații despre topologia rețelei în care sunt implementate pentru a găsi cea mai bună cale de comunicare exploatabilă. Utilizarea ICE oferă, de asemenea, o măsură de securitate, deoarece împiedică paginile web și aplicațiile care nu sunt de încredere să trimită date către gazde care nu se așteaptă să le primească.

Protocolul STUN (definit în RFC5389) permite unei aplicații gazdă să descopere prezența unui traductor de adrese de rețea și, în acest caz, să obțină combinaţiile de adrese IP și porturi publice alocate pentru conexiune. Pentru a face acest lucru, protocolul necesită asistență din partea unui server STUN configurat, de terță parte, care trebuie să se afle în rețeaua publică.

Protocolul TURN (definit în RFC5766) permite unei gazde aflate dincolo de NAT să obțină o adresă IP publică și un port de la un server releu care se află pe internetul public. Datorită adresei de transport retransmise, gazda poate primi apoi medii de la orice omolog care poate trimite pachete către internetul public.

Fluxul media de date efectuat la nivelul standardului WebRTC sunt criptate cu ajutorul protocolului DTLS (definit în RFC6347), acesta fiind conceput pentru a preveni interceptarea, modificarea sau falsificarea mesajelor în transportul de datagrame oferit de protocolul UDP şi având la bază protocolul TLS cu specificaţii de scuritate asemănătoare.

### Signal Protocol

Criptarea E2EE este un tip de criptografie asimetrică, care protejează datele, făcându-le disponibile doar destinatarului. La momentul actual, E2EE este considerat cel mai sigur mecanism de protecţie a datelor, întrucât participanţii sunt direct implicaţi în schimbul de mesaje, ofuscarea datelor realizându-se doar la nivelul lor fără intervenţia serverului sau al altor entităţi. În acest procedeu, serverul are scopul doar de a realiza schimbul de chei publice dintre participanţi fără să aibă posibilitatea de decriptare a mesajelor.

Protocolul Signal este un protocol dezvoltat de Open Whisper Systems în 2013 prin intermediul căruia se poate realiza o comunicaţie securizată E2EE. Etapele protocolul Signal sunt următoarele:

* Schimbul inițial de chei, sau protocolul X3DH (extended triple Diffie-Hellman), care combină chei Diffie-Hellman cu nivel de persistenţă în timp ridicat, mediu și scurt pentru a genera cheia de bază partajată;
* O etapă asimetrică de tip „ratchet”(termen care în traducere înseamnă clichet, în literatura de specialitate fiind folosit pentru a exprima propritatea sistemului de a funcţiona într-un singur sens, fără posibilitate de inversare a operaţilor), în care utilizatorii trimit alternativ noi chei Diffie-Hellman efemere pe baza cheilor private de bază generate anterior pentru a genera o serie de chei de sesiune;
* O etapă simetrică cu clichet de tip „ratchet”, în care utilizatorii nu iau nicio entropie suplimentară, ci folosesc în schimb funcţii de derivare pentru a genera chei de criptare simetrice.

Mecanismele de tip „ratchet” generează o nouă cheie private pe baza celei precedente, oferind sistemului confidenţialitate şi în cazul în care una din cheile private de bază este compromisă. Astfel, un posibil atacator nu va reuşi să decripteze decât o serie minimă de mesaje[15].

# API-uri folosite

Pentru realizarea proiectului s-a folosit limbajul de programare orientat obiect JavaScript însoţit de bibliotecile aferente acestuia. Pentru aplicaţia server a fost utilizat NodeJS, iar pentru realizarea aplicaţiei client ReactJS. Pentru persistenţa datelor am ales folosirea unei baze de date MySQL de tip relaţional, din necesitatea de a reţine relaţiile de interacţiune dintre utilizatori sub forma unei reprezentări grafice bine definite. Limbajul de programare JavaScript facilitează folosirea formatului de date JSON, un format inteligibil pentru oameni şi folosit pentru transmiterea datelor structurate prin reţea.

## Aplicaţia Server

NodeJS este un mediu de execuţie JavaScript asincron bazat pe evenimente, open-source şi cross-platform. Spre deosebire de cele mai multe alte medii moderne, un proces Node nu se bazează pe multithreading. pentru a susține execuția concurenţială a proceselor, ci se bazează pe un model asincron de desfășurare a evenimentelor Input/Output (vezi Fig. ????). Limbajul JavaScript se potrivește excelent pentru această abordare pentru că suportă returnările de evenimente. Natura funcțională a JavaScript îl face extrem de facil pentru crearea obiectelor funcționale anonime ce pot fi înregistrate ca gestionări de evenimente. Printre altele, NodeJS, în comparaţie cu alte medii de execuţie, prezintă şi capabilitatea de a gestiona un număr foarte mare de conexiuni simultane cu un debit ridicat de date, ceea ce echivalează cu o scalabilitate ridicată[16].

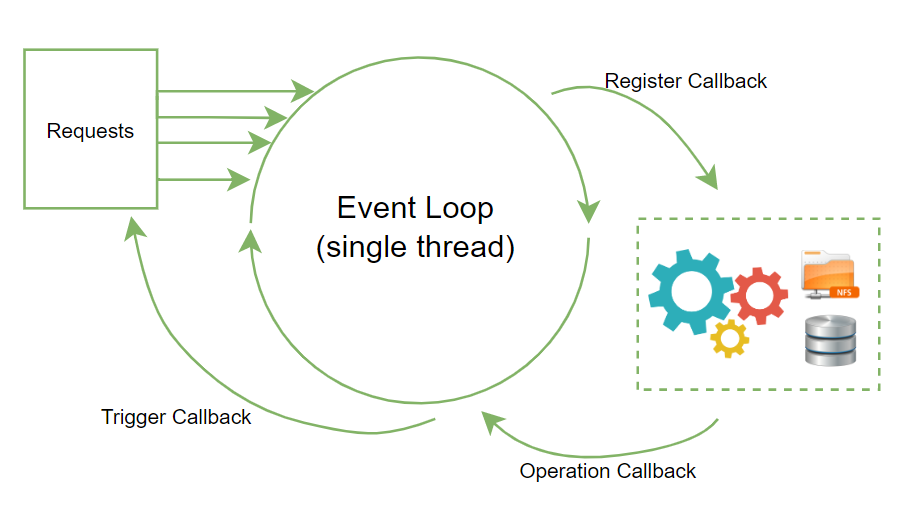


Fig. 9 - Modelul asincron de tratare al evenimentelor - NodeJS

Express.js este un framework pentru modul de execuţie NodeJS utilizat pentru reducerea timpului de dezvoltare al aplicaţiilor şi organizarea arhitecturii aplicaţiei după modelul MVC. În cadrul proiectului, am folosit acest framework, deoarece oferă un mecanism avansat de rutare ce ajută la păstrarea stării paginii web cu ajutorul URL-ului lor şi de asemenea pune la dispoziţie mai multe caracteristici utilizate în mod obişnuit de NodeJS sub formă de „middleware” care pot fi apelate oriunde la nivelul programului[17]. Principalel componentele „middleware” folosite pentru realizarea aplicaţiei server sunt următoarele:

