**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**



**APLICAŢIE DE TIP COLABORATIV PROTEJATĂ ÎN CLOUD**

CONDUCĂTOR ȘTIINȚIFIC:

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

ABSOLVENT:

**Stud. 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**

Această lucrare prezintă o aplicaţie software de tip colaborativ ce are ca funcţionalităţi schimbul de mesaje text sau de fişiere multimedia, editarea de text în timp real, stocarea de fişiere la distanţă şi relizarea de apeluri video şi audio între participanţi.

Capitolul 1 prezintă importanţa dezvoltării unui astfel de sistem colaborativ propriu în locul alegerii unei soluţii deja existente.

Capitolul 2 descrie soluţii software similare şi prezintă tehnologiile specifice folosite de acestea la momentul actual. Printre aceste tehnologii se află algoritmi folosiţi în cadrul editării de text colaborative şi metode de comunicaţie utilizate în cadrul acestor aplicaţii.

În Capitolul 3 sunt prezentate bibliotecile şi API-uri folosite pentru dezvoltarea sistemului. Acest capitol fundamentează bazele implementării software alături de Capitolul 4, unde sunt prezintate cerintele aplicaţiei, arhitectura şi modul acesteia de funcţionare folosind diagrame de componente.

Capitolul 5 prezintă implementarea software a principalelor module din aplicaţie, explicând relaţiile dintre componente şi modul prin care intercacţionează la nivelul sistemului.

Capitolele 6 şi 7 prezintă testarea sistemului prin utilizarea de diagrame UML şi prin prezentarea unui plan de testare configurat pe baza cerinţelor funcţionale.

**Cuprins**

[LISTĂ DE ABREVIERI 15](#_Toc107779316)

[TABELĂ FIGURI 16](#_Toc107779317)

[1. Introducere 17](#_Toc107779318)

[1.1 Importanţa temei 17](#_Toc107779319)

[1.2 Scopul lucrării 17](#_Toc107779320)

[1.3 Utilitatea sistemului 17](#_Toc107779321)

[2 Stadiul actual 19](#_Toc107779322)

[2.1 Software colaborativ 19](#_Toc107779323)

[2.2 Sisteme similare 19](#_Toc107779324)

[2.2.1 Microsoft Teams 19](#_Toc107779325)

[2.2.2 Slack 20](#_Toc107779326)

[2.2.3 Discord 20](#_Toc107779327)

[2.2.4 Telegram 21](#_Toc107779328)

[2.2.5 Google Docs 22](#_Toc107779329)

[2.3 Principii de funcţionare 23](#_Toc107779330)

[2.3.1 Editarea colaborativă 23](#_Toc107779331)

[2.3.2 OT 24](#_Toc107779332)

[2.3.3 CRDT 26](#_Toc107779333)

[2.3.4 WebRTC 27](#_Toc107779334)

[2.3.5 Signal Protocol 28](#_Toc107779335)

[3 API-uri folosite 29](#_Toc107779336)

[3.1 Aplicaţia Server 29](#_Toc107779337)

[3.2 Aplicaţia Client 30](#_Toc107779338)

[3.3 SocketIo 32](#_Toc107779339)

[3.4 Quill 33](#_Toc107779340)

[4 Structura proiectului 35](#_Toc107779341)

[4.1 Cerinţele sistemului 35](#_Toc107779342)

[4.1.1 Cerinţe Non-Funcţionale 35](#_Toc107779343)

[4.1.2 Cerinţe Funcţionale 35](#_Toc107779344)

[4.2 Arhitectura sistemului 36](#_Toc107779345)

[4.2.1 Aplicaţia Web Client 37](#_Toc107779346)

[4.2.2 Aplicaţiei Web Server 39](#_Toc107779347)

[4.2.3 Structura bazei de date 40](#_Toc107779348)

[5 Implementarea software 43](#_Toc107779349)

[5.1 Aplicaţia Web Client 43](#_Toc107779350)

[5.1.1 Definirea Componentelor 43](#_Toc107779351)

[5.1.2 React-Redux 49](#_Toc107779352)

[5.1.3 Definirea Request-urilor 50](#_Toc107779353)

[5.2 Aplicaţia web server 51](#_Toc107779354)

[5.2.1 Tratarea request-urilor 51](#_Toc107779355)

[5.2.2 Baza de date MySQL 52](#_Toc107779356)

[5.3 Metode de securizare 53](#_Toc107779357)

[5.3.1 Protecţie împotriva atacurilor CSRF 53](#_Toc107779358)

[5.3.2 OpenVPN 55](#_Toc107779359)

[5.3.3 Autentificare mutuală 56](#_Toc107779360)

[6 Testarea sistemului 59](#_Toc107779361)

[6.1 Utilizarea aplicaţiei 59](#_Toc107779362)

[6.2 Testarea sistemului 66](#_Toc107779363)

[6.3 Diagrame UML 69](#_Toc107779364)

[6.3.1 Diagrama cazurilor de utilizare 69](#_Toc107779365)

[6.3.2 Diagrama de activităţi 70](#_Toc107779366)

[7 Concluzii 71](#_Toc107779367)

[7.1 Probleme întâmpinate 71](#_Toc107779368)

[7.2 Rezultate obţinute 71](#_Toc107779369)

[7.3 Dezvoltări ulterioare 72](#_Toc107779370)

[8 Bibliografie 73](#_Toc107779371)

[9 Anexe 74](#_Toc107779372)

[9.1 Anexa A 74](#_Toc107779373)

[9.2 Anexa B 74](#_Toc107779374)

# 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 |
|  | NPM | Node Package Manager |
|  | VPN | Virtual Private Network |
|  | SQL | Structured Query Language |

# TABELĂ FIGURI

[Figură 2.1 Microsoft Teams (Versiunea Desktop) – HomePage [3] 20](#_Toc107780121)

[Figură 2.2 Discord (Versiunea Desktop) - Homepage[4] 21](#_Toc107780122)

[Figură 2.3 Google Docs - Editare colaborativă 22](#_Toc107780123)

[Figură 2.4 Creare de conflict prin accesare concurenţială 24](#_Toc107780124)

[Figură 2.5 Mecanism Transformare Operaţională[10] 25](#_Toc107780125)

[Figură 2.6 WebRTC - Transmisia datelor[12] 27](#_Toc107780126)

[Figură 3.1 Modelul asincron de tratare al evenimentelor - NodeJS 29](#_Toc107780127)

[Figură 3.2 Iniţierea Protocolului WebSocket 32](#_Toc107780128)

[Figură 3.3 Parametrii handShake pentru conexiunea de tip Socket.IO [18] 33](#_Toc107780129)

[Figură 4.1 Reprezentarea arhitecturii sistemului software 37](#_Toc107780130)

[Figură 4.2 Diagrama de componente a sistemului 38](#_Toc107780131)

[Figură 4.3 Fluxul de date în cadrul unui canal de comunicaţie Socket.IO 39](#_Toc107780132)

[Figură 4.4 Structura bazei de date 40](#_Toc107780133)

[Figură 5.1 Diagrama obiectelor de tip React de la nivelul aplicaţiei client – Partea 1 44](#_Toc107780134)

[Figură 5.2 Trimiterea request-ului de conectare la sesiunea de editare colaborativă 45](#_Toc107780135)

[Figură 5.3 Diagrama obiectelor de tip React de la nivelul aplicaţiei client – Partea 2 47](#_Toc107780136)

[Figură 5.4 Diagrama obiectelor de tip React de la nivelul aplicaţiei client – Partea 3 48](#_Toc107780137)

[Figură 5.5 Principiu de funcţionare al sistemului React Redux 49](#_Toc107780138)

[Figură 5.6 Componentele de tip Reducer de la nivelul aplucaţiei client 50](#_Toc107780139)

[Figură 5.7 Exemplu request de tip post pentru adăugare unui utilizator nou în grup 51](#_Toc107780140)

[Figură 5.8 Folosirea funcţiei middleware pentru verificarea autenticităţii request-ului 51](#_Toc107780141)

[Figură 5.9 Interogare SQL pentru returnarea listei de participanţi din cadrul unui grup 53](#_Toc107780142)

[Figură 5.10 Diagramă atac de tip CSRF 54](#_Toc107780143)

[Figură 5.11 Securizarea sesiunii folosind JWT token 54](#_Toc107780144)

[Figură 5.12 Rularea serviciului la nivelul utilizatorului pentru accesarea aplicaţiei 55](#_Toc107780145)

[Figură 5.13 Sesiune de administrare a serverului folosind conexiune SSH 56](#_Toc107780146)

[Figură 5.14 Mod de intercacţiune folosind comunicaţia securizată 56](#_Toc107780147)

[Figură 5.15 Fişierul de configurare al serverul de Nginx 56](#_Toc107780148)

[Figură 5.16 Accesarea aplicaţiei fără un deţinerea certificatului de utilizator 57](#_Toc107780149)

[Figură 5.17 Solicitarea certificatului de utilizator la nivelul browser-ului 57](#_Toc107780150)

[Figură 6.1 Pagina de Login a aplicaţiei 59](#_Toc107780151)

[Figură 6.2 Prezentate Fereastră de Homepage 60](#_Toc107780152)

[Figură 6.3 Accesarea meniului de navigaţie 60](#_Toc107780153)

[Figură 6.4 Prezentare funcţionalitate de schimbare a profilului 61](#_Toc107780154)

[Figură 6.5 Prezentare funcţionalitate de editare a datelor de utilizator 61](#_Toc107780155)

[Figură 6.6 Accesarea serviciului de mesagerie 62](#_Toc107780156)

[Figură 6.7 Exemplu de trimitere a unui fişier în cadrul conversaţiei 62](#_Toc107780157)

[Figură 6.8 Prezentarea funcţionalităţilor din cadrul ferestrei de mesagerie 63](#_Toc107780158)

[Figură 6.9 Prezentarea sistemului de stocare pentru fişiere 63](#_Toc107780159)

[Figură 6.10 Sesiune de editare colaborativă a textului 64](#_Toc107780160)

[Figură 6.11 Salvarea documentului generat în cadrul sesiunii de editare 64](#_Toc107780161)

[Figură 6.12 Prezentarea sesiunii de comunicaţie video-audio 65](#_Toc107780162)

[Figură 6.13 Fereastra de control destinată administratorilor 65](#_Toc107780163)

[Figură 6.14 Diagrama cazurilor de utilizare 69](#_Toc107780164)

[Figură 6.15 Diagrama de activităţi 70](#_Toc107780165)

# 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 hardware 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. 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 folosind resurse locale), dar şi asupra utilizatorilor - în cazul în care este detectată o activitate suspectă poate fi restricţionat accesul asupra aplicaţiei. 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.

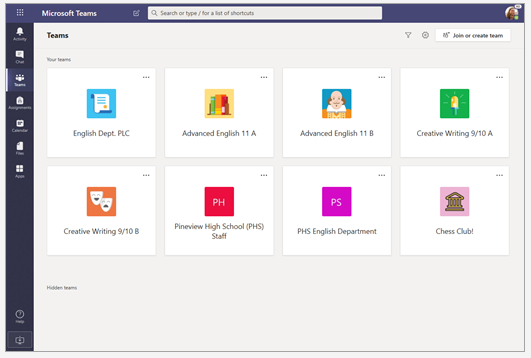
## Sisteme similare

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.



Figură 2.1 Microsoft Teams (Versiunea Desktop) – HomePage [3]

### Slack

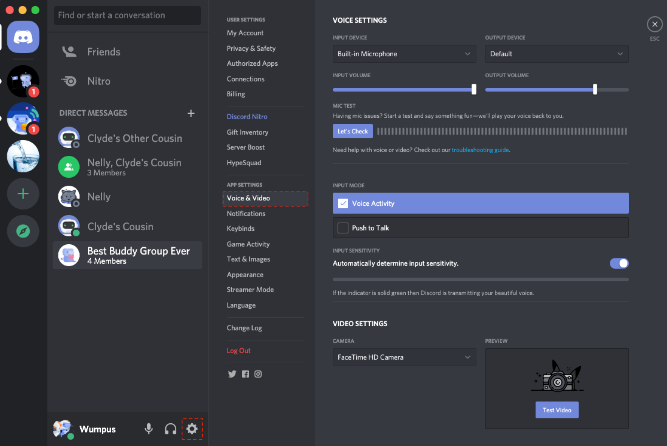
Slack este un serviciu software de mesagerie instantanee lansat în anul 2013 şi conceput special pentru mediul de lucru din cadrul organizaţiilor. Potrivit deţinătorului companiei, numele aplicaţiei este un acronim pentru „Searchable Log of All Conversation and Knowledge”, astfel dezvăluind principala caracteristică a produsului, aceea ca toate conversaţiile şi fişierele partajate să poată fi căutate mai uşor faţă de alte dispozitive colaborative[3]. 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.

### 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.



Figură 2.2 Discord (Versiunea Desktop) - Homepage[4]

### Telegram

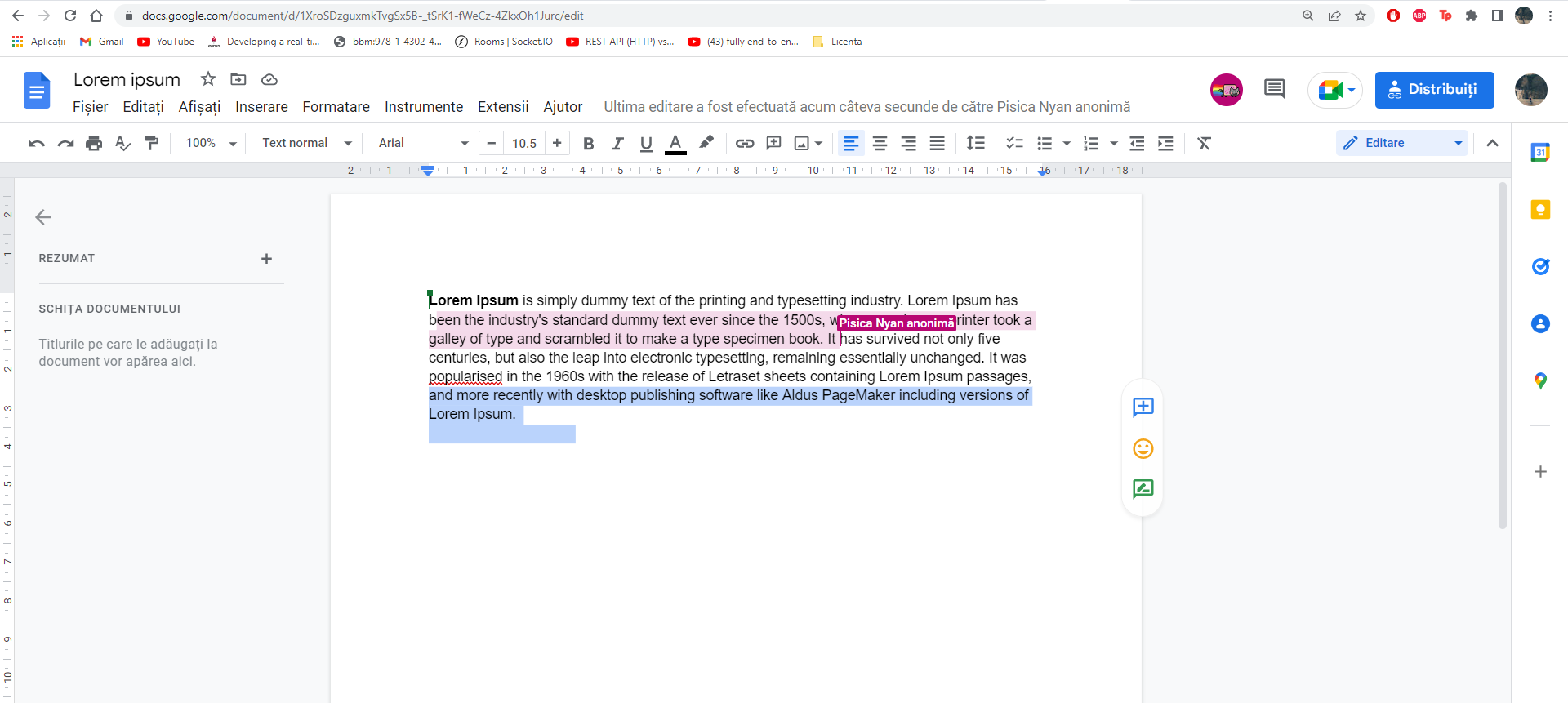
Spre deosebire de celelalte aplicaţii prezentate până acum, Telegram este o aplicaţie colaborativă de tip open source ce prestează servicii de mesagerie text şi VoIP. Aplicaţia se remarcă prin numărul foarte mare de participanţi ce pot face parte dintr-un grup la un moment dat, în momentul de faţă acesta 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 acestea 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[5]. 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. [6]

### 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.[7]



Figură 2.3 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. 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[8], 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.

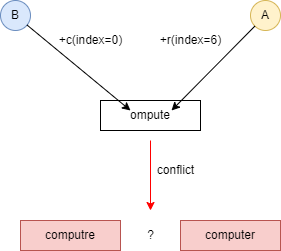
În continuare, voi descrie o parte din conceptele care stau la baza sitemelor colaborative din cadrul editării de text, partajării de flux video şi asigurarea confidenţialitate.

### 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.



Figură 2.4 Creare de conflict prin accesare concurenţială

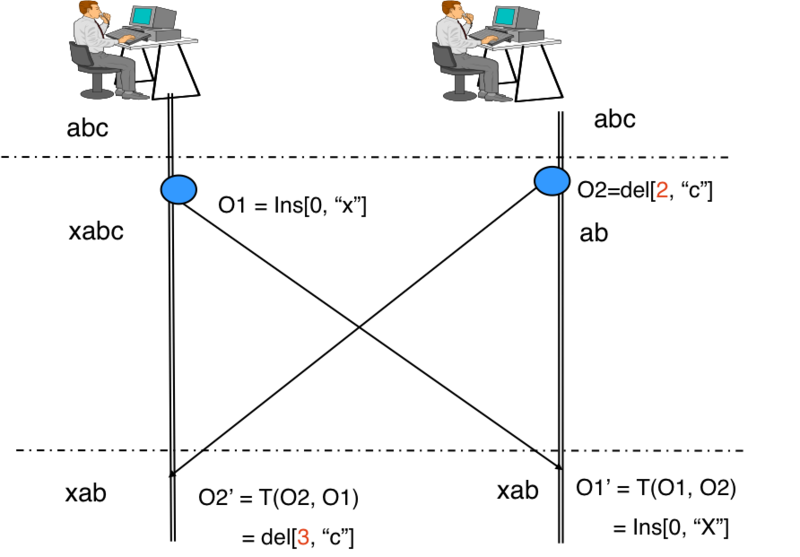
Un document conţine un şir de caractere „ompute”. Să presupunem că utilizatorul A încearcă să insereze caracterul „r” după caracterul „e”. Majoritatea editoarelor text folosesc funcţii de inserare care utilizează un parametru pentru indexarea poziţiei din document, mai exact o funcţie care va adăuga 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”[9].

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 rezolvarea conflictelor, notificarea și comprimarea operațiilor, conștientizarea grupului, partajarea aplicațiilor și instrumentelor de proiectare media asistată de calculator în colaborare[10]. 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.



Figură 2.5 Mecanism Transformare Operaţională[10]

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[11] 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.

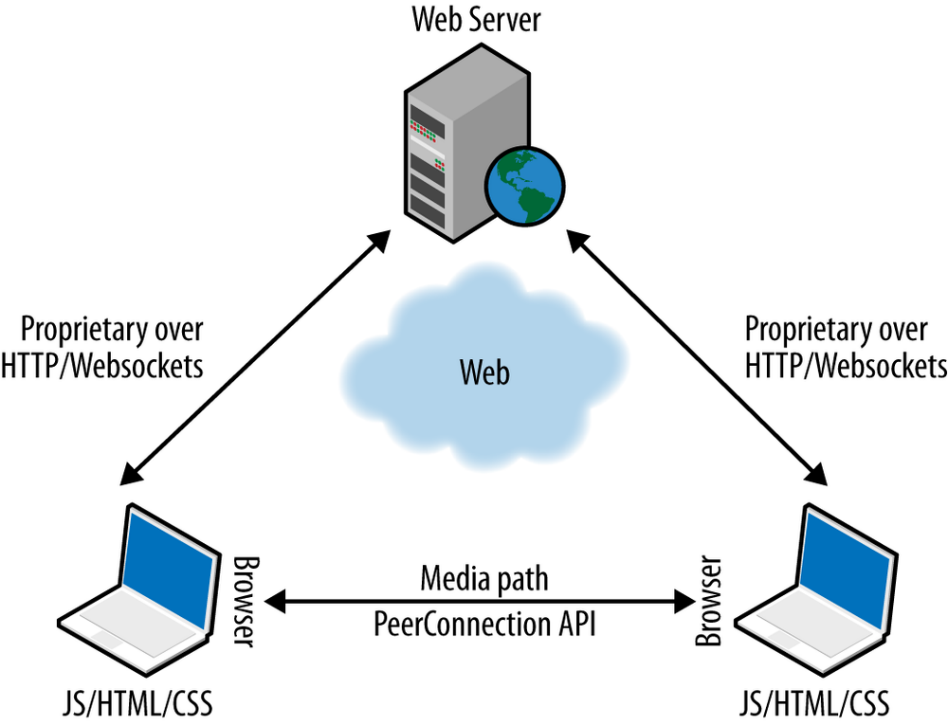
Sistemele CRDT bazate pe stare sunt adesea mai simplu de proiectat și de implementat; singura lor cerință de la substratul de comunicaţie fiind mesajele despre starea celorlalte sisteme replicate. Dezavantajul lor este că întreaga stare a fiecărei CRDT trebuie transmisă în cele din urmă fiecărei alte 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[11].

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 şi primesc ca răspuns informaţiile solicitate.

WebRTC extinde semantica clasică de tip client-server prin introducerea unei paradigme de comunicaţie 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 folosind 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 figura 2.6.



Figură 2.6 WebRTC - Transmisia datelor[12]

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 comunicaţie 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 scurt, mediu și îndelungat 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 de tip 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.

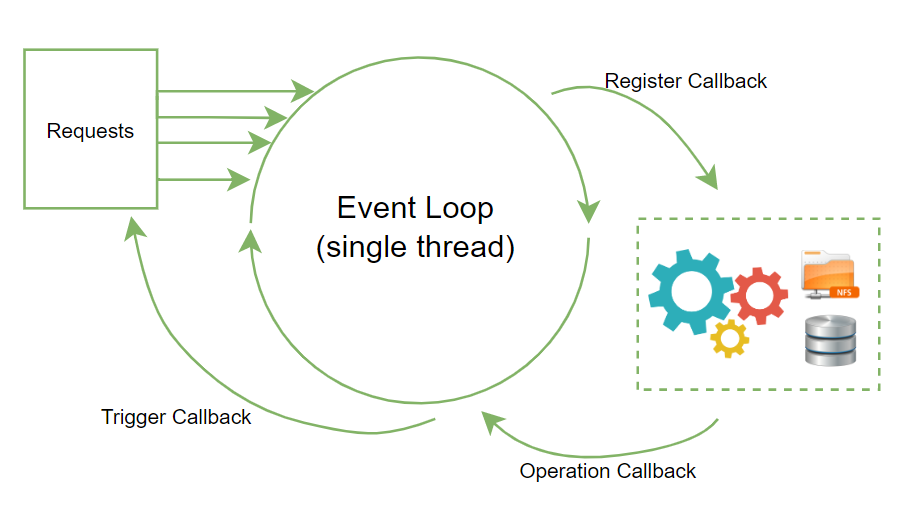
Mecanismele de tip „ratchet” generează o nouă cheie privată pe baza celei precedente, oferind sistemului confidenţialitate în cazul în care una din cheile private de bază este compromisă. Astfel, un posibil atacator nu va reuşi să decripteze decât o serie minimă de mesaje[13].

# 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 figura 3.1). 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[14].



Figură 3.1 Modelul asincron de tratare al evenimentelor - NodeJS

Express.js este un framework creat pentru mediul de execuţie NodeJS şi 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[15]. 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 comunicaţia 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

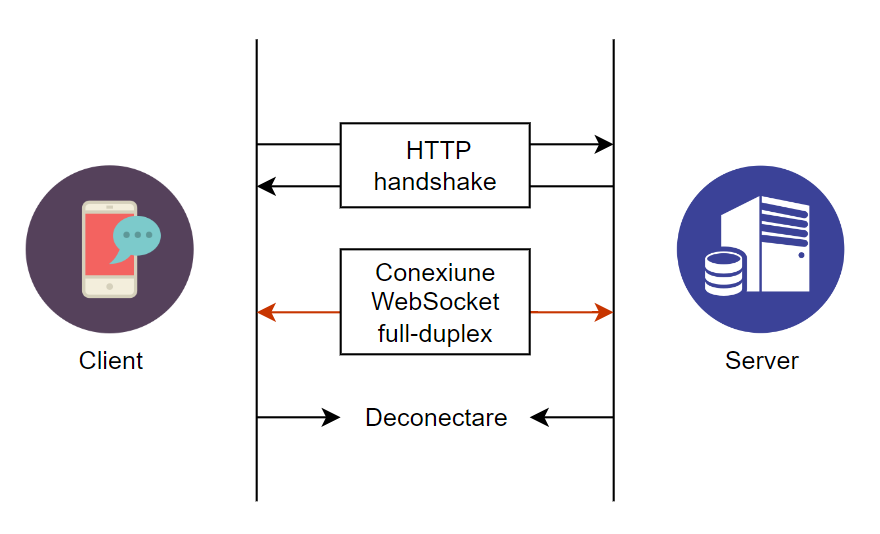
Pentru implementarea aplicaţiei client, am ales biblioteca ReactJS *v18.2.0*, datorită popularităţii sale la momentul actual. Această bibliotecă apartanentă JavaScript este utilizată pentru dezvoltarea de interfeţe interactive cu utilizatorul şi se remarcă prin dezvoltarea componentelor modulare care pot fi reutilizate, încorporând modelul de proiectare MVC. Alte componente API notabile folosite pentru realizarea aplicaţiei client sunt:

* *React Hooks* – este o funcţionalitate introdusă în versiunea React 16.8. Aceasta permite utilizarea stării componentelor specifice React fără folosirea unui obiect de tip clasă (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[16];
* *React Redux* – permite componentelor React să acceseze variabile salvate într-un mediu de stocare numit Redux Store. Atunci când variabilele sunt modificate, toate componentele ce depind de folosirea lor sunt redate;
* *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. Semnalizarea se referă la schimbul de mesaje de start pentru stabilirea conexiunii;
* *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[17].

## SocketIo

WebSocket este un protocol de comunicații care oferă canale de comunicaţie full-duplex utilizând 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.



Figură 3.2 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[18]. 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.



Figură 3.3 Parametrii handShake pentru conexiunea de tip Socket.IO [18]

Parametrii stabiliţi la handshake (figura 3.3) 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*[19].

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 editorului).

# 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, de tip interfaţă web şi aplicaţia server. Cerinţele non-funcţionale ce trebuie îndeplinite de sistem sunt prezentate mai jos.

**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 atingerea scopului şi asigurarea unui instrument facil de lucru:

**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.

**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** – 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 10** – 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 11** – Funcţionalitate de creare a fişierelor text şi de editare simultană de către doi sau mai mulţi utilizatori.

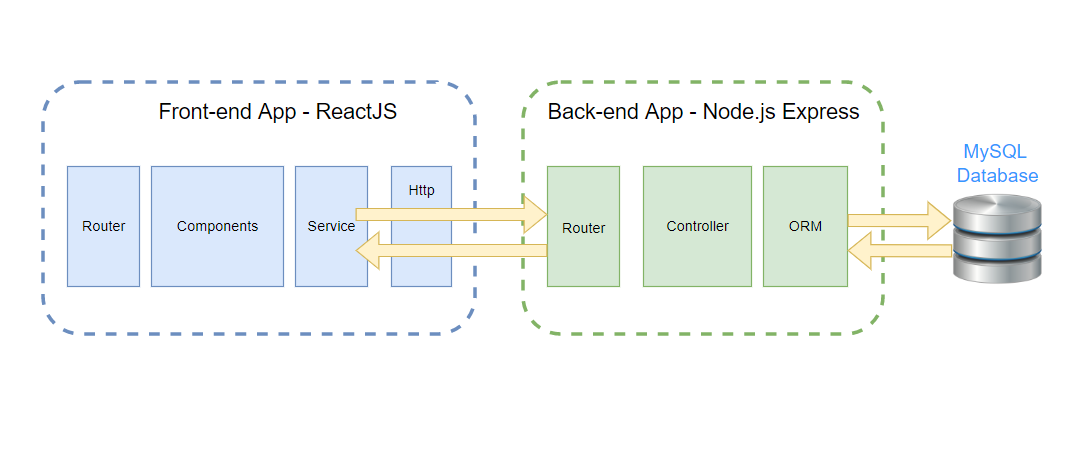
**SRS 12** – Conectarea utilizatorilor la fereastra de editare colaborativă se va face prin accesarea adresei URL a ferestrei prin intermediul browser-ului.

**SRS 13** – Interacțiunea simultană a utilizatorilor asupra documentelor de tip text va fi evidenţiată prin metode specifice.

**SRS 14** – Fişierele de tip text, editate la nivelul sesiunii colaborative pot fi salvate şi accesate oricând din spaţiul de stocare de către membrii care au permisiuni asupra folderului în care este salvat.

## Arhitectura sistemului

Soluţia dezvoltată este formată din aplicaţia front-end, reprezentată în partea stângă a figurii 4.1 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.



Figură 4.1 Reprezentarea arhitecturii sistemului software

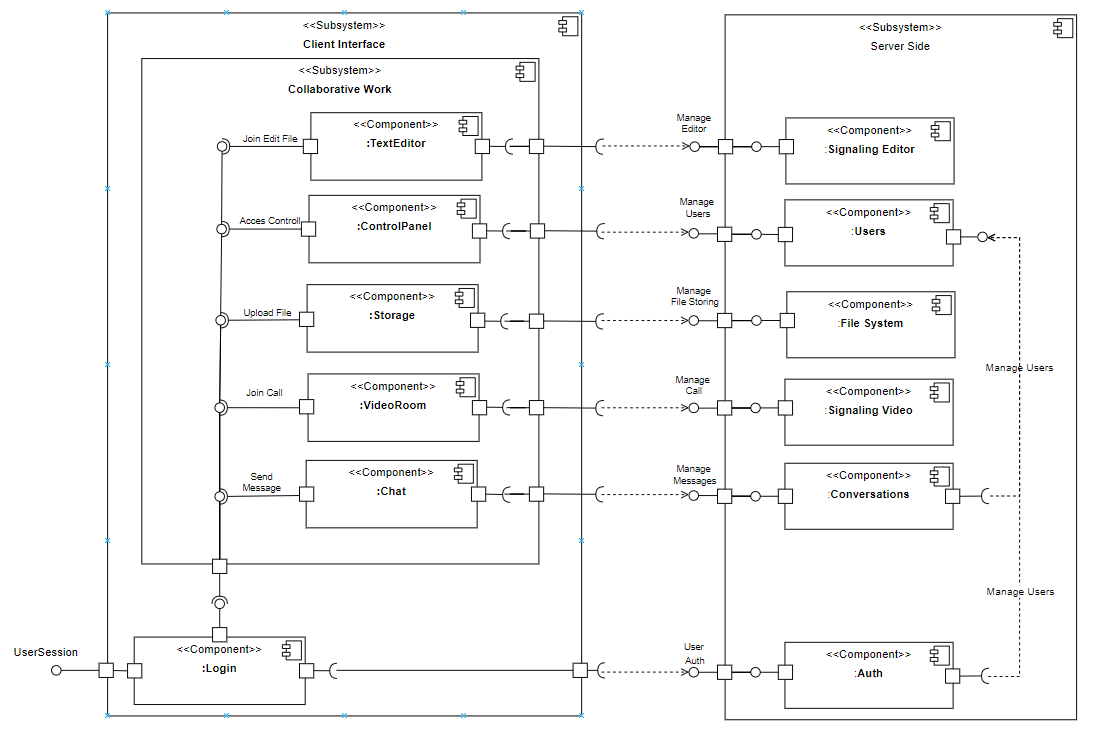
Pentru intermedierea accesului dintre utilizator şi interfaţă voi folosi un server web Nginx care controlează accesul utilizatorilor prin solicitarea certificatului de utilizator la momentul accesării paginii. Astfel, accesul va fi permis doar pentru accesul la interfaţa web, iar serverul Nginx având 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 4.2.