* *body-parser* – analizează corpul request-ului înainte de prelucrarea acestuia;
* *cookie-parser* – parsează antetul Cookie din request sub formă de perechi cheie-valoare;
* *cors* – folosit pentru activarea politicilor CORS cu diferite opţiuni;
* *dotenv* – încarcă variabilele de mediu, globale, din fişierul „.env” în „process.env”;
* *date-and-time* – colecţie minimalistă de funcţii pentru prelucrarea datei şi orei;
* *multer* – permite manipularea „multipart/form-data” cu scopul încărcării şi salvării de fişiere;
* *mysql* – folosit pentru interogarea bazei de date MySQL;
* *socket.io* – permite comunicarea bidirecţională în timp real bazată pe evenimente folosind protocolul WebSocket;
* *rand-token* – genereză token-uri aleatoare pe baza input-ului furnizat de utilizator;
* *moment* – folosit pentru analizarea, validarea şi formatarea formatului de timp;
* *uuid* – furnizeayă id-uri aleatoare folosite la identificarea obiectelor;
* *crypto-js* – furnizează o colecţie de algoritmi criptografici folosiţi pentru aigurarea confidenţialităţii la nivelul aplicaţiei;
* *fs* – permite creare de foldere şi fişiere atât în mod sincron, cât şi asincron;
* *node* – oferă utilitate pentru lucrul cu căile de acces la fişiere şi directoare.

## Aplicaţia Client

Pentru implementarea aplicaţiei client, am ales biblioteca ReactJS, datorită popularităţii sale la momentul actual. Această bibliotecă apartanentă JavaScript este utilizată pentru dezvoltarea de interfeţe interactive cu utilizatorul şi se remarcă prin dezvoltarea componentelor modulare care pot fi reutilizate, încorporând modelul de proiectare MVC. Alte componente API notabile folosite pentru realizarea aplicaţiei client sunt:

* *React Hooks* – este o caracteristică introdusă în versiunea React 16.8. Acestea permit utilizarea stării componentelor specifice React fără folosirea unei clase (de menţionat că acestea nu funcţionează în interiorul unei clase). Denumirea de „Hooks” denotă caracteristica acestora de a se „agăţa” de starea şi de caracteristicile ciclului de viaţă din componentele funcţionale[18];
* *bootstrap* – framework de tip front-end ce cuprinde elemente de design HTML şi CSS predefinite pentru dezvoltarea interfeţelor web;
* *axios* – modul specific mediului NodeJS, ce facilitează trimiterea de request-uri HTTP asincrone către server cu scopul efectuării de operaţii CRUD;
* *crypto-browserify* – modul ce asigură funcţiile modulului *crypto-js*, dar pentru utilizarea în browser;
* *font-awesome* – este un set de pictograme şi fonturi bazat pe CSS şi LESS;
* *simple-peer* – bibliotecă utilizată pentru crearea de conexiuni de tip peer-to-peer între aplicaţiile de tip browser. De menţionat că nu include funcţionalitate de semnalizare print intermediul serverului;
* *socket.io-client* – permite conectarea clientului de la nivelul browser-ului la conexiunea bazată pe protocolul WebSocket;
* *quill* – este un modul specific JavaScript ce implementează un editor de text cross-platform; acesta suportă un set extins de operaţii dedicate formatării de text;
* *quill-cursors* – este un modul *quill* dedicat pentru afişarea cursorilor altor utilizatori pentru asigurarea unei experienţe interactive de editare colaborativă;
* *MediaDevices.getUserMedia()* – solicită utilizatorului permisiunea de a utiliza intrările media (cameră, microfon sau serviciu de partajare ecran) care produc un flux media cu outputul generat de acestea.

ReactJS s-a remarcat ca fiind un sistem eficient şi performant datorită conceputului de DOM virtual pe care îl foloseşte. Cu alte cuvinete, menține un model de DOM virtualizat în interiorul memoriei, iar atunci când o modificare trebuie să fie adusă paginii web, în loc să se actualizeze instantaneu sistemul DOM de navigare, sunt efectuate mai întâi modificări la nivelul sistemului de DOM virtual din cadrul ReactJS. După realizarea modificărilor la nivelul de DOM virtual, se aplică un algoritm diferenţial care compară modelul virtual şi modelul de navigare, iar apoi actualizează doar nodurile relevante și dorite din arborelui DOM din browser. Fluxul datelor la nivelul ReactJS este permis de sus în jos, unidirecţional şi susţinut. În cazul în care o anumită modificare urmează să fie făcută asupra datelor din componentele superioare, componentele care utilizează acele date se vor reda automat pentru a actualizare[19].

## SocketIo

WebSocket este un protocol de comunicații care oferă canale de comunicare full-duplex printr-o singură conexiune TCP. WebSocket este distinct de HTTP. Ambele protocoale se află la nivelul 7 în modelul OSI și depind de TCP la nivelul 4. Deși sunt diferite, RFC 6455 afirmă că WebSocket „este conceput pentru a funcționa pe porturile HTTP 443 și 80, precum și pentru a suporta proxy-uri și intermediari HTTP”, făcându-l astfel compatibil cu HTTP. Pentru a obține compatibilitatea, handshake-ul WebSocket utilizează antetul HTTP Upgrade pentru a trece de la protocolul HTTP la protocolul WebSocket.

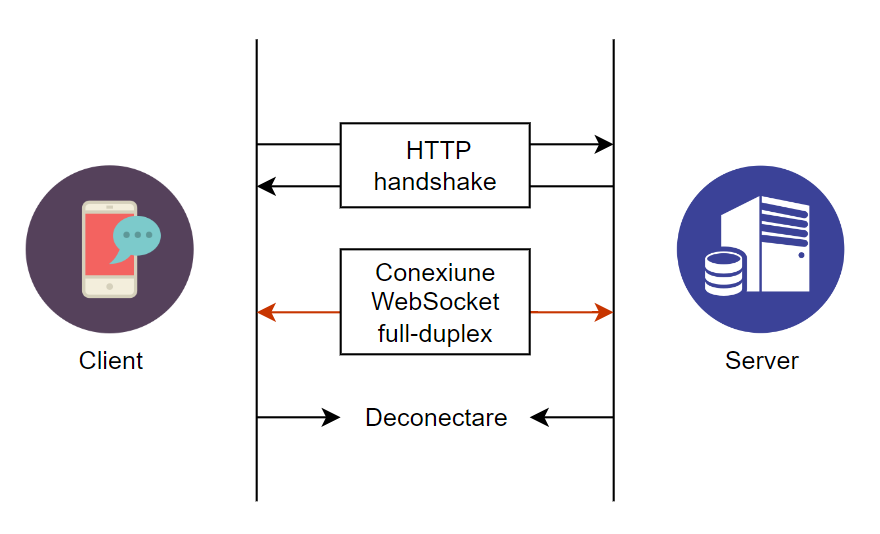


Fig. 10 - Iniţierea Protocolului WebSocket

Socket.IO este o bibliotecă care permite comunicaţia, bidirecțională, fără întârzieri și bazată pe evenimente între Client și Server. Necesitatea folosirii acestei biblioteci în cadrul proiectului este dată de nevoia transmisiei de mesaje cu latenţă cât mai scăzută. Acesta este construit pe baza protocolului WebSocket și oferă garanții suplimentare de transmitere a datelor, cum ar fi revenirea la o interogare de tip HTTP sau reconectarea automată[20]. Socket-ul folosit folosit pentru realizarea conexiunii emulează un socket de reţea prin diferite mecanisme de transport şi are diferite etape în ciclul său de funcţionare care sunt următoarele:

* *connecting*;
* *connected*;
* *disconnecting*;
* *disconnected*;

Socket-ul se stabileşte doar după ce clientul trimite o cerere de conectare către server și se inițiază un handshake. După ce handshake-ul este finalizat, se deschide o conexiune folosind protocolul de transport negociat, iar starea socket-ului este setată ca *connected*. Pentru a verifica dacă socketul este conectat serverul poate solicita trimiterea de mesaje *heartbeat* de la client la server în intervale regulate. În absența unui astfel de mesaj sau în cazul deconectării layer-ului transport, socket-ul va fi deconectat. În acest caz, clientul va iniția o reconectare, iar în cazul în care conexiunea este restabilită în intervalul de timp aşteptat, conexiunea se va restabili, iar mesajele stocate în memorie vor fi trimise. În cazul în care conexiunea nu este restabilită, clientul va iniția o nouă cerere de conexiune, prin iniţierea unui nou handshake. Socket-ul este închis atunci când metoda *close()* este apelată fie de client, fie de server. Odată ce conexiunea de transport este stabilită, schimbul de mesaje dintre client şi server se realizează doar prin intermediul socket-ului.



Fig. 11 – Parametrii HandShake[20]

Parametrii stabiliţi la handshake (Fig. ???) reprezintă:

* sid este id-ul sesiunii, care trebuie inclus în parametrul de interogare sid în toate cererile HTTP ulterioare;
* upgrades conţine lista protocoalelor de transport stabilite cu serverul;
* valorile pingInterval și pingTimeout sunt utilizate în mecanismul heartbeat.

## Quill

Quill este un editor de text modern şi cross-platform construit pentru compatibilitate și extensibilitate. Acest modul API oferă suport şi pentru aplicaţile web, reprezentând un punct de plecare foarte important pentru construirea funcţionalităţii de editare colaborative. Conţinutul editorului este reprezentat în format JSON, fiind uşor de extras şi de prelucrat. Modificările aduse de către utilizator sunt extrase sub formă de obiecte cu denumirea de *delta*[21].

Obiectele de tip *delta* sunt reprezentate într-un format simplu, dar expresiv, care poate fi utilizat pentru a descrie conținutul și modificările din Quill. Formatul este un subansamblu strict de tip JSON, lizibil pentru oameni și ușor de analizat de către program. Obiectele *delta* pot descrie orice document generat de Quill, include toate informațiile de text și formatare, fără ambiguitatea și complexitatea formatului HTML. Funcţionalităţile editorului folosite în proiect sunt:

* *getContents()* - preia conținutul editorului, cu date de formatare, reprezentate de un obiect *delta*;
* *getLength() –* returnează lungimea conţinutului editorului;
* *getModule()* – returnează un modul care a fost adăugat editorului, în contextul actual este folosit pentru adăugarea modulului de afişare a cursorului;
* *updateContents()* – actualizeză conţinutul editorului pe baza obiectului *delta*;

De asemenea, modulul Quill pune la dispoziţie un mecanism de detectarea a evenimentelor generate atunci când acesta îşi schimbă starea. Aceste evenimente pot fi tratate prin intermediul funcţiei „quill.on(event, function(delta, oldDelta, source)”. Evenimentele folosite în cadrul aplicaţiei sunt:

* *text-change* – se emite atunci când conținutul Quill s-a schimbat. Sunt furnizate detalii privind modificarea, reprezentarea conținutului editorului înainte de modificare, precum și sursa modificării. Sursa va fi „user” dacă provine de la utilizatori;
* *selection-change* – se emite atunci când un utilizator sau API determină modificarea selecției, cu un interval care reprezintă limitele selecției. Un interval *null* indică pierderea selecției (cauzată, de obicei, de pierderea focalizării de la editor).

# Structura proiectului

## Cerinţele sistemului

Aplicaţia propusă are scopul de a asigura munca de tip colaborativ dintre membrii unei organizaţii care lucrează cu documente clasificate. Din necesitatea de a-şi asigura confidenţialitatea, organizaţia nu poate utiliza servicii software existente care funcţionează pe sisteme Cloud externe şi este nevoie de o soluţie software „on-premises”. Pentru a răspunde acestui caz de utilizare am proiectat şi dezvoltat o aplicaţie de tip colaborativ, web based, care va răspunde setului de cerinţe definite mai jos. O aplicaţie de tip web based, răspunde cerinţelor de mobilitate impuse de activitatea membrilor organizaţiei.

### Cerinţe Non-Funcţionale

Soluţia software propusă va fi formată din două componente: aplicaţia client, web based, ce poate fi accesată prin intermediul unui browser-ului şi aplicaţia server.

**Ergonomie** – aplicaţia client trebuie să pună la dispozişie o interfaţă intuitivă prin care utilizatorul poate accesa uşor funcţionalităţile puse la dispoziţie.

**Confidenţialitate** – datele utilizatorului sunt protejate prin anonimizarea accesului (datele personale ale utilizatorului sunt modificate).

**Securitate** – canalul de comunicaţie va fi securizat prin criptarea mesajelor fişierelor şi a fluxului video. Accesul la aplicaţie se va face prin filtrarea adreselor IP legitime, iar în cazul încercărilor de acces neautorizat se vor produce alerte. Se va asigura un mecanism de protecţie împotriva atacurilor de tip Cross-Site Request Forgery prin folosirea token-urilor de acces JWT.

**Performanţă** – aplicaţia va suporta utilizarea simultană de către mai mulţi utilizatori cu timpi de răspuns scăzuţi pentru a nu afecta performanţa serviciului şi cu experienţa utilizatorul.