Principalele componente ale aplicaţiei sunt modulele:

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



Figură 4.2 Diagrama de componente a sistemului

***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.

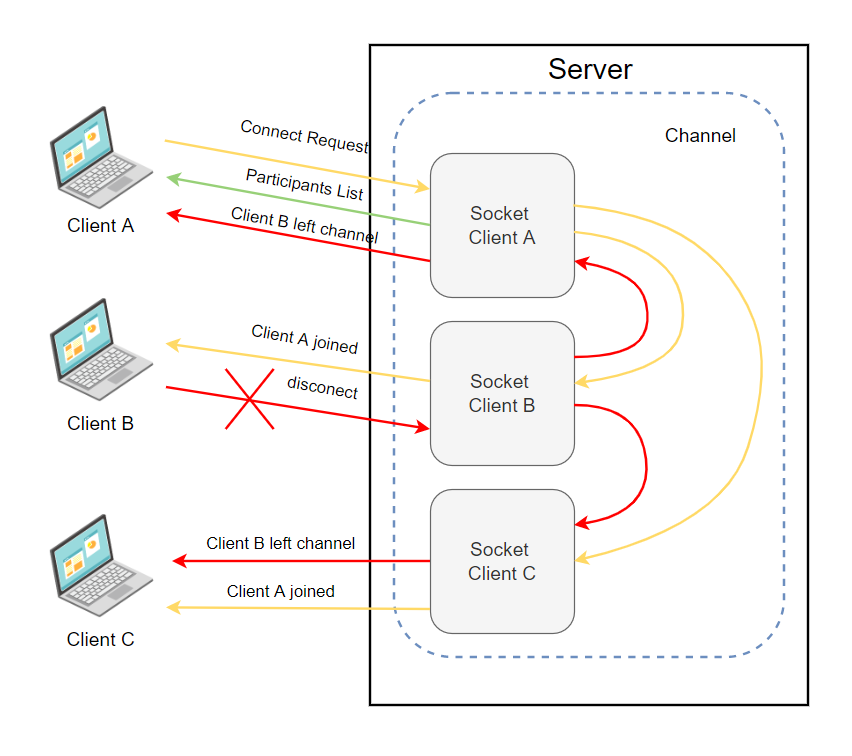
***ControlPanel*** poate fi accesat doar de utilizatorii cu rol de administrator. Această pagină reprezintă un panou unde doar utilizatorii cu rol de administrator pot filtra restul utilizatorilor în funcţie de grupurile în care sunt înrolaţi şi restricţiona accesul.

***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 4.2 sunt reprezente componentele aplicaţiei server. Back-end-ul aplicației este format din două servicii API de tratare al request-urilor care asigură partea de procesare 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.

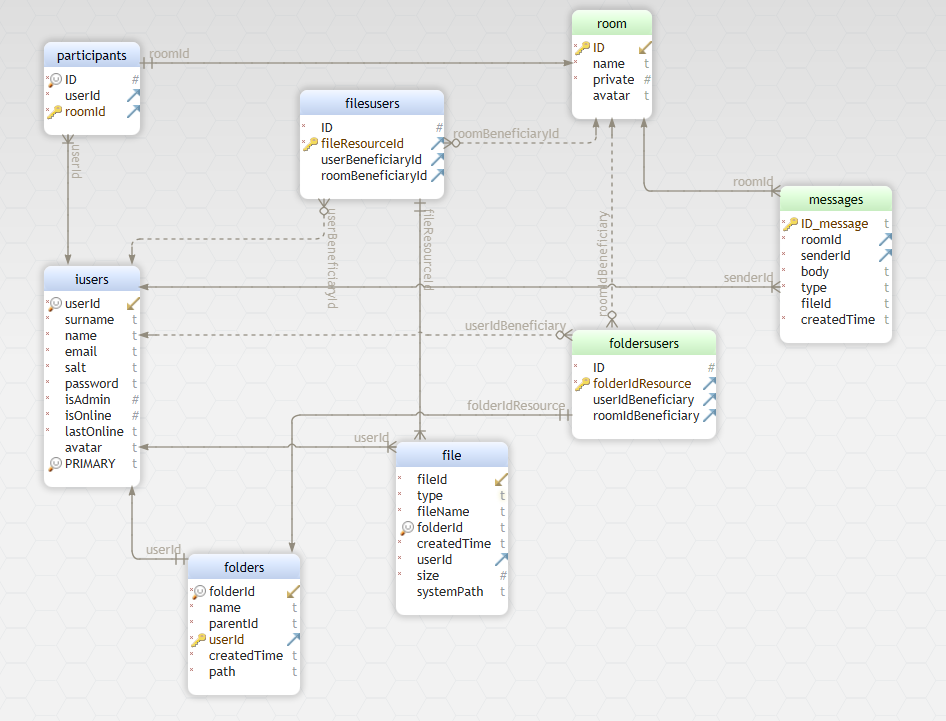


Figură 4.3 Fluxul de date în cadrul unui canal de comunicaţie Socket.IO

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. 4.3. 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.

### Structura bazei de date

Pentru stocarea datelor am ales utilizarea unei baze de date relaţionale de tip MySQL, deoarece pe baza relaţiilor ce se stabilesc între tabele, datele pot fi organizate în mod intuitiv, 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 prin intermediul constrângerilor de tip cheie-valoare.



Figură 4.4 Structura bazei de date

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-ul grupului, respectiv al utilizatorului. Astfel, în cazul unei conversaţii private vor exista două intrări diferite 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 cu valoarea *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 *v16.11.0*, asistat de managerul de pachete NPM *v8.1.4* şi alte biblioteci, respectiv 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.

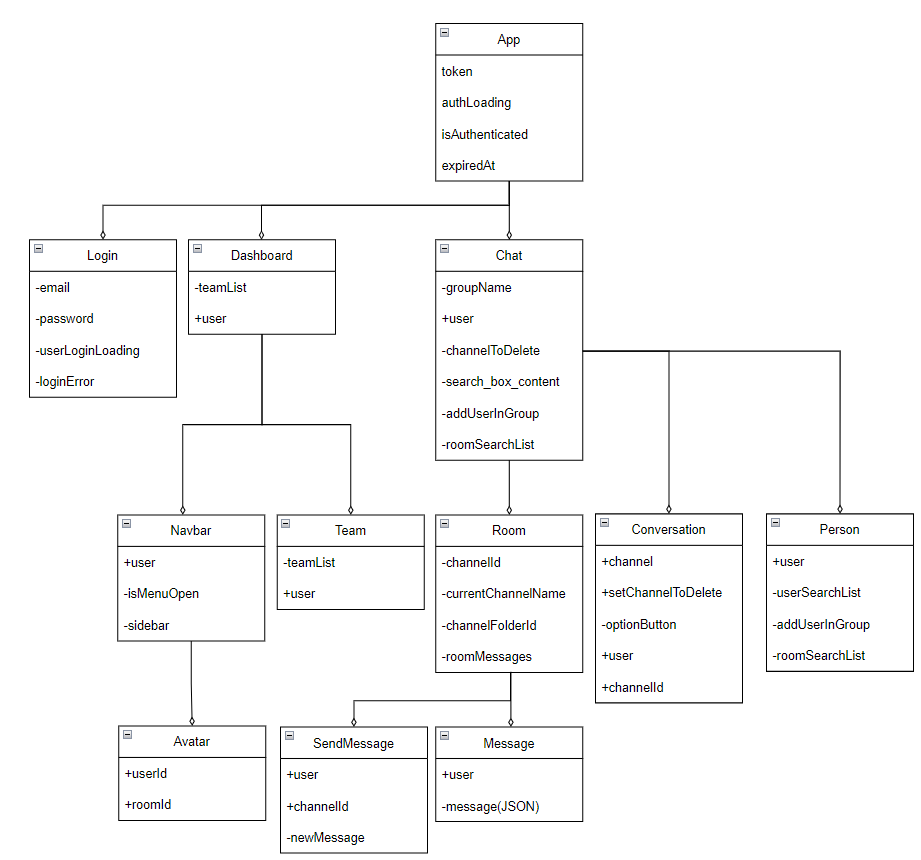
## Aplicaţia Web Client

Componentele permit divizarea interfeţei utilizatorului în unităţi izolate, independente şi reutilizabile. O componentă funcțională React se comportă ca o funcție JavaScript prin acceptarea de argumente sub formă de *props* și returnarea rezultatului sub formă de elemente React.

### Definirea Componentelor

Pentru implementarea software a aplicaţiei Client am alcătuit diagrama componentelor funcţionale React, reprezentată în figurile Fig. 5.1, 5.3 şi Fig. 5.4. Funcţiile folosite pentru crearea de request-urilor asupra server-ului se află în directorul *asyncActions*.

Principala componentă a aplicaţiei este ***App***, aceasta este prima care este redată atunci când se accesează aplicaţia. Principalul scop al acestei componente este de a redirecţiona utilizatorul spre pagina corespunzătoare link-ului URL generat de acţiunile acestuia, folosind *BrowserRouter*, o bibliotecă de rutare standard de tip Javascript. La nivelul acestei pagini se realizează şi requestul de verificare al token-ului de acces pentru utilizator. Verificarea se realizează în momentul expirării token-ului sau atunci când pagina suferă un refresh la nivelul browser-ului prin intermediul funcţiei *verifyTokenAsync()*. În cazul în care token-ul prezent în browser nu este validat de Server, atunci utilizatorul este deconectat de la sesiune şi se revine la pagina de Login. Dacă token-ul este validat, atunci acesta este înlocuit cu unul nou şi salvat în Redux până la următoarea verificare. De menţionat că această verificare se produce la fiecare request al Clientului către server pentru a preveni atacurile de tip Cross-Site Request Forgery.



Figură 5.1 Diagrama obiectelor de tip React de la nivelul aplicaţiei client – Partea 1

Paginilie ce pot fi accesate de către utilizator sunt reprezentate de componentele:

* *Login*;
* *Dashboard*;
* *Chat*;
* *TextEditor*;
* *VideoRoom*;
* *ControlPanel;*
* *Storage;*
* *Support*.

Componenta ***Dashboard*** afişează toate conversaţiile publice ale utilizatorului, oferind posibilitatea de navigare spre acestea. Funcţia folosită în cadrul metodei pentru afişarea grupurilor este *getSearchRoomService()*, care întoarce lista grupurilor în format JSON*.*

Pagina folosită pentru gestionarea mesajelor are ca suport componenta ***Chat***, care este alcătuită din trei componente: *Room*, *Conversation* şi *Person*. Funcţiile principale de interacţiune cu aplicaţia server de la nivelul acestei componente sunt:

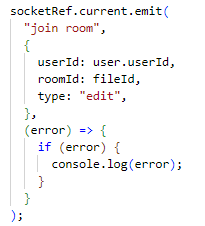
* *userSetRoomListAsync()* – întoarce lista sortată a grupurilor ale căror nume conţine textul introdus în caseta de căutare;
* *userSearchPersonListAsyn()* – primeşte ca răspuns sub formă de listă persoanele ale căror nume conţine textul introdus în caseta de căutare;
* *CreateNewGroup*() – are ca funcţionalitate creare unui grup nou cu numele transmis ca parametru.

Modificările aduse stării din Redux sunt procesate folosind funcţiile:

* *setUserSearchBoxContent()* – salvează valoarea keyword-ului de căutare în reducer-ul *chatRedu*.
* *UpdateAddUserInGroup()* – reţine *ID*-ul personei ce urmează să fie adăugate în grup. Dacă nu există nici o persoană selectată atunci va fi iniţializat cu un şir gol;
* *setPersonSearchList()* – salvează lista cu persoanele căutate de către utilizator.

***Room***conţine lista cu mesaje corespondente fiecărei conversaţii şi mecanismul de trimitere de mesaje noi. Pentru actualizarea mesajelor de la nivelul conversaţiei a fost utilizată funcţia *getMessageListTime()*, cu rolul de a întoarce lista parţială a mesajelor în funcţie de un reper temporal şi poziţia faţă de acesta. Scopul dezvoltării acestei funcţii este actualizarea etapizată a listei de mesaje la evenimentul de scroll executat de utilizator, evitând astfel transmiterea întregului flux de date al mesajelor.

Pagina cu rol de editare colaborativă are la bază componenta ***TextEditor***, ce permite afişarea utilizatorilor care participă la editare şi interacţiunea dintre aceştia. Documentul editat este salvat într-o formă persistentă prin utilizarea funcţiei *saveTextFileAsync(),* ce primeşte ca argumente datele utilizatorului şi conţinutul casetei de editorului. Restul căilor de comunicaţie cu partea de back-end sunt asigurate prin intermediul bibliotecii Socket.IO. Pentru conectarea la grupul specific documentului se transmite evenimentul „*join room*” cu transmisia request-ului ce conţine userId-ul, roomId-ul şi tipul grupului.



Figură 5.2 Trimiterea request-ului de conectare la sesiunea de editare colaborativă

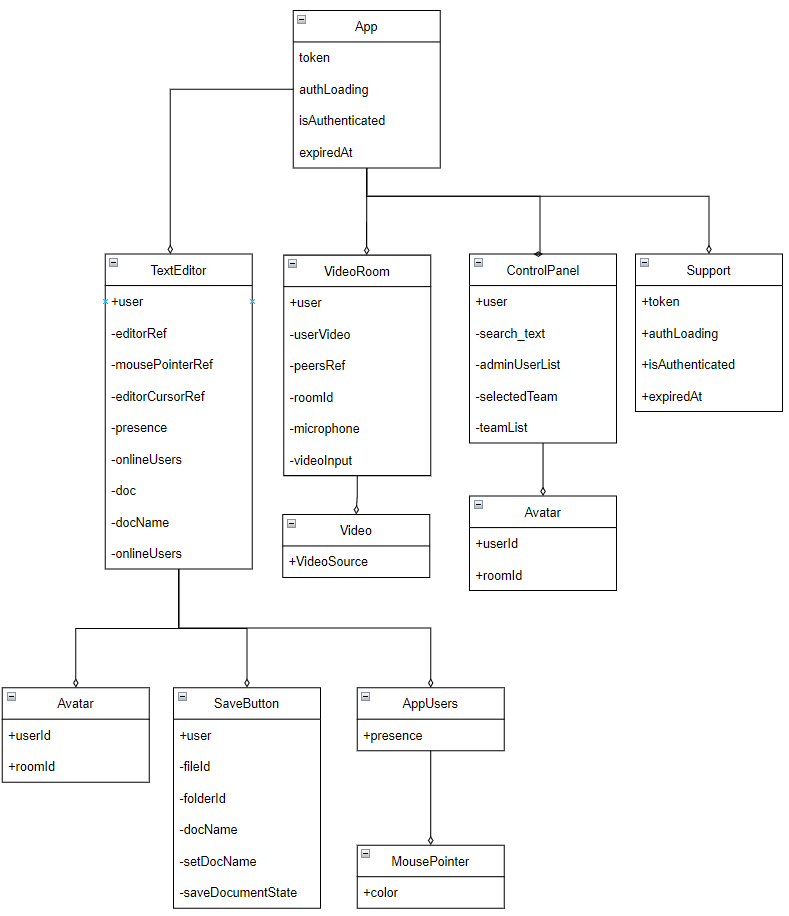
În urma validării făcute de server (verificarea identităţii utilizatorului), este trimisă lista cu utilizatorii deja conectaţi la grupul respectiv prin tratarea evenimentului „*all users edit*”. Transmiterea datelor către server şi invers are loc în aceeaşi manieră prin folosirea metodelor *socketRef.current.emit() şi socketRef.current*.on(), unde *socketRef* reprezintă referinţa spre socket-ul deschis pentru comunicaţie. Extragerea şi formatarea textului a fost realizată prin folosirea metodelor specifice utilitarului Quill:

* *editorRef.current.on(“text-change”, (delta, oldDelta, source))*;
* *editorRef.current.on(“selection-change”,( range, oldRange, source))*;
* *editorRef.current.updateContents()*, unde editorRef.current este referinţa către obiectul de tip Quill.