**Portabilitate** – funcţionalităţile asigurate de aplicaţie vor putea fi accesate din browser folosind orice dispozitiv de tip desktop conectat la internet.

**Flexibilitate** – toate datele vor fi salvate în baza de date şi vor putea fi accesate la fiecare nouă sesiune a utilizatorului.

### Cerinţe Funcţionale

Aplicaţia propusă spre dezvoltare va trebui să respecte următoarele cerinţe pentru îndeplinirea scopului final:

**SRS[[1]](#footnote-1) 1** – Soluţia propusă este capabilă să fie accesată prin intermediul browser-elor Google Chrome, Mozilla Firefox şi Microsoft Edge folosind sisteme hardware de tip desktop.

**SRS 2** – Accesul în aplicaţie se va face folosind autentificare mutuală pe bază de certificate, respectiv credenţiale (email şi parolă). Fiecare utilizator beneficiază de un cont de utilizator creat de administrator şi nu este disponibilă opţiunea de register din motive de securitate.

**SRS 3** – Căutarea de utilizatori pentru începerea conversaţiilor de tip privat sau de conversaţii deja existente, rezultatele obţinute în urma căutării fiind ordonate alfabetic în funcţie de subşirul de căutare.

**SRS 4** – Asigurarea comunicaţiei bidirecţionale la nivel de mesaje text între doi utilizatori folosind canale de comunicaţie private şi securizate prin criptare de tip EE2E.

**SRS 5** – Crearea unui mediu comun de stocare al fişierelor sub formă de folder pentru participanţii unei conversaţii de tip privat. Acest *folder* va fi reprezentat utilizatorilor purtând numele interlocutorului.

**SRS 6** – Crearea de grupuri cu mai mult de doi participanţi. La crearea unui grup nou, va fi creat şi un mediu de stocarea care va fi partajat cu fiecare nou utilizator adăugat.

**SRS 7** – Opţiune de adăugare de noi participanţi la grupurile publice. Aceştia vor avea acces la spaţiul de stocare specific grupului.

**SRS 8** – Trimiterea de fişiere la nivelul convesaţiei sub formă de mesaj şi posibilitatea de a vizualiza detalii despre acestea la nivelul conversaţiei. Fişierele respective vor fi afişate şi în spaţiul comun de stocare, însoţite de detalile aferente (denumire, dată creare, autor).

**SRS 9** – La nivelul spațiului de stocare va exista un sistem de navigare rapidă între folderele utilizatorului curent, format din două componente: o bară de navigare și o reprezentare arborescentă a folderelor și fișierelor salvate.

**SRS 10** – Fişierele partajate vor putea fi descărcate, atât de la nivelul conversaţiei, cât şi din spaţiul de stocare comun de la nivelul sistemului de fişiere.

**SRS 11** – Utilizatorii pot iniţa apeluri video-audio prin accesarea unei pagini dedicate din canalul de mesagerie. Pentru conectarea utilizatorilor la un apel video-audio este suficient accesarea adresei URL din browser care poate fi partajată şi cu alţi utilizatori care nu aparţin acelui grup sau conversaţii.

**SRS 12** – Utilizatorii pot încheia convorbirea prin apăsarea unui buton dedicat.

**SRS 13** – Utilizatorii pot opri camera, respectiv microfonul în timpul convorbirii folosind butoane intuitive.

**SRS 14** – Funcţionalitate de creare fişiere text şi de editare simultană de către doi sau mai mulţi utilizatori.

**SRS 15** – Conectarea utilizatorilor la fereastra de editare colaborativă se va face prin accesarea adresei URL a ferestrei prin intermediul browser-ului.

**SRS 16** – Documentul text generat de editor va fi salvat în spaţiul de stocare unde a fost făcută cererea de generare a acestuia (dacă este un spaţiu de stocare comun, toţi utilizatorii autorizaţi vor putea să-l acceseze).

**SRS 17** – Utilizatorul va avea posibilitatea de a-şi edita datele personale, cum ar fi numele, adresa de email sau parola folosită la autentificare.

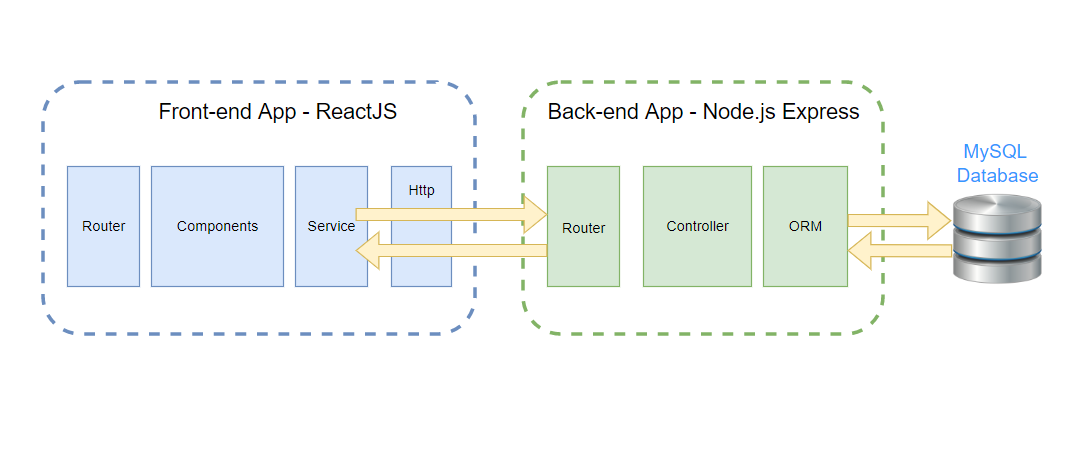
**SRS 18** – Utilizatorul va putea să-şi schimbe poza de profil cu una personalizată.

**SRS 19** – Grupurile vor dispune de poză de profil, iar mebrii din cadrul acestora vor avea posibilitea de o personaliza.

**SRS 20** – Interacțiunea utilizatorilor la nivelul documentului text prin crearea de cursoare și pointeri personalizați.

## Arhitectura sistemului

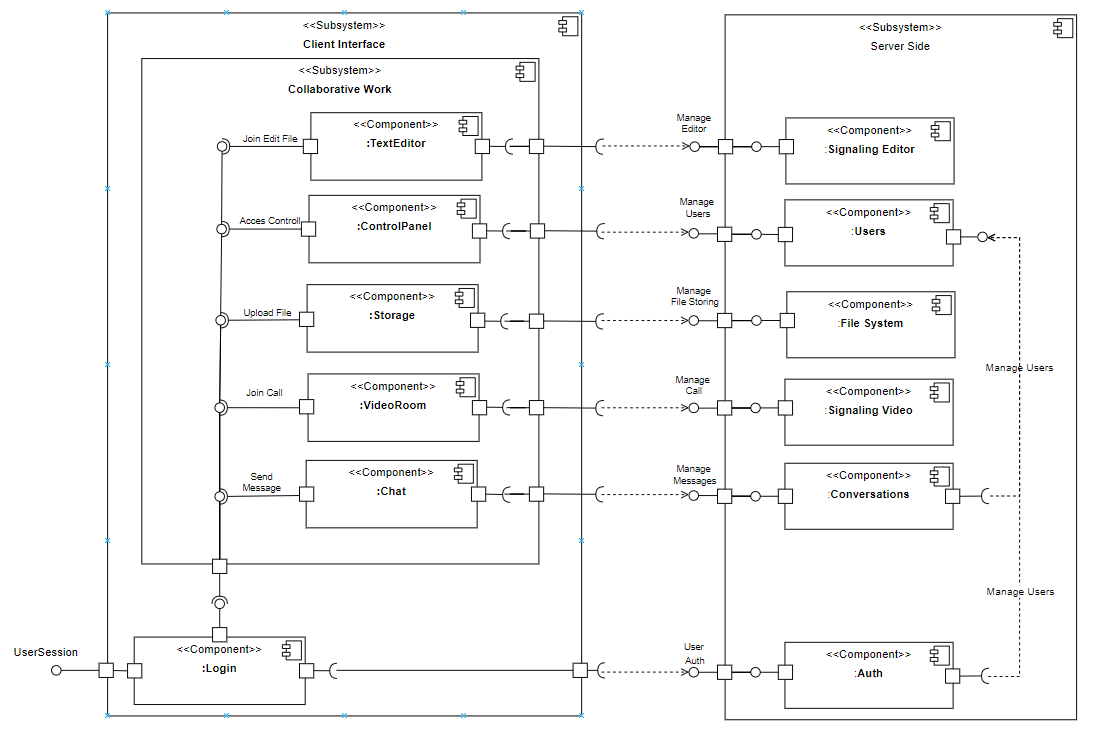
Soluţia dezvoltată este formată din aplicaţia front-end, reprezentată în partea stângă a figurii ??? şi aplicaţia server reprezentată în partea din dreapta. Utilizatorul interacţionează cu aplicaţia interfaţă pentru a beneficia de funcţionalităţile puse la dispoziţie de aceasta, care la rândul său face request-uri către server pentru procesare. Utilizatorul nu are acces în niciun moment la infrastructura serverului şi nu poate interacţiona cu aceasta.



Pentru intermedierea accesului dintre utilizator şi interfaţă voi folosi un server web Nginx care controlează accesul utilizatorilor prin solicitarea certificatului de utilizator la momentul accesării paginii. Astfel, accesul va fi permis doar pentru accesul la interfaţa web, iar serverul Nginx având rolul de API *gateway*. Reprezentarea pentru modelul infrastructurii va fi reprezentată în secţiunea de implementarea aplicaţiei. În continuare voi descrie principale părţi componente ale sistemului folosind reprezentarea grafică de tip UML.

### Aplicaţia Web Client

Aplicaţia va pune la dispoziţie o pagină pentru autentificarea în sistem a utilizatorului. Dacă se încearcă accesarea unei alte resurse din aplicaţie prin accesarea URL-ului respectiv, fără ca utilizatorul să fie deja autentificat, atunci acesta va fi redirecţionat spre pagina de autentificare sau Login. După validarea credenţialelor, utilizatorul poate accesa funcţionalităţile aplicaţiei reprezentate în diagrama componentelor reprezentate în figura ???.



Principalele componente ale aplicaţiei sunt modulele:

* *Chat*;
* *VideoRoom*;
* *Storage*;
* *ControlPanel*;
* *TextEditor*.

*Chat* permite utilizatorului să acceseze conversaţii existente sau să creeze conversaţii noi la nivelul cărora să trimită mesaje de tip text sau fişiere multimedia. De asemenea, utilizatorul are opţiunea de a crea grupuri noi şi de a adăuga noi utilizatori sau de a şterge grupurile deja existente.

*VideoRoom* oferă funcţionalitatea de a genera apeluri video-audio la nivelul grupurilor, dar şi adăugarea altor participanţi din afara grupurilor prin accesarea adresei URL specifică paginii.

*Storage* pune la dispoziţie un spaţiu de stocare unde sunt salvate fişierele trimise la nivelul conversaţiilor, iar pe lângă acestea utilizatorul poate uploada şi fişiere proprii, private.

*ControlPanel* poate fi accesat doar de utilizatorii cu rol de administrator. Această pagină reprezintă un panou unde doar utilizatorii cu rol de administrator pot filtra restul utilizatorilor în funcţie de grupurile în care sunt înrolaţi şi restricţiona accesul.

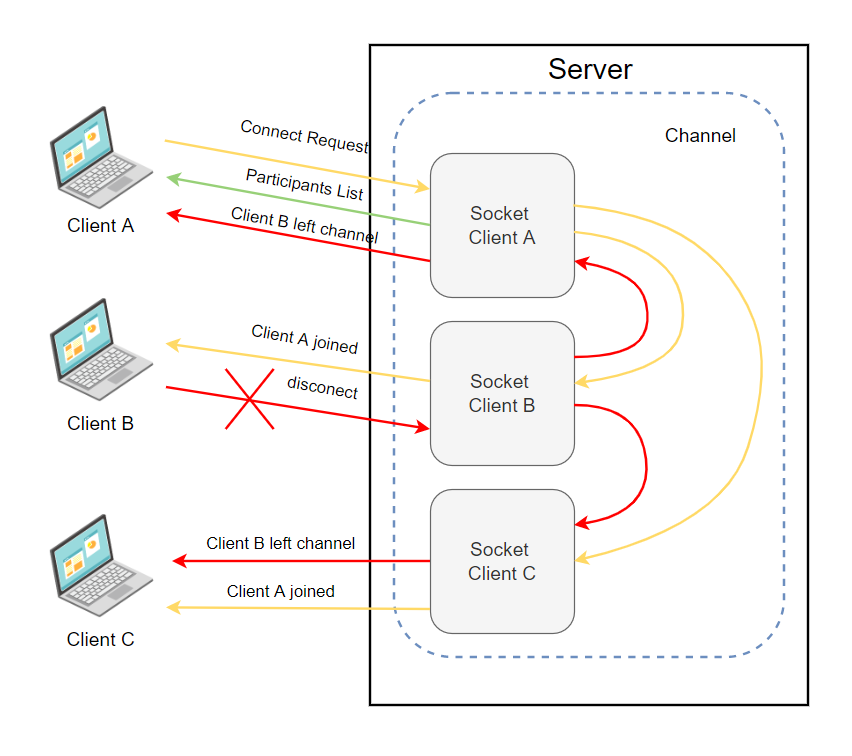
*TextEditor* este modulul destinat editării documentelor în mod colaborativ. Acesta permite utilizatorilor să interacţioneze prin intermediul unui editor text, fiecare având un cursor customizat pentru interacţiunea cu restul utilizatorilor. La finalul sesiunii de lucru, documentul este salvat în spaţiul de stocare al autorului şi va putea fi accesat doar de utilizatorii care au permisiuni de acces. De menţionat că, pe timpul sesiunii de lucru pot fi adăugaţi oricât de mulţi utilizatori prin distribuirea link-ului de acces al pagini.