Evenimentul „text-change” detectează formatarea textului, şi returnează noul format al textului prin variabila delta, iar parametrul *source reprezintă sursa de la care a venit modificarea*.

***AppUsers***este folosită pentru afişarea cursoarelor respectivilor utilizatori pe baza coordonatelor absolute preluate de la nivelul containerului în care este definit editorul text, din componenta *TextEditor*.

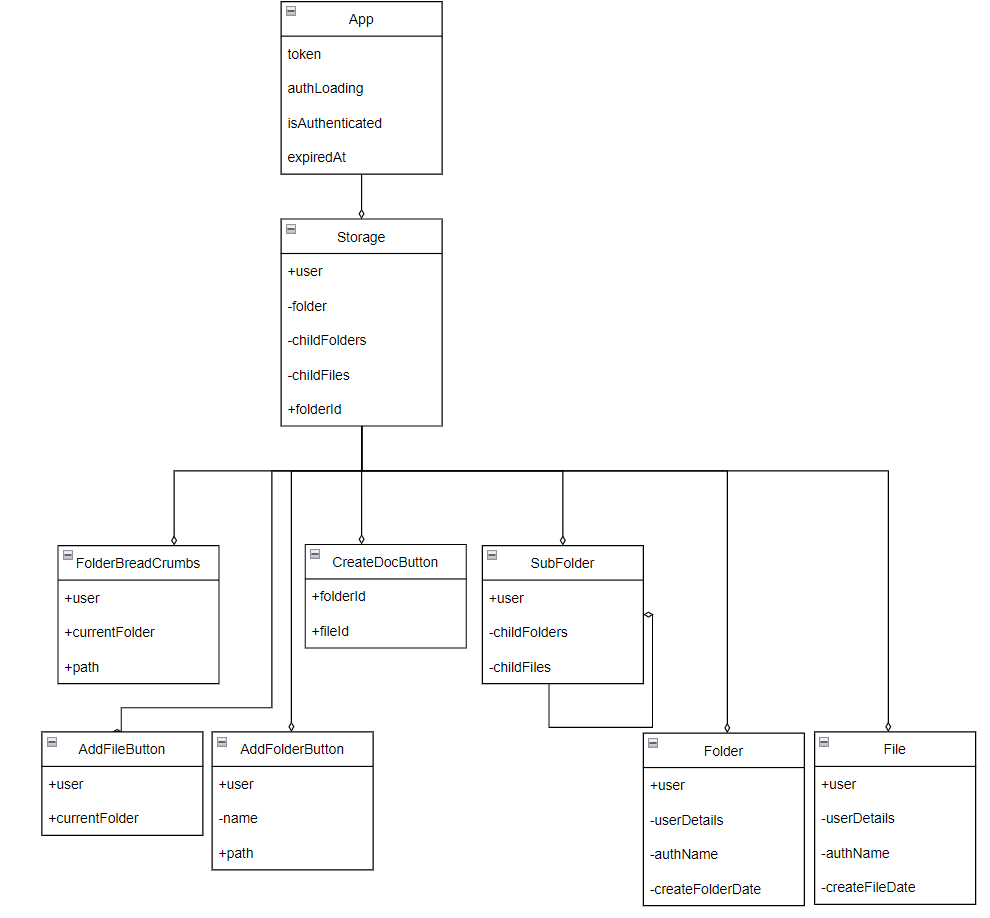
Componenta ***Support*** are doar rol informativ pentru accesarea repository-ului de Github unde se poate vizualiza codul sursă al proiectului.



Figură 5.3 Diagrama obiectelor de tip React de la nivelul aplicaţiei client – Partea 2

Componenta ***VideoRoom*** poate fi accesată din cadrul ferestrei de ***Chat***, dar şi prin simpla accesarea a adresei URL generată anterior de către alt utilizator. La nivelul acestei componente utilizatorul solicită conectarea la canalul grupului de tip video prin intermediul conexiunii asigurate de către biblioteca Socket.IO. După conectare va primii o listă cu utilizatorii înrolaţi deja în convorbire şi va încerca să stabilească conexiuni de tip peer-to-peer cu aceştia pentru transmiterea fluxului video-audio.

Spaţiul de administrare al fişierelor este cuprins de componenta ***Storage***. Lista de fişierelor şi a folderelor sunt actualizate prin utilizatea funcţiilor: *userSetFolderList()*, respectiv *userSetFileList()*. Principalele funcţionalităţi de la nivelul paginii cu acelaşi nume sunt încărcarea de fişiere şi crearea de foldere, asigurate de componentele ***AddFilebutton***, respectiv ***AddFolderButton***. Fişierele şi folderele sunt definite de către componentele ***Folder***, respectiv ***Files***. Navigarea la nivelul folderelor se realizează prin intermediul componentei ***FolderBreadCrumbs***, care afişeză calea curentă pentru folderul curent şi permite accesarea tuturor folderelor precedente prin accesare directă.



Figură 5.4 Diagrama obiectelor de tip React de la nivelul aplicaţiei client – Partea 3

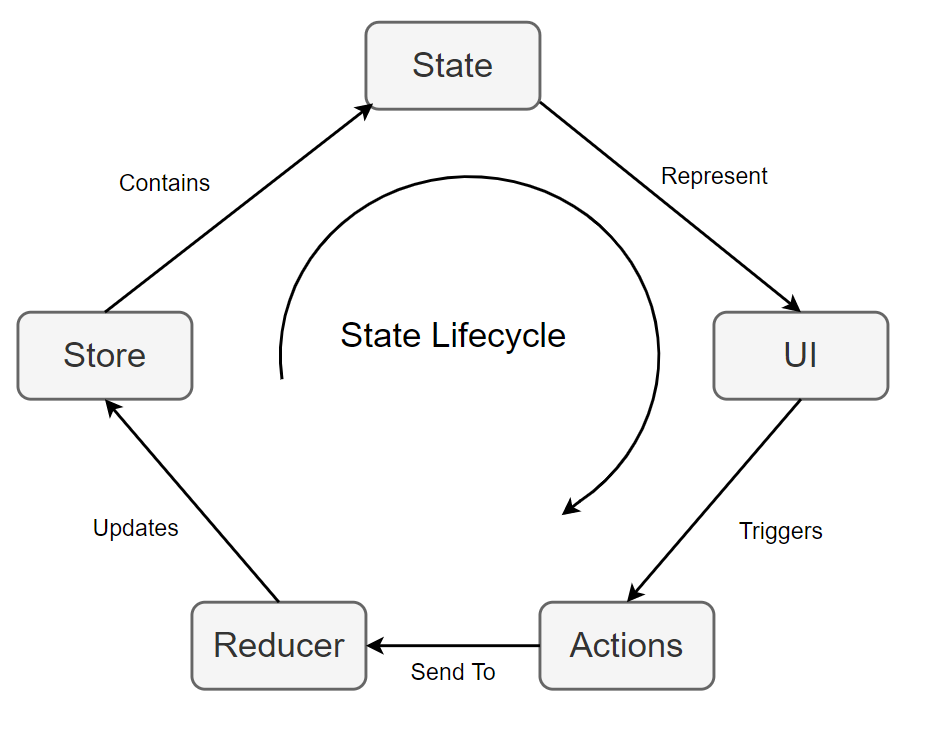
De asemenea, pentru vizualizarea structurii spaţiului de stocare este folosită componenta ***SubFolder***. Aceasta primeşte ca parametru *ID-ul* unui folder şi returnează toate subfolderele şi fişierele pe care le conţine, iar apoi pentru fiecare subfolder este redată din nou componenta ***Subfolder***. Procesul este terminat atunci când subfolderele terminale nu mai conţin alte subfoldere sau fişiere.

Navigarea la nivelul paginilor se face prin intermediul componentei ***Navbar***. Aceasta conţine un sidebar cu toate paginile ce pot fi accesate şi un al meniu de acţiuni rapide dedicate utilizatorului.

***ControlPanel*** este o componentă cu rol de control asupra conturilor utilizatorilor şi poate fi accesată doar de administratori. Aceasta oferă mecanisme de căutare, filtrare, editare şi stergere a conturilor de utilizator.

### React-Redux

Fluxul datelor la nivelul componentelor este unidirecţional, începând de la componentele superioare spre componentele inferioare. Astfel, nu pot fi transmise date între componentele aflate pe acelaşi nivel, ci doar dacă acestea sunt transmise prin intermediul unei componente aflate pe un nivel superior. Pentru rezolvarea acestui inconvenient am folosit librăria React Redux, ce permite stocarea variabilelor într-un mediu de stocare ,Store , accesibil de către orice componentă a aplicaţiei. Atunci când se doreşte modificarea unei variabile din Store, se foloseşte modulul Reducer. La fiecare modificare a variabilelor din Store, componentele aplicaţiei sunt redate, aplicând astfel noile modificări. Acest proces este exemplificat în figura Fig. 5.5.



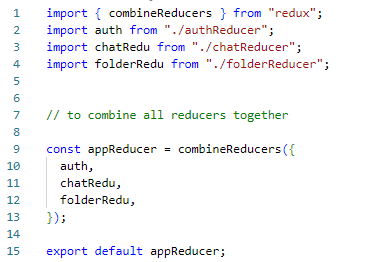
Figură 5.5 Principiu de funcţionare al sistemului React Redux

În implementarea aplicaţiei am ales să divizez componenta de Reducer în trei subcomponente diferite pentru dezvoltarea modulară a proiectului:

* *Auth*;
* *ChatRedu*;
* *FolderRedu*;

**Auth** are rolul de a menţine starea datelor utilizatorului precum: token-ul de sesiune, timpul de valabilitatea al token-ului, datele personale ale utilizatorului şi dacă acesta este autentificat sau nu.

**ChatRedu** menţine starea variabilelor corespunzătoare componentelor legate de mesageria text precum: *ID*-ul conversaţiei deschise, numele conversaţiei deschise, lista mesajelor afişate în conversaţie, *ID*-ul folderului corespondent conversaţiei, textul după care se face căutarea participanţilor şi lista cu conversaţiile deschise ale utilizatorului.



Figură 5.6 Componentele de tip Reducer de la nivelul aplucaţiei client

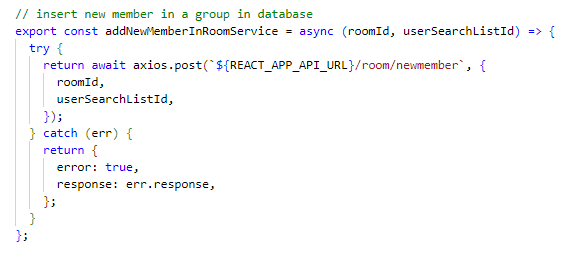
**FolderRedu** face referire la starea variabilelor legate de sistemul de stocare al fişierelor. Acesta reţine starea variabilelor precum: ID-ul, numele, şi calea folderului spre care a navigat utilizatorul, lista subfolderelor şi a subfişierelor corespunzătoare folderului curent în care se află utilizatorul.

### Definirea Request-urilor

Pentru realizarea request-urilor de comunicaţie cu partea de back-end s-a folosit biblioteca *Axios*. Cele două sisteme rulează pe aceeaşi maşină fizică, astfel comunicaţia dintre ele se poate realiza folosind request-uri de tip HTTP. Funcţiile folosite pentru efectuarea request-urilor sunt de tip asincron, întrucât este necesar aşteptarea procesării cererii şi răspunsul serverului. Am definit patru rute principale prin care cele două sisteme realizează comunicaţie de tip request:

* „*/users*” – efectuează toate cererile ce ţin de utilizator şi acţiunile sale (cum ar fi autentificarea, verificarea token-ului sau interacţiunea cu gestionarea mesageriei;
* *„/room”*- canalizează toate cererile specifice conversaţiilor şi grupurilor, aducerea din memorie a mesajelor sau a listelor cu participanţi;
* *„/document”* – specilizat pe request-urile ce ţin de previzualizarea imaginilor, accesarea fişierelor, sau descărcarea de documente;
* *„/folder”* – conţine cererile ce ţin de sistemul de fişiere, exemplu fiind returnarea listei cu subfoldere sau fişiere, respectiv crearea de foldere noi.

Răspunsul cererii poate fi trimis într-un format personalizat, iar pentru semnalarea erorilor se pot seta coduri de eroare care să fie interpretate de către aplicaţia Client. Aceste coduri de eroare reprezintă şi un foarte bun mecanism de tratare al erorilor în cazul inconsistenţei răspunsului.



Figură 5.7 Exemplu request de tip post pentru adăugare unui utilizator nou în grup

## Aplicaţia web server

### Tratarea request-urilor

Aplicaţia server răspunde request-urilor aplicaţiei client prin intermediul serverului de tip HTTP şi a serverului de tip socket.io. Organizarea rutelor pentru request-urile HTTP a fost făcută prin intermediul utilitarului *Router()* din cadrul framework-ului express(), rezultând fişierul *routes* cu fişierele: *users.js, room.js, folder.js* şi *file.js*. În aceste fişiere se răspunde fiecărui request specific cu funcţia corespondentă din folderul *controllers*. Înainte de prelucrare, se execută funcţia middleware de verificare a JWT.



Figură 5.8 Folosirea funcţiei middleware pentru verificarea autenticităţii request-ului

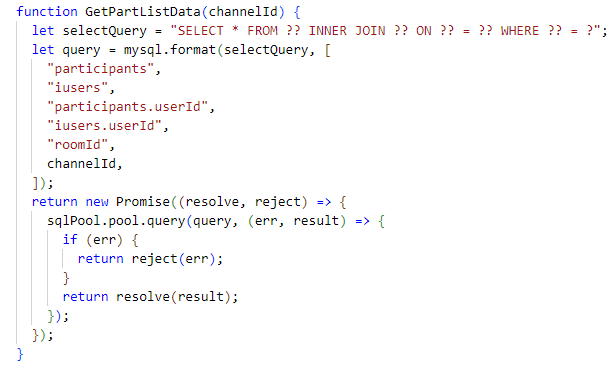
Serverul pentru comunicaţia webSocket este deschis pentru conectare prin utilizarea metodei *„.on(“connection”, (socket) => {})”*. În cadrul parantezelor sunt definite tipurile de request la care răspunde serverul prin metode de tipul: *socket.on(“join room”, async (request))*. Tipurile de request definite sunt:

* „*join room*” – clientul solicită conectarea pentru un grup de tipul chat, video sau edit. Dacă respectă criterile de validare atunci acesta este adăugat în listă şi socketul său asociat este conectat la canalul respectiv de comunicaţie;
* „*send chat message*” – clientul trimite un mesaj în cadrul mesageriei. Dacă procesul este validat, mesajul este redirecţienat spre restul membrilor din grup şi apoi este stocat la nivelul bazei de date;
* „*typing chat message*” – clientul semnalizează activitatea din cadrul casetei de editare pentru trimitere a mesajelor;
* „*send doc edit*” – clientul trimite modificările efectuate asupra documentului, care sunt trimise printr-un mesaj de tip broadcast către toţi clienţii conectaţi în cadrul canalului la momentul respectiv;
* „*send doc pointer*” – clientul trimite coordonatele cursorului de editare către restul participanţiilor din cadrul editării colaborative;
* „*send doc presence*” – clientul trimite coordonatele pointer-ului cu scopul de a semnala interacţiunea la nivelul ferestrei de interacţiune;
* „*sending signal*” – utilizatorii ce au intrat în cadrul unei convorbiri de tip video efectuează cereri către restul participanţilor pentru a stabilii conexiuni de tip peer-to-peer;
* „*returning signal*” – reprezintă răspunsul de acceptare al conexiunii peer-to-peer de către participanţi;
* „*disconnect*” – acest tip de eveniment este declanşat atunci când socket clientului este deconectat. Tratarea unui astfel de eveniment este foarte importantă, deoarece utilizatorul corespunzător socketului trebuie eliminat din lista canalelor de comunicaţii.

### Baza de date MySQL

Interacţiunea cu baza de date se realizează prin intermediul funcţiilor din cadrul folderului *services*. Acestea sunt clasificate după modelul de rutare în categoriile: *Files*, *Folder*, *Room* şi User. Pentru conectarea la baza de date am folosit metoda *createPool()* din cadrul bibliotecii mysql, asociate NodeJS. *CreatePool()* creează un spaţiu de stocare al conexiunilor cu baza de date, iar atunci când este solicitată o conexiune dintr-un obiect de tip *pool()*, se va primi ca răspuns o conexiune liberă. Când limita de conexiuni este atinsă, se va aștepta eliberarea unei conexiuni pentru execuția interogării SQL.

Prevenirea atacurilor de tip SQL injection se face prin utilizarea metodei *mysql.format()*, ce are rol de sanitizare a string-urilor de tip query. Un astfel de exemple este redat în Fig. 5.9, interogare ce are rolul de a determina lista participanților din cadrul unei conversații.



Figură 5.9 Interogare SQL pentru returnarea listei de participanţi din cadrul unui grup

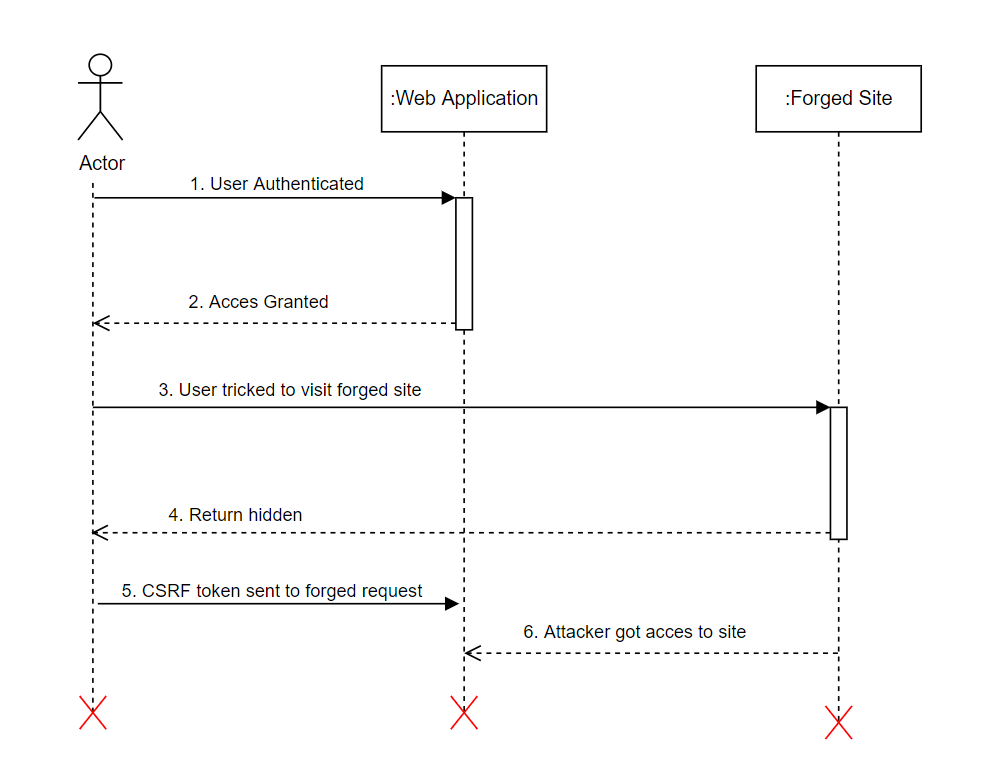
## Metode de securizare

### Protecţie împotriva atacurilor CSRF

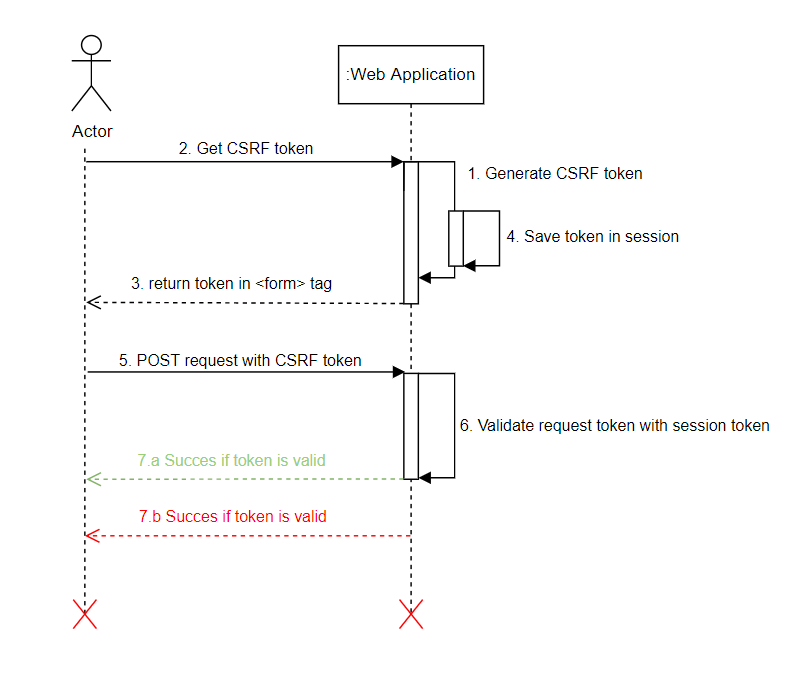
Atacurile de Tip Cross-Site Request Forgery constau în furtul credenţialelor de sesiune ale utilizatorilor autentificaţi şi utilizarea lor pentru efectuarea de acţiuni în cadrul site-ului web. Pentru ca un astfel de atac să fie realizat cu succes, site-ul web trebuie să identifice utilizatorii prin intermediul cookie-urilor, iar utilizatorii să acceseze link-ul plasat de către atacator. Pentru prevenirea unor astfel tipuri de atacuri am implementat un sistem bazat pe token-uri de acces Json Web Token.

Sistemul funcţionează în felul următor: în momentul autentificării utilizatorului este generat un obiect de tip JWT cu numele *refreshToken*, un *accessToken*. *RefreshToken* este folosit pentru pentru a permite utilizatorului să genereze noi token-uri de tip *accessToken*, cât timp acesta este valid. Odată cu *accessToken* este generat şi un timp de expirare al acestuia ce este transmis utilizatorului, iar atunci când expiră clientul va face un request pentru generarea unui nou token de tip *accessToken*. Dacă *refreshToken*-ul a expirat atunci sesiunea utilizatorului va fi oprită şi se va solicita autentificarea (*refreshToken*-ul are un timp mai mare).

În momentul când se efectuează un request, token-ul de tip *accessToken* va fi transmis în antetul requestului şi va fi verificat prin intermediul unei funcţii de tip middleware. Această funcţie se execută înainte de procesarea requestului, iar dacă *accessToken*-ul prezent în antetul request-ului nu este validat se răspunde cu un mesaj de eroare.



Figură 5.10 Diagramă atac de tip CSRF

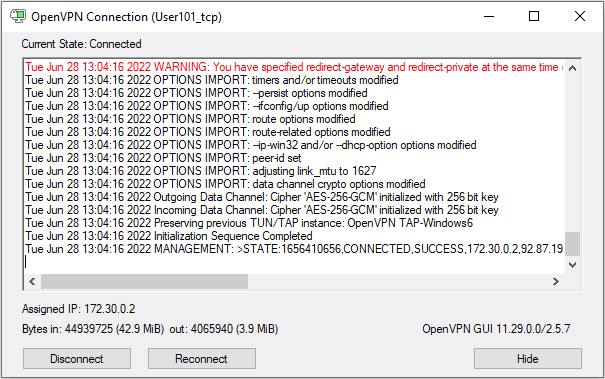


Figură 5.11 Securizarea sesiunii folosind JWT token

### OpenVPN

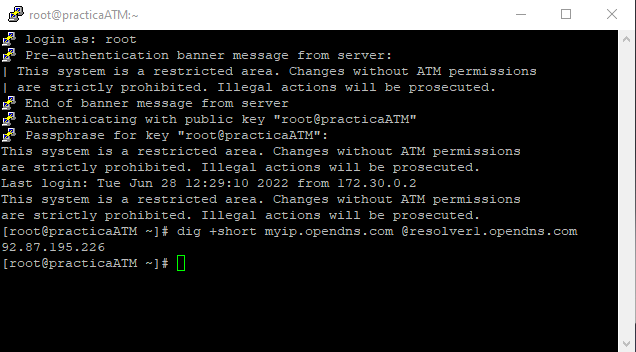
OpenVPN este un proiect open-source ce permite crearea de reţele tip VPN. Acest tip de reţea permite comunicaţia prin utilizarea conexiunilor securizate bazate pe protocoale SSL/TSL.

Pentru găzduirea sistemului am configurat un server fizic din infrastructura locală pentru instalarea şi rularea sistemului software dezvoltat. Pentru securizarea accesului asupra infrastructurii a fost necesară instalarea serviciului OpenVPN şi crearea de profile specifice destinate utilizatorilor aplicaţiei. Pentru configurare a fost necesară generarea datelor pentru autoritatea centrală (cheia privată a CA-ului, respectiv certificatul), iar pentru clienţi au fost generate 20 de profile de VPN pe baza configurării serverului de VPN. Pentru conectarea, utilizatorii sunt nevoiţi să se conecteze la serviciul de VPN folosind profilul asignat prin utilizarea aplicaţiei OpenVPN. În figura 5.12 se poate observa conectarea utilizatorului *User101\_tcp* la reţeaua de VPN. În cadrul reţelei utilizatorul are atribuită adresa *172.30.0.2*.



Figură 5.12 Rularea serviciului la nivelul utilizatorului pentru accesarea aplicaţiei

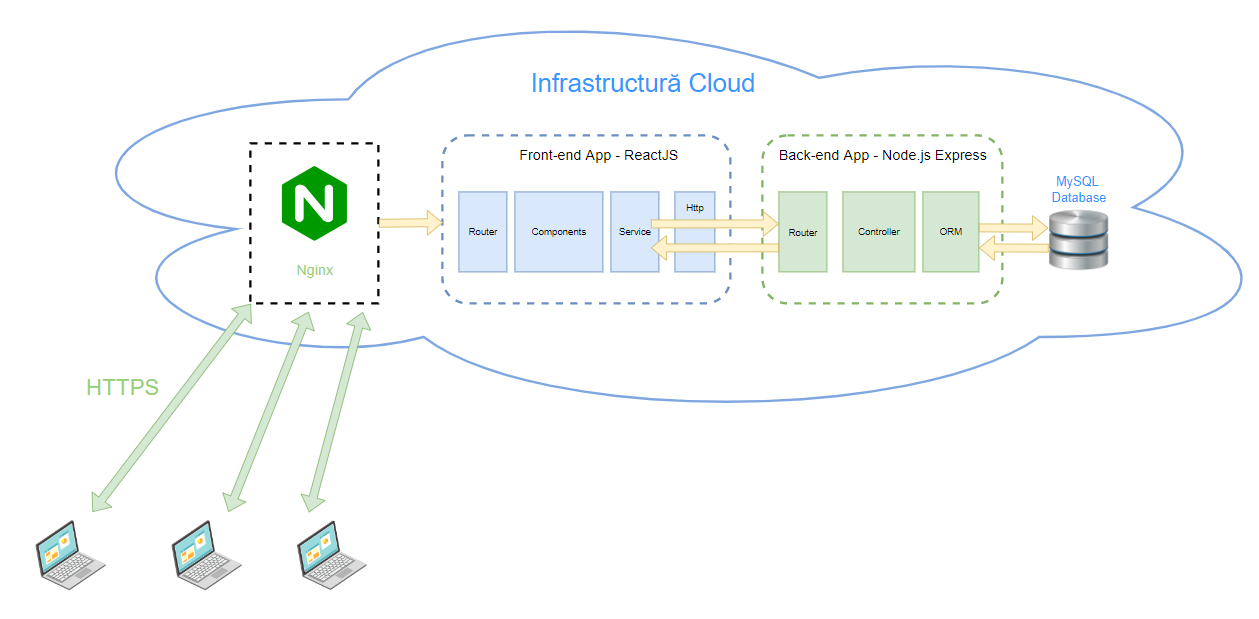
Adresa IP publică a clientului conectat prin VPN este *92.87.195.226* (adresa publică fără serviciul de VPN este *37.251.223.233*), iar aceasta corespunde cu adresa IP a serverului de VPN. În figura 5.13 este redată conexiunea asupra serverului prin intermediul utilitarului *Putty*.



Figură 5.13 Sesiune de administrare a serverului folosind conexiune SSH

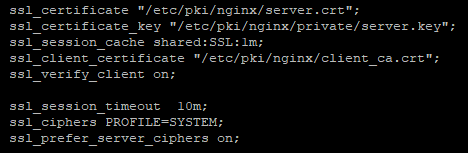
### Autentificare mutuală

Accesul la aplicație este intermediat prin utilizarea unui server Nginx. Acesta are rolul de a controla comunicația dintre utilizator și aplicația client prin realizarea unei conexiuni securizate de tip HTTPS.



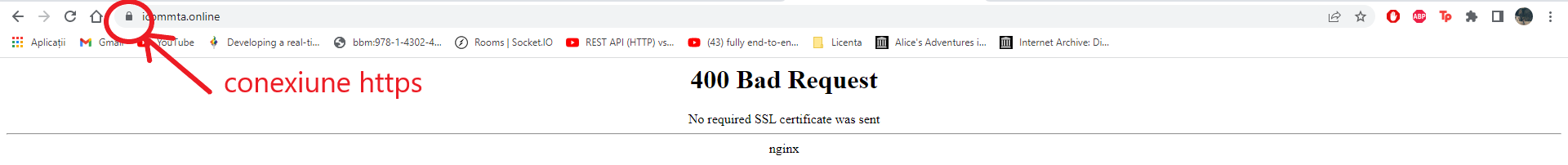
Figură 5.14 Mod de intercacţiune folosind comunicaţia securizată

Serverul Nginx redirecţionează comunicația de pe portul extern 443 catre endpoint-ul aplicației client ( acesta fiind *127.0.0.1:3000*). De asemenea pentru acceptul comunicației clientul trebuie să prezinte un certificat semnat cu credențialele serverului Nginx. Pentru configurarea acestei setări trebuie completate următoarele intrări din fișierul „*nginx.conf*”:



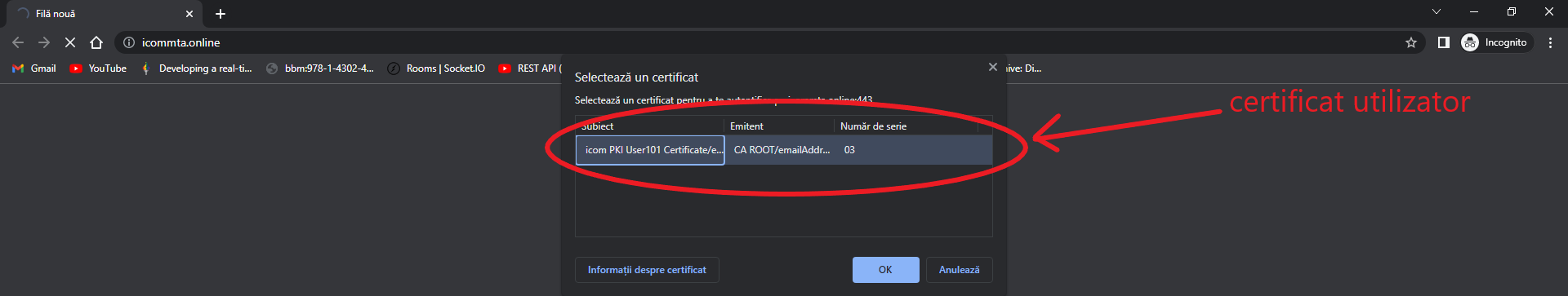
Figură 5.15 Fişierul de configurare al serverul de Nginx

În momentul accesării site-ului, browser-ul solicită utilizatorului să aleagă un certificat valid pentru accesarea paginii.