### Aplicaţiei Web Server

În partea dreaptă a figurii Fig. ??? sunt reprezente componentele aplicaţiei server. Back-end-ul aplicației este format din două servicii API de tratare al request-urilor care asigură logica funcțională a aplicației. Primul serviciu este sistemul pentru tratarea request-urilor de tip POST și GET, iar cel de-al doilea serviciu este sistemul de comunicaţie dintre utilizatori, bazat pe protocolul Websocket. Acesta este folosit pentru trimiterea de mesaje, iniţierea apelurilor video și editarea colaborativă.

Folosirea unui serviciu de comunicaţii bazat pe protocolul WebSocket este justificat de necesitatea transmisiei datelor aproape instant către grupurile corespunzătoare. Socket.IO, permite diferenţierea fiecărui terminal conectat prin atribuirea unui *ID* unic care poate fi folosit la diferenţierea şi gruparea utilizatorilor pe canale de comunicaţie foarte uşor.

Astfel, pentru fiecare conversaţie de tip chat, modulul *Conversations* creează câte un nou canal pe baza *ID-ului* specific grupului, iar atunci când un utilizator se autentifică, acesta este conectat la toate canalele asociate conversaţiilor sale din baza de date. După validarea criterilor de verificare, se salvează la nivelul serverului ID-ul utilizatorului împreună cu ID-ul socket-ului ţi tipul canalului într-o lista pentru un managementul canalelor. În aceeaşi manieră, atunci când un utilizator se alătură unui apel video, acesta conectat la canalul respectiv prin intermediul modulului *Signaling Video* pentru a primii lista cu participanţii care sunt deja conectaţi, Fig. ???. Pentru participarea la editarea unui document se procedează în mod asemănător, dar folosind modului de Signaling Editor. În momentul în care socket-ul este deconectat este este semnalat evenimentul la nivelul serverului pentru a-l exclude din lista utilizatorilor conectaţi.



### Baza de date

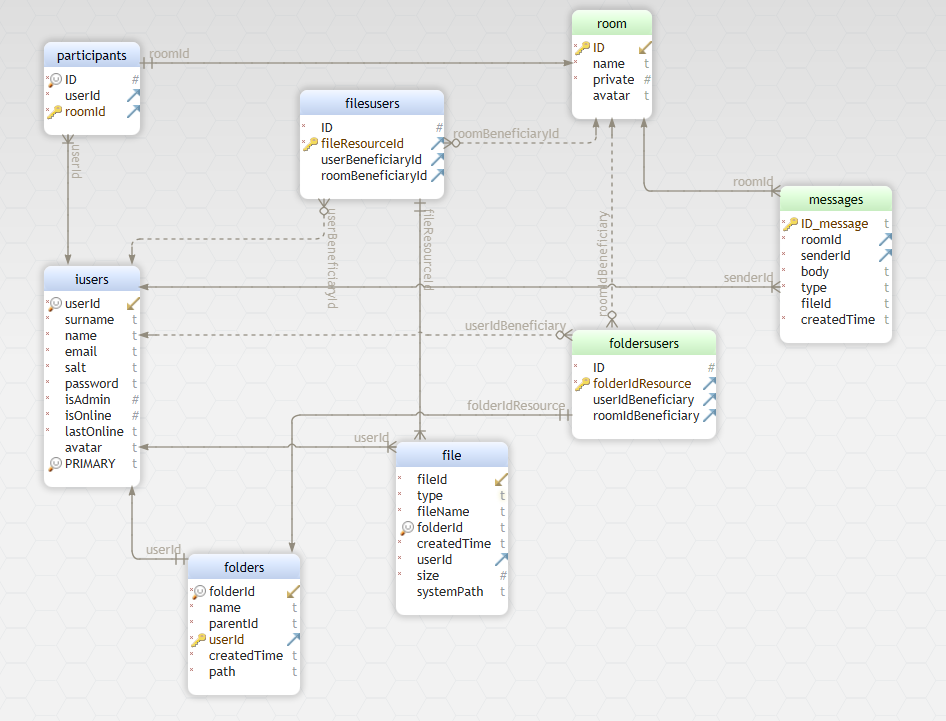
Pentru salvarea datelor am ales o bază de date relaţională de tip MySQL, deoarece pe baza relaţiilor ce se stabilesc între tabele, datele pot fi organizate mai uşor , iar aplicaţia nu necesită stocarea unui volum foarte mare de date. Pentru crearea şi administrarea bazei de date am utilizat utilitarul XAMPP, deoarece pune la dispoziţie interfaţă grafică uşor de folosit.Baza de date este formată din opt tabele legate între ele folosind constrângeri de tip cheie valoare.

Tabelul *iusers* reţine datele despre utilizatori. Pentru stocarea datelor despre conversaţii s-a folosit tabelul *room*. Pentru reprezentarea participării unui utilizator la o conversaţie este necesară completarea tabelului *participants* cu ID grupului, respectiv al utilizatorului. Astfel, în cazul unei conversaţii private vor exista două intrări în tabelul *participants*.

Tabelul *folders* reprezintă datele ce emuleză structura de tip folder din cadrul spaţiului de stocare. Pentru ca un utilizator să poată accesa un anumit folder, este necesară completarea unei noi intrări în tabelul *foldersusers*.

Tabelul *file* reţine datele despre fişierele trimise de către utilizatori. Pentru a furniza utilizatorilor acces la un anumit fişier (în cazul în care se doreşte partajarea unui anumit fişier) este nevoie să fie completat un nou set de date format din ID-ul fişierului, ID-ul utilizatorului şi ID-ul grupului. Dacă fişierul aparţine unei singure persoane, ID-ul grupului va fi *NULL*.

Pentru stocarea mesajelor va fi utilizat tabelul *messages*, în care vor fi stocate ID-ul grupului căruia aparţin, ID-ul autorului, timpul când a fost trimis şi tipul de mesaj (dacă este un mesaje de tip text sau fişier).