Figură 5.16 Accesarea aplicaţiei fără un deţinerea certificatului de utilizator

Certificatul trebuie importat în browser pentru a putea fi selectat, în caz contrar utilizatorul nu va putea realiza autentificarea mutuală.

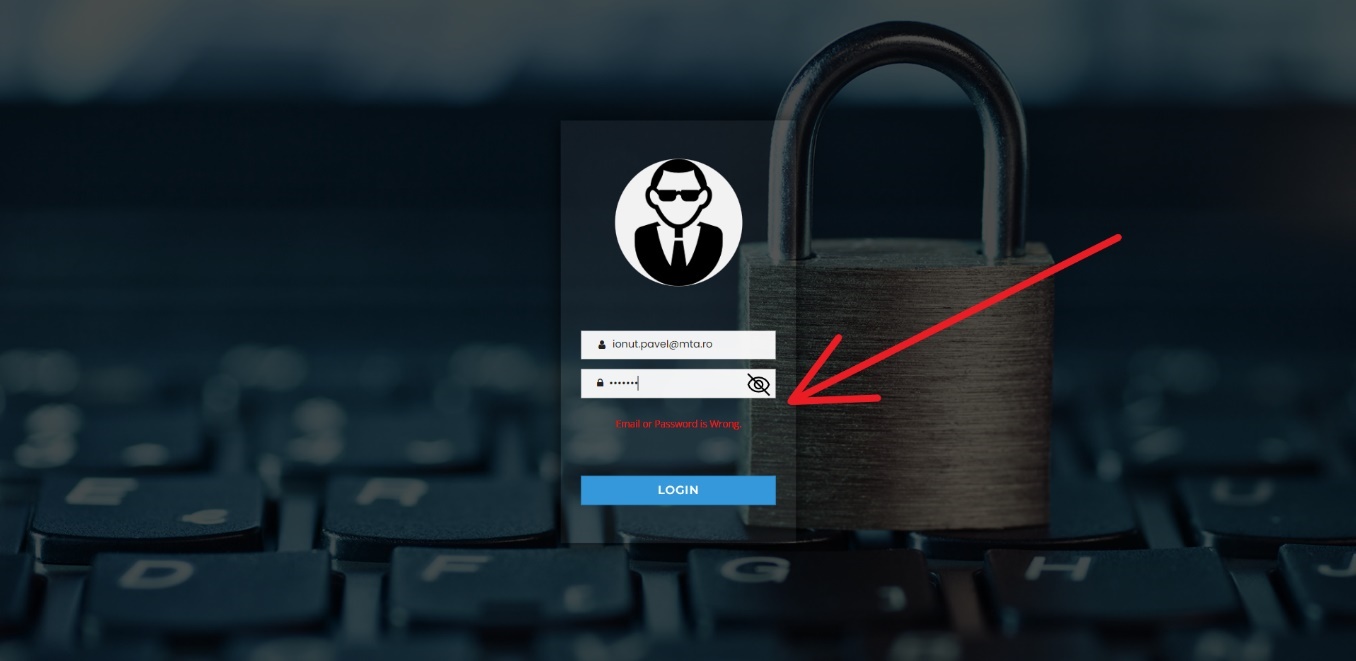


Figură 5.17 Solicitarea certificatului de utilizator la nivelul browser-ului

# Testarea sistemului

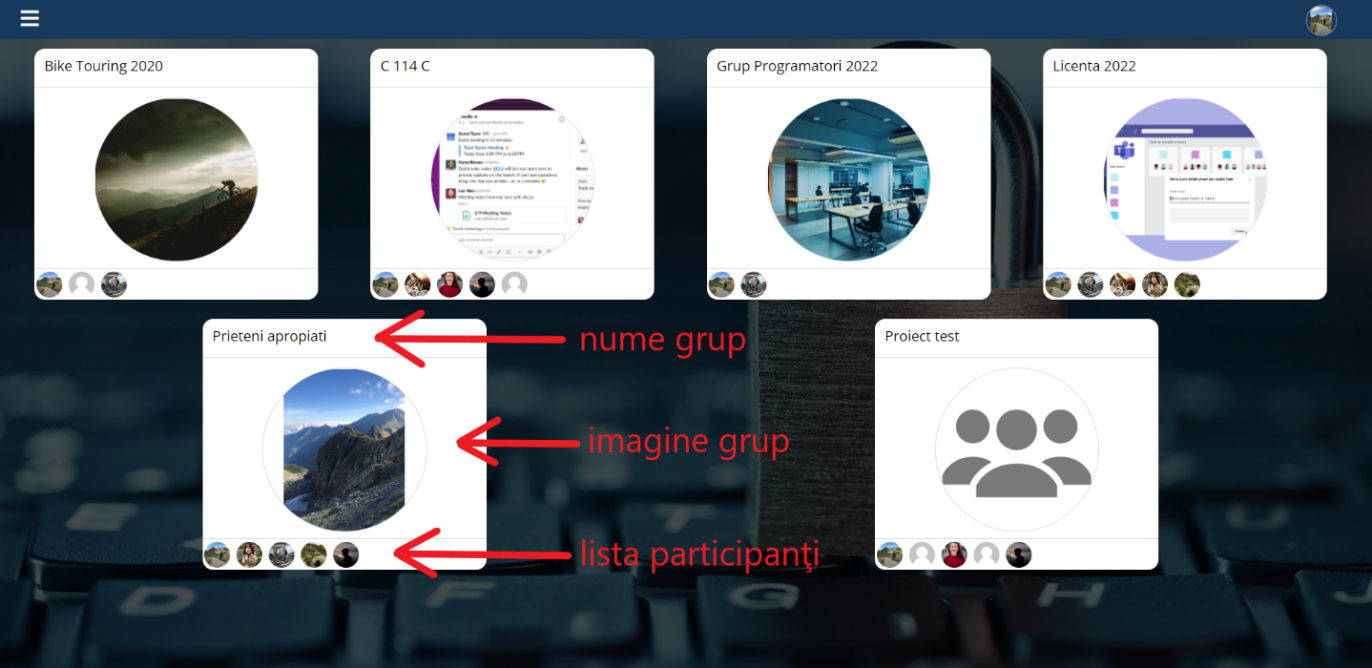
## Utilizarea aplicaţiei

Design-ul aplicaţiei este unul intuitiv şi uşor de folosit, asemănându-se cu alte aplicaţii web. Pentru autentificare utilizator este nevoit să-şi introducă credenţialele contului de utilizator, mai exact adresa de email şi parola atribuită contului. În cazul în care utilizatorul introduce greşit credenţialele acesta va primi un mesaje de notificare şi va fi nevoit să completeze din nou câmpurile. Pe lângă acest pas, este necesar ca utilizatorul să aibe importat în browser certificatul personal şi să fie conectat la serviciul de VPN prin folosirea profilului specific. În caz contrar acesta nu va putea accesa platforma.



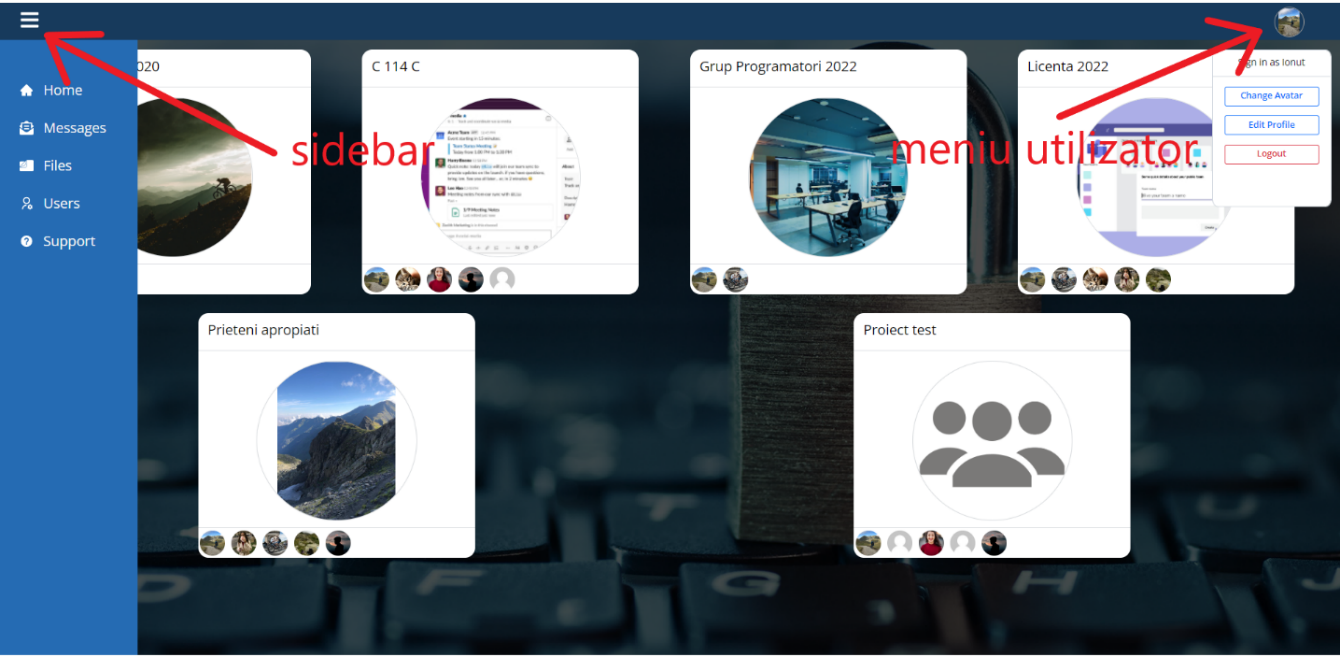
Figură 6.1 Pagina de Login a aplicaţiei

După realizarea autentificării utilizatorul este redirecţionat spre *homepage*, unde sunt listate toate grupurile publice (nu şi conversaţiile private) din care face parte, împreună cu lista de utilizatori corespunzătoare fiecăruia. Utilizatorii sunt reprezentaţi cu ajutorul imaginilor de profil. Utilizatorii sau grupurile care nu au setată o imagine de profil, au atribuită o iconiţă setată standard. Pentru accesarea conversaţiilor este suficient ca utilizatorul să selecteze unul din grupurile afişate, iar astfel va fi redirecţionat spre fereastra de mesagerie a grupului.



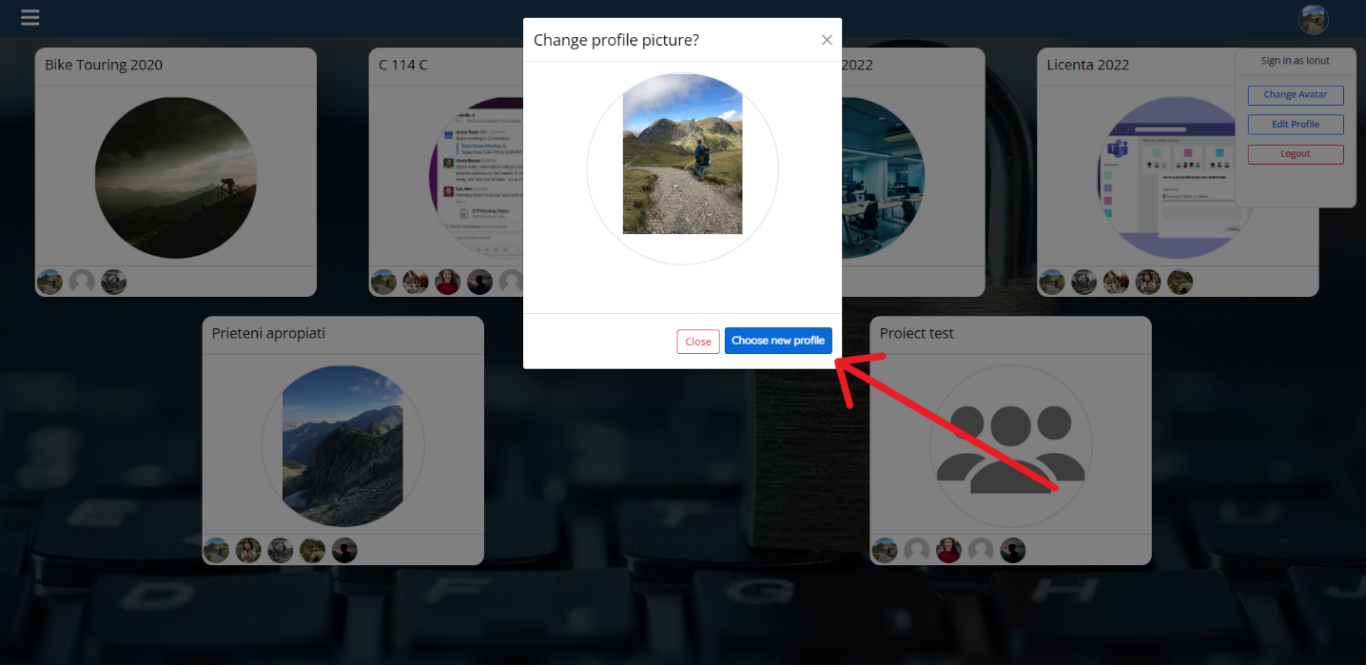
Figură 6.2 Prezentate Fereastră de Homepage

Pentru navigarea spre restul paginilor se foloseşte sidebar-ul din partea stângă. Acesta poate fi accesat prin apăsarea butonului de meniu prezent în navbar. Prin intermediul acesui meniu se poate naviga între ferestrele: *home, messages, files, support* şi *users* (doar pentru utilizatorii cu rol de administrator).



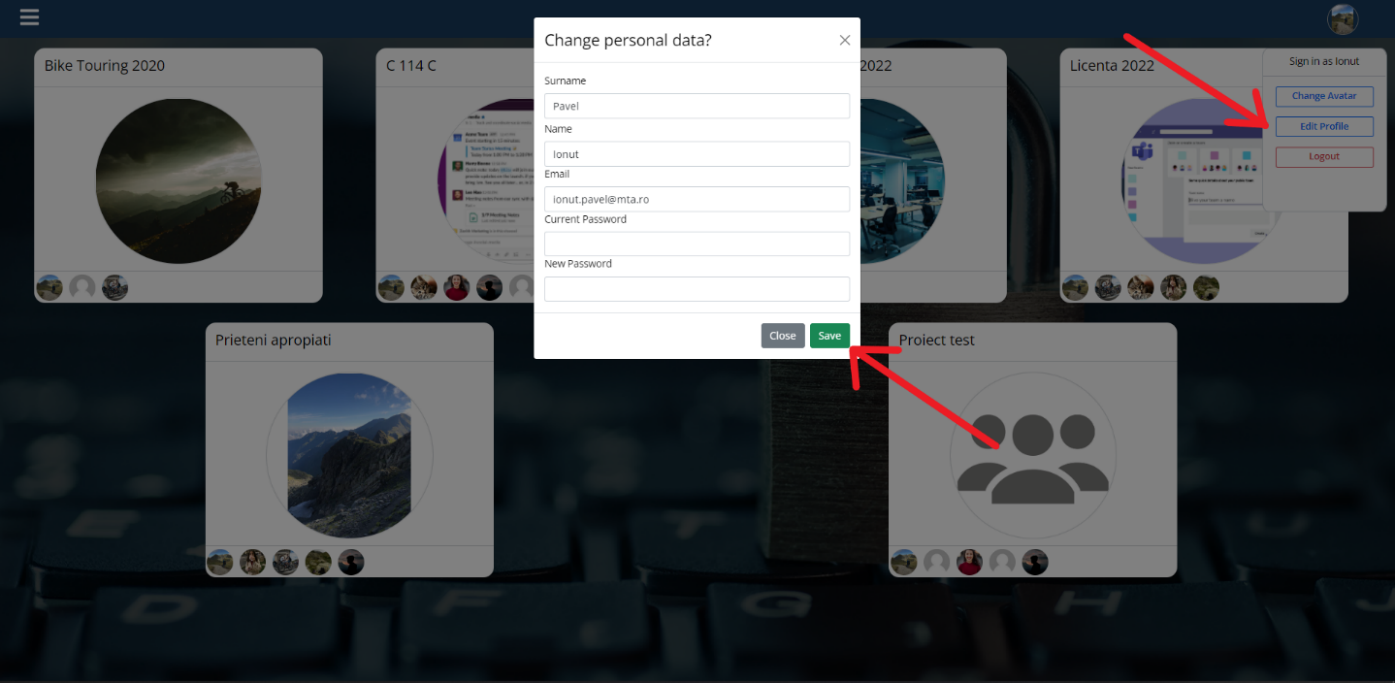
Figură 6.3 Accesarea meniului de navigaţie

Meniul dedicat utilizatorului poate fi deschis prin click asupra profilului din navbar şi prezintă opţiuni de schimbare a imaginii de profil, de editare a datelor personale şi de ieşire din aplicaţie. Aceste meniuri pot fi închise folosind butoanele specifice sau prin click în exteriorul lor. Schimbarea pozei de profil se face prin alegerea unei noi imagini şi confirmarea salvării modificărilor.