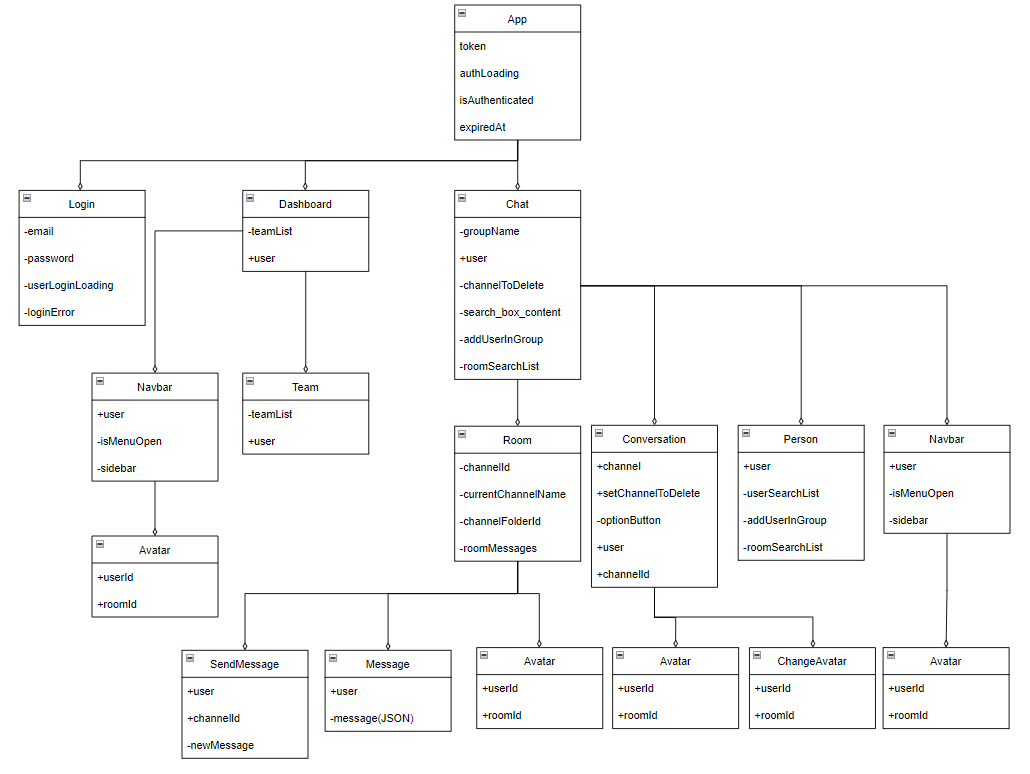
# Implementarea software

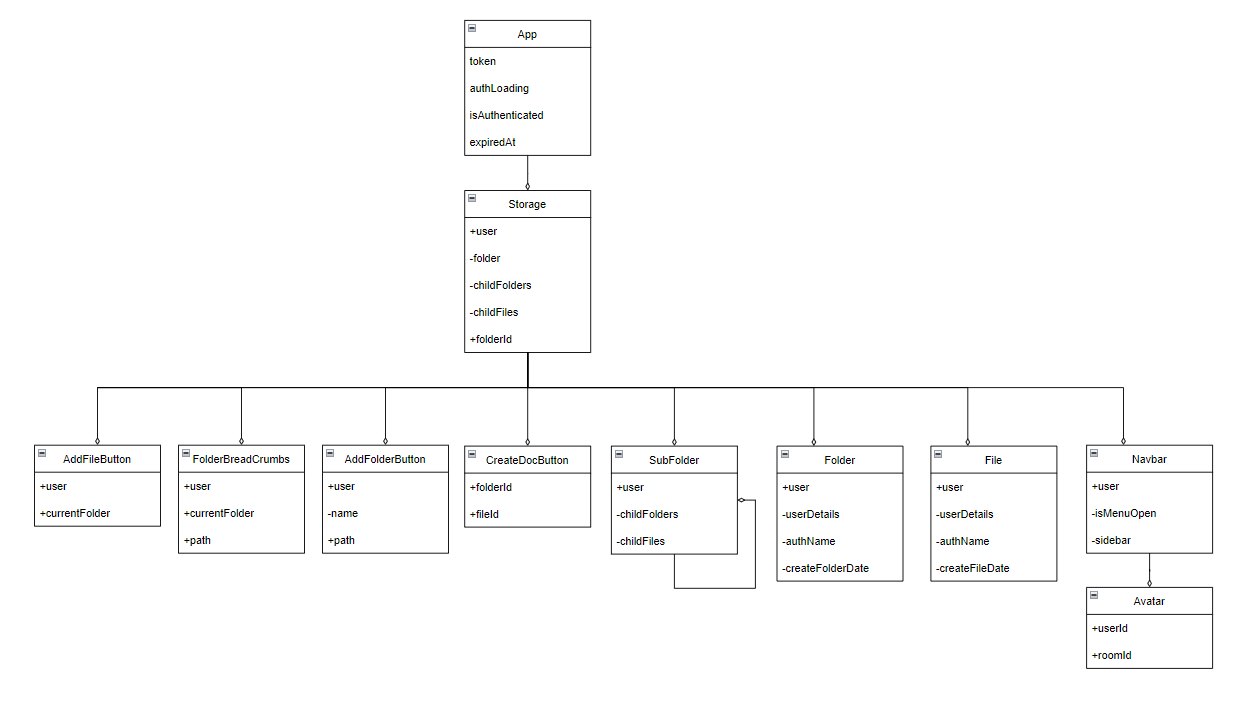
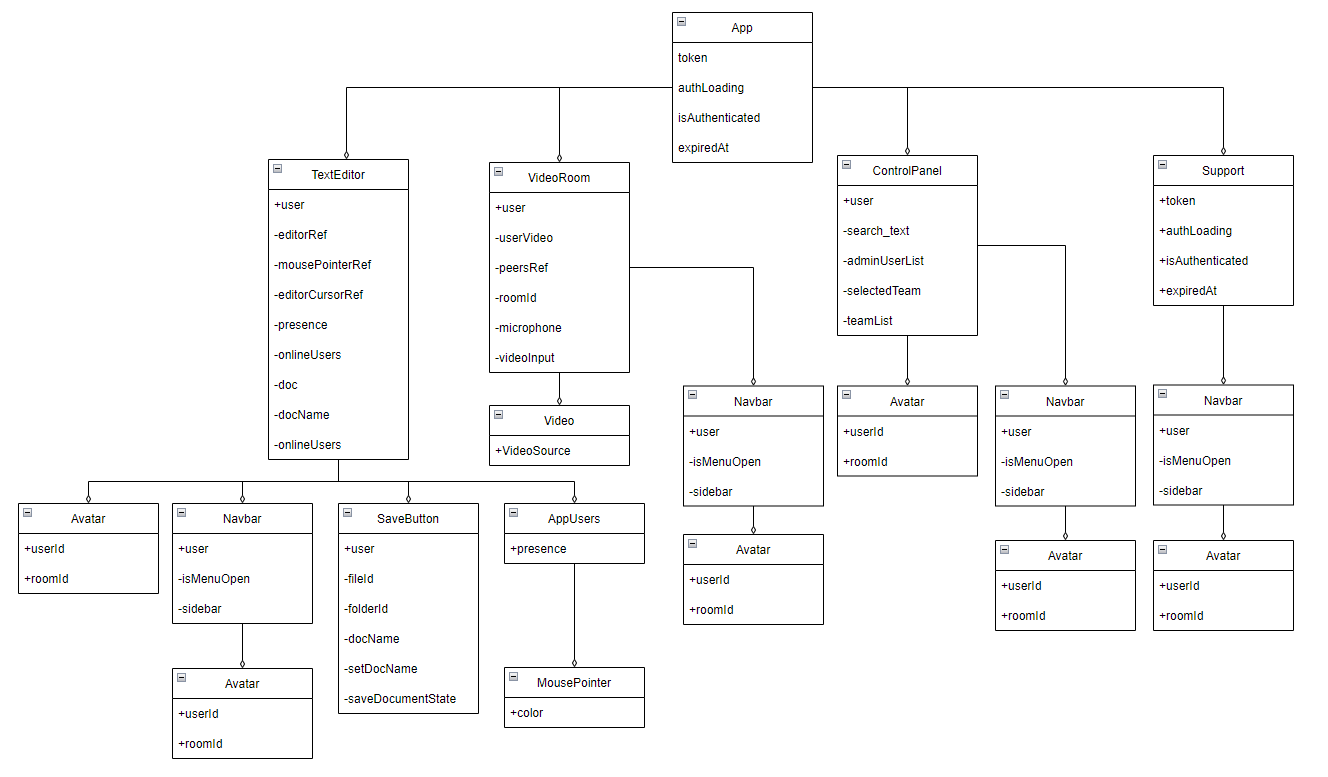
Pentru dezvoltarea aplicaţiei am folosit utilitarul Visual Studio Code datorită editorului pus la dispoziţie de acesta, ce uşurează munca prin evidenţierea sintaxei, potrivirea parantezelor şi identarea automată.