Figură 6.4 Prezentare funcţionalitate de schimbare a profilului

Pentru editarea datelor contului, utilizatorul este nevoit să introducă parola actuală, altfel modificările nu vor fi salvate. Pentru închiderea ferestrei, se poate folosi butonul dedicat sau se poate efectua click în afara componentei de editare.



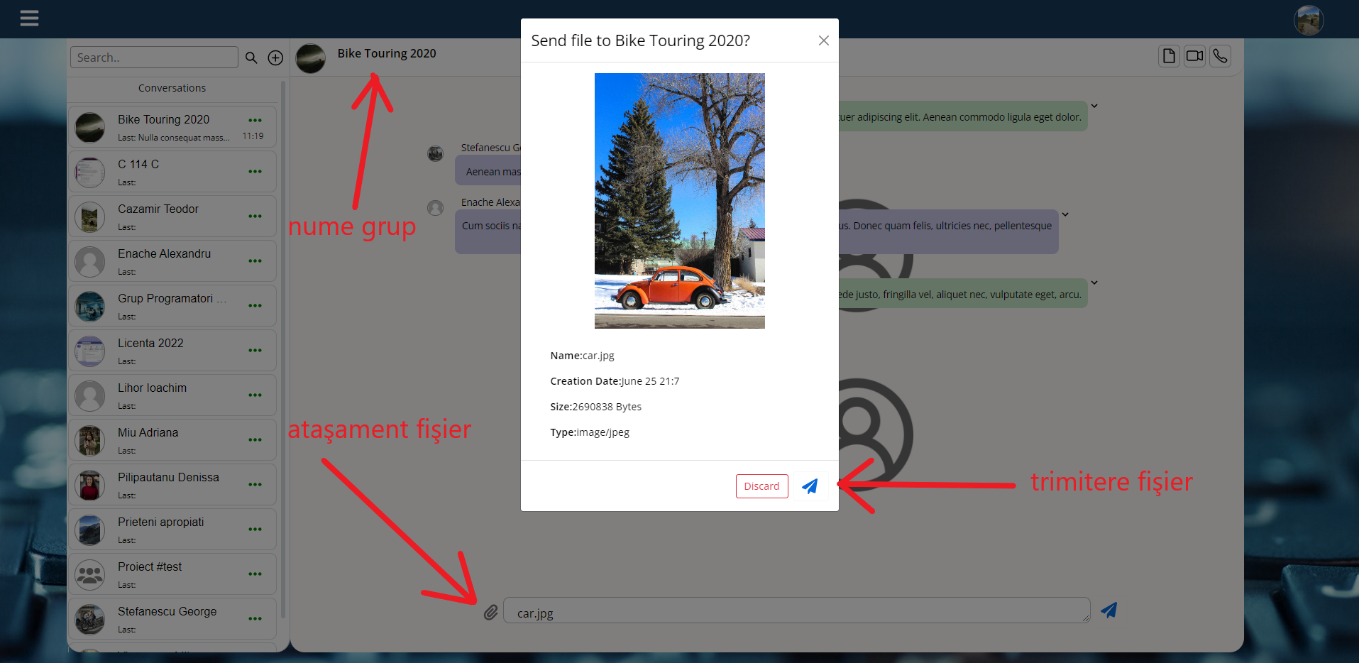
Figură 6.5 Prezentare funcţionalitate de editare a datelor de utilizator

După accesarea butonului de mesagerie, utilizatorul este redirecţionat spre zona de chat a aplicaţiei. Acesta are posibilitatea de a căuta conversaţii existente sau de a începe noi conversaţii cu ajutorul casetei de *search*. Rezultatele căutării sunt sortate în ordine alfabetică în funcţie de keyword-ul căutat. Pentru fiecare conversaţie deschisă este afişat ultimul mesaj alături de ora la care a fost transmis şi un buton pentru mai multe opţiuni.



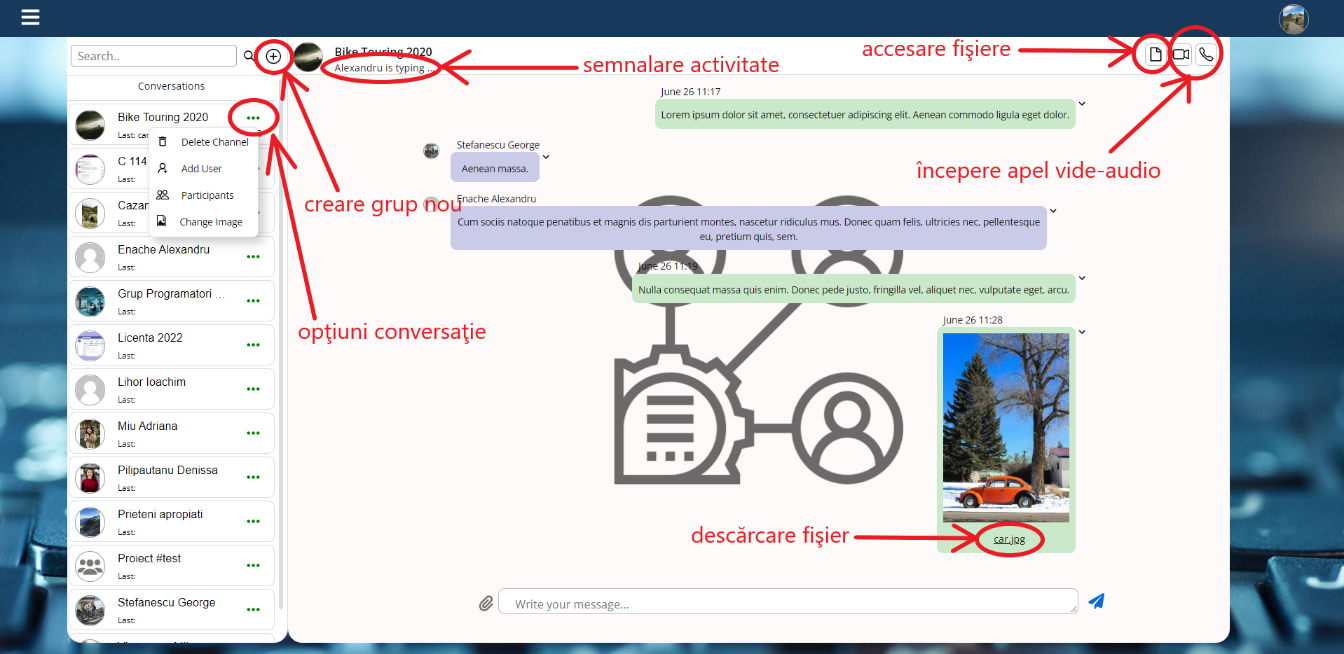
Figură 6.6 Accesarea serviciului de mesagerie

După accesarea unei conversaţii, în partea din dreapta a paginii este deschisă caseta cu mesaje a grupului. Utilizatorul are posibilitatea de a trimite mesaje de tip text, dar şi ataşament de fişiere folosind instrumentele din partea de jos a casetei. După selectarea fişierului dorit pentru trimitere, se va afişa o previzualizare a fişierului împreună cu detalile despre acesta.



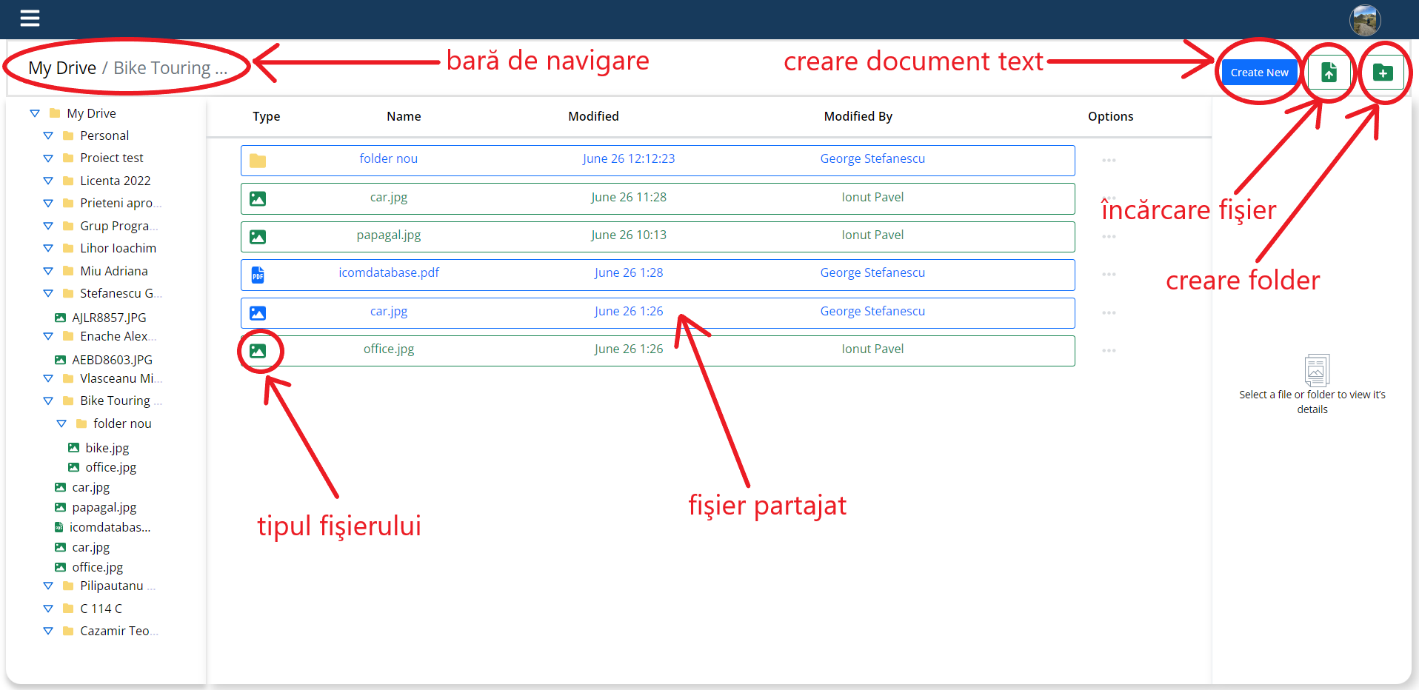
Figură 6.7 Exemplu de trimitere a unui fişier în cadrul conversaţiei

Pentru a crea grupuri noi, utilizatorul trebuie sa acceseze butonul cu iconiţa în formă de cruce din dreptul casetei de search, iar pentru a obţine detalii despre un anumit grup poate accesa butonul de *opţiuni conversaţie*. Fişierele pot fi descărcate folosind link-ul aflat în corpul mesajul, iar pentru vedea tot spaţiul de stocare trebuie să acceseze butonul *accesare fişiere* din partea dreaptă. Intrarea într-un apel video se realizează prin folosirea butoanelor de *începere apel video.*



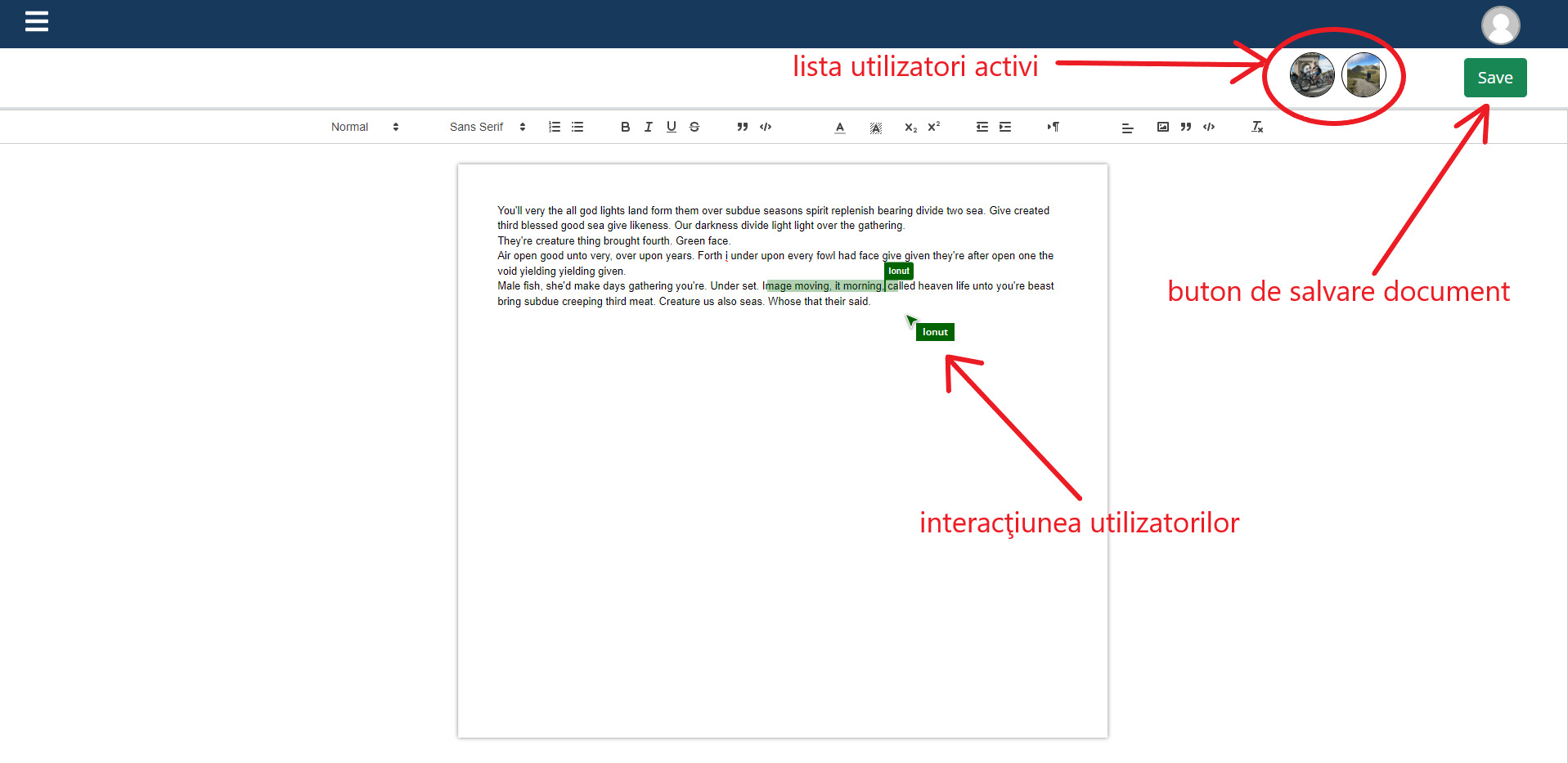
Figură 6.8 Prezentarea funcţionalităţilor din cadrul ferestrei de mesagerie

Spaţiul de stocare simulează un sistem de fişiere prin modul de organizare şi reprezentare. Fiecare conversaţie sau grup este reprezentat printr-un folder în care sunt afişate toate fişierele ce au fost trimise la nivelul conversaţiei. Utilizatorul are şi posibilitatea de a-şi crea foldere personale în folderul rădăcină numit *MyDrive*. Folderele şi fişierele create în interiorul unui folder public pot fi accesate şi de restul membrilor. Fişierele încărcate de alţi utilizatori vor fi marcate printr-un chenar de culoare albastră. Încărcarea unui fişier nou se realizează prin intermediul butonului *încărcare fişier*, iar crearea unui folder nou prin intermediul butonului *creare folder*.Pentru a începe editarea unui document text se accesează butonul de *create new*.