Mediul folosit pentru devoltarea aplicaţiei a fost NodeJS şi alte biblioteci şi framework-uri specificce Javascript enumerate în capitolul „API-uri folosite”. Datorită utilizării limbajului Javasript, pentru dezvoltarea aplicaţiei Client am avut de ales paradigma de programare funcţională în detrimentul celei orientate pe obiecte.

Programarea funcţională este o paradigmă de programare declarativă ce constă în folosirea de funcţii pure, mai exact funcţii ce primesc variabile de intrare pe care nu le modifică şi generează altele noi ca rezultat. Astfel, rezultatul unei funcţii pure depinde doar de parametrii de intrare, fără să producă impact asupra celorlalte componente din cod. Programarea orientată pe obiecte este o paradigmă de programare care organizează datele și structura software-ului pe baza conceptului de clase și obiecte.

## Aplicaţia Web Client





## Aplicaţia web server

### Răspunderea request-urilor

## Implementarea bazei de date

### Diagramă

## Metode de securizare

### Securizarea canalului de comunicaţie

### Criptarea fişierelor stocate

### Protecţie împotriva atacurilor CSRF

### Controlul accesului prin firewall

### Alertare în cazul încercărilor neautorizate

# Utilizarea aplicaţiei

## Actorii sistemului

## Cazurile de utilizare

## Testarea sistemului

## Raport de testare

## Diagrame UML

### Diagrama cazurilor de utilizare

### Diagrama de activităţi

# Concluzii

## Analizarea rezultatelor obţinute

## Dezvoltări ulterioare

# Bibliografie

[1] “What is Collaboration Software?,” Mar. 2021. https://kissflow.com/digital-workplace/collaboration/collaboration-software-guide/ (accessed May 13, 2022).

[2] Shari Kjerland, “Microsoft Teams service description,” 2022. https://docs.microsoft.com/en-us/office365/servicedescriptions/teams-service-description (accessed Jun. 13, 2022).

[3] “Prezentarea experienței Microsoft teams pentru educație simplificată.” https://support.microsoft.com/ro-ro/topic/prezentarea-experien%C8%9Bei-microsoft-teams-pentru-educa%C8%9Bie-simplificat%C4%83-fd5b0668-4156-4ce1-a51a-e6f54827973d (accessed May 13, 2022).

[4] Donna Tam, “Flickr founder plans to kill company e-mails with Slack.” https://www.cnet.com/tech/tech-industry/flickr-founder-plans-to-kill-company-e-mails-with-slack/ (accessed May 13, 2022).

[5] “Slack for Windows.” https://slack.com/downloads/windows (accessed May 13, 2022).

[6] “Discord Support.” https://support.discord.com/hc/en-us/articles/360041721052-Video-Calls (accessed May 13, 2022).

[7] “MTProto Mobile Protocol.” https://core.telegram.org/mtproto (accessed May 13, 2022).

[8] Marino Miculan and Nicola Vitacolonna, “Formal verification of Telegram chat protocol MTProto 2.0.” https://github.com/miculan/telegram-mtproto2-verification (accessed May 13, 2022).

[9] Cesar Ghali, Adam Stubblefield, J. L. Ed Knapp, Benedikt Schmidt, and Julien Boeuf, “Application Layer Transport Security - White Paper”.

[10] Manuela Aparicio and Carlos J. Costa, “Collaborative Systems: Characteristics and Features,” p. 1, 2012.

[11] Michel Beaudoin-Lafon, *Computer Supported Co-operative Work*. 1999.

[12] C. Sun, “OT FAQ.” https://web.archive.org/web/20200623064915/https://www3.ntu.edu.sg/home/czsun/projects/otfaq/ (accessed May 15, 2022).

[13] Marc Shapiro, Nuno Preguiça, Carlos Baquero, and Marek Zawirski, “Conflict-Free Replicated Data Types,” pp. 2–5, 2014.

[14] Salvatore Loreto and pietro Romano, *Real-Time Communication with WebRTC*. 2014.

[15] Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila, “A Formal Security Analysis of the Signal Messaging Protocol,” pp. 1–5, 2019, Accessed: May 17, 2022. [Online]. Available: https://eprint.iacr.org/2016/1013.pdf

[16] “About Node.js.” https://nodejs.org/en/about/ (accessed May 18, 2022).

[17] “What is Express.js\_.” https://www.besanttechnologies.com/what-is-expressjs (accessed May 18, 2022).

[18] “React Hooks.” https://www.javatpoint.com/react-hooks (accessed May 19, 2022).

[19] David Choi, *Full-Stack React, TypeScript, and Node*. 2020.

[20] “What Socket.IO is.” https://socket.io/docs/v4/ (accessed May 18, 2022).

[21] “Quill Documentation.” https://quilljs.com/docs/quickstart/ (accessed May 19, 2022).

# Anexe

1. SR – abreviere pentru Software Requirements Specification [↑](#footnote-ref-1)