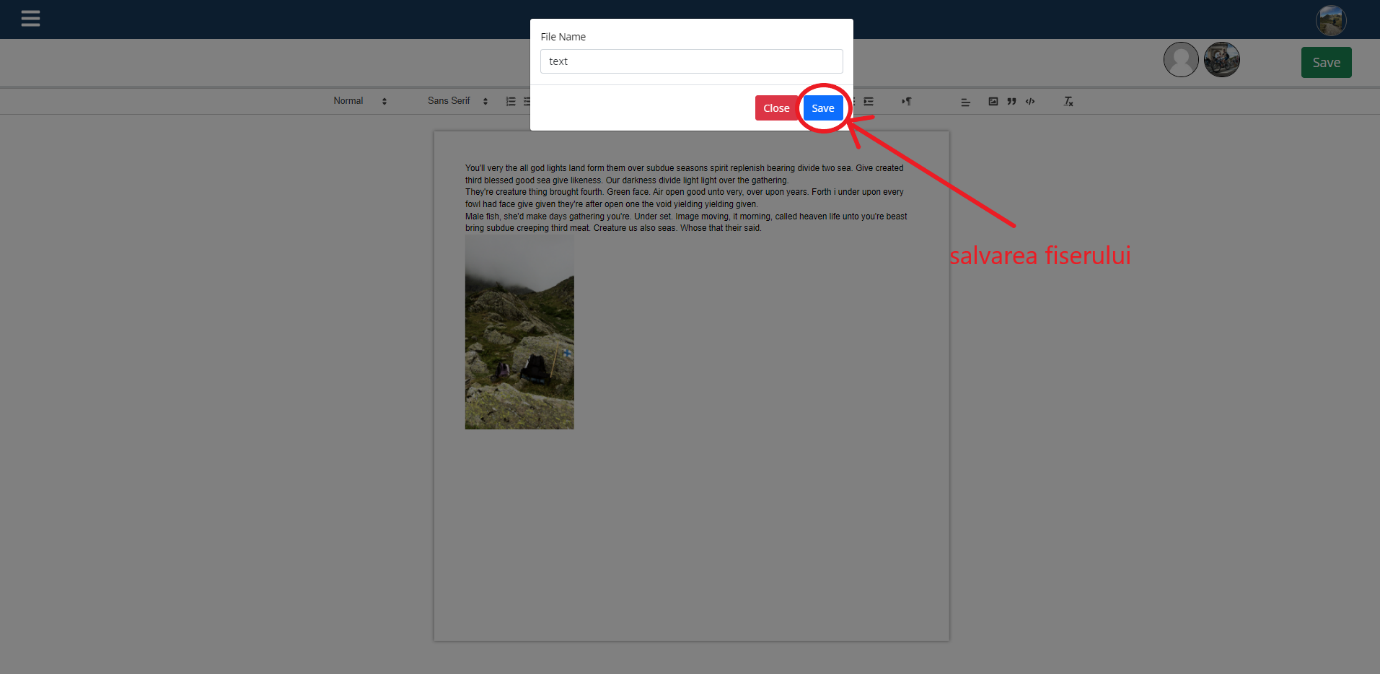
Figură 6.9 Prezentarea sistemului de stocare pentru fişiere

Caseta de text oferă diverse comenzi de formatare a textului. Lista cu utilizatori activi este afişată în colţul din dreapta al paginii alături de butonul de salvare.



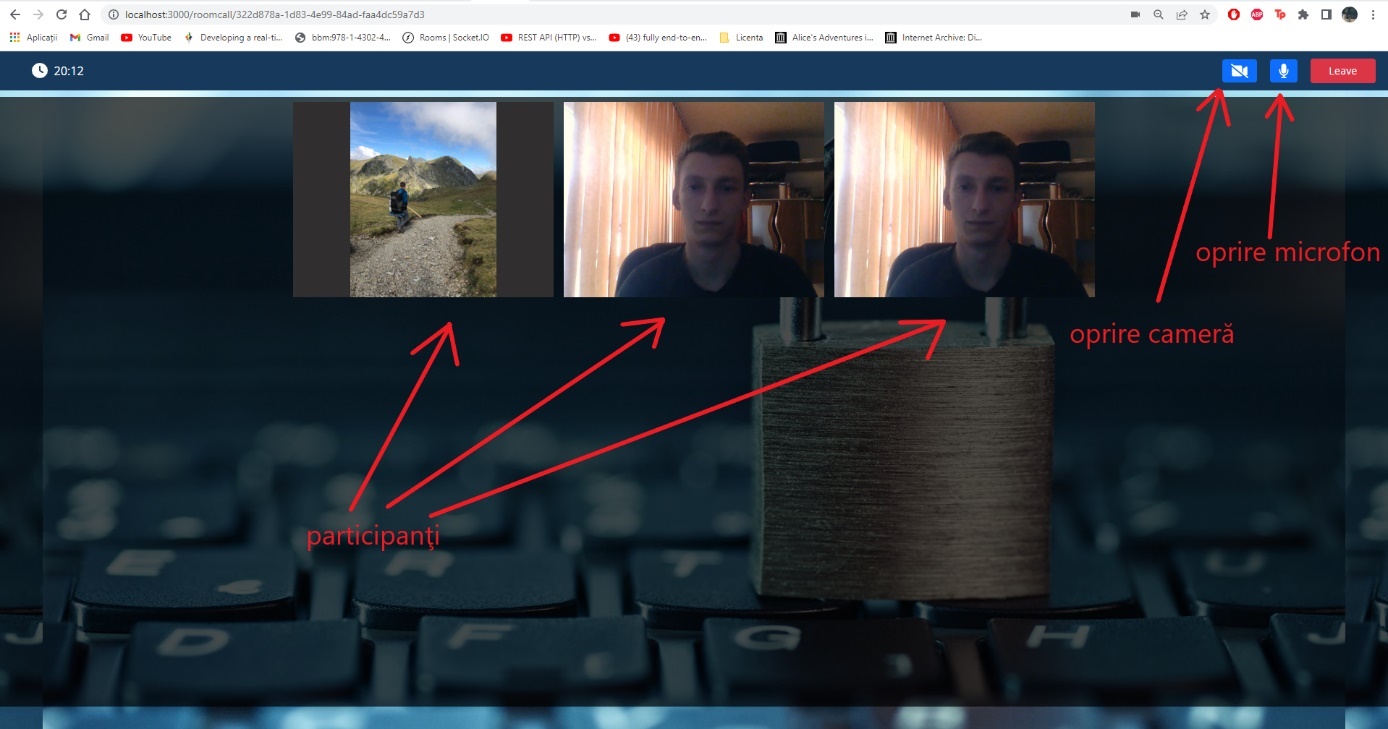
Figură 6.10 Sesiune de editare colaborativă a textului

Fiecare utilizator care participă la editarea documentului poate utiliza pointerul colorat într-o culoare diferită de a celorlalţi. O alta caracteristică funcţională este că fiecare utilizator poate folosi un cursor de text colorat în aceeaşi culoare pentru a evidenţia fragmente de text.



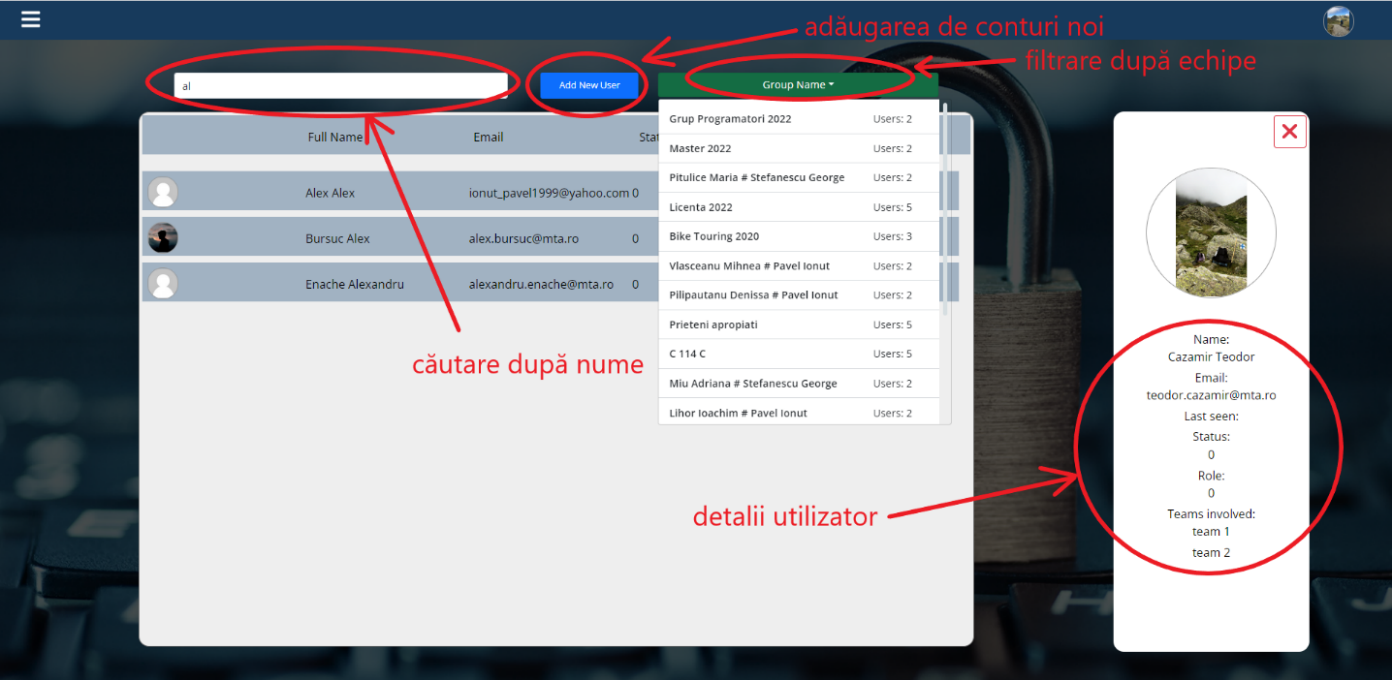
Figură 6.11 Salvarea documentului generat în cadrul sesiunii de editare

În fereastra de apeluri video, participanţii au posibilitatea de a controla camera şi microfonul folosind butoanele dedicate. Pentru participanţii care îşi opresc fluxul video imaginea este înlocuită cu poza acestora de profil. În figura de mai jos sunt trei participanţi în convorbire: primul are camera oprită, iar ceilalţi doi împart acelaşi flux video, deoarece testarea a fost realizată pe infrastructura locală de dezvoltare. Cu alte cuvinte, a fost folosit acelaşi suport fizic pentru simularea participării în cadrul apelului. Părăsirea apelului se poate face prin butonul de *Leave* sau prin accesarea altei pagini din aplicaţie.



Figură 6.12 Prezentarea sesiunii de comunicaţie video-audio

Pentru administrarea conturilor de utilizatori, administratorii au permisiunea de a accesa fereastra de *Users* prin folosirea navbar-ului (opţiune disponibilă doar pentru administratori). În cadrul ferestrei administratorii au posibilitatea de a căuta utilizatori după numele acestora folosind caseta de căutare sau prin folosirea filtrului pus la dispoziţie de meniul de tip dropdown. Pentru crearea unui cont nou de utilizator trebuie accesat butonul de *Add New User*, iar apoi completate câmpurile formularului cu datele utilizatorului. De asemnea, administratorii pot edita sau şterge conturile utilizatorilor, în funcţie de necesitate.



Figură 6.13 Fereastra de control destinată administratorilor

## Testarea sistemului

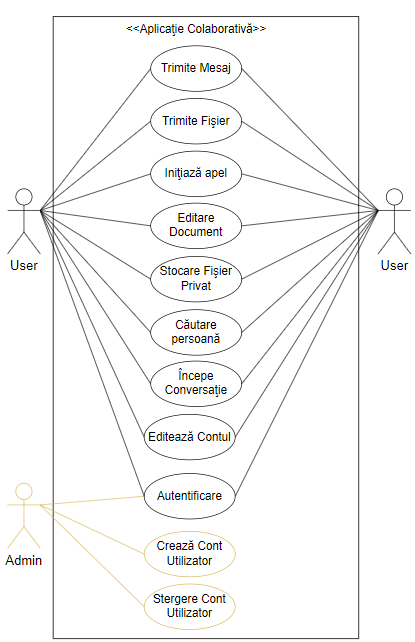
Aplicaţia a fost supusă mai multor serii de teste pentru a verifica funcţionalitatea componentelor, majoritatea dintre acestea desfăşurându-se în etapa de dezvoltare. Testele ce urmau a fi efectuate pentru mediul de producţie nu au putut fi realizare din cauza unei deficienţe de ordin tehnic prezente în etapa de implementare (aplicaţia back-end necesită rularea unui server de tip HTTPS pentru conectarea modului Socket.IO), astfel în mediul de producţie s-au efectuat doar teste legate de posibilitatea de accesare a platformei. Pentru realizarea testelor legate de componentele software este necesar instalarea mediului de execuţie NodeJS *v16.11.0* (pentru rularea celor două aplicaţii) şi crearea bazei de date MySQL de tip MariaDB. Pentru detalierea testelor finale efectuate a fost întocmit următorul raport de testare organizat sub formă tabelară.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nr. Test** | **Nr. Cerinţă** | **Descriere Cerinţă** | **Metoda de Testare** | **Comportament Aşteptat** | **Rezultatul Obţinut** |
| T1 | SRS 1/  SRS2 | Utilizarea aplicaţiei folosind Google Chrome. | Se accesează link-ul aplicaţiei prin Google Chrome. | Browser-ul va solicita introducerea unui certificat valid. | Nu poate fi accesat link-ul, decât după este selectat certificatul de client stocat în store. |
| T2 | SRS 1/  SRS2 | Utilizarea aplicaţiei folosind Mozilla FireFox. | Se accesează link-ul aplicaţiei prin Mozilla FireFox. | Browser-ul va solicita introducerea unui certificat valid. | Nu poate fi accesat link-ul, decât după este selectat certificatul de client stocat în store. |
| T3 | SRS 3 | Căutarea de utilizatori pentru iniţierea unei conversaţii noi. | Se utilizează searchbox-ul din mesagerie pentru căutarea unei persoane. | Pe măsură ce utilizatorul introduce caractere, vor fi afişate persoanele ce conţin sintagma introdusă. | Rezultatele afişate se potrivesc patternului. Se pot crea conversaţii private prin accesarea rezultatelor în mod individual. |
| T4 | SRS 4 | Comunicaţia dintre utilizatori folosind mesaje text. | La nivelul conversaţiei private se trimit mesaje text şi se verifică dacă sunt primite de către destinatat. | Destinatarul va fi în măsură să primească mesajele în ordinea corectă şi sub acelaşi format în care au fost trimise. | Destinatarul a recepţionat mesajele în ordinea corectă în momentul în care acestea au fost transmise. |
| T5 | SRS 5 | Accesarea fişierelor trimise la nivelul conversaţiei. | După crearea unei conversaţii noi utilizatorul A trimite utilizatorului B un fişier. După primirea fişierului utilizatorul B va accesa butonul pentru accesarea storage-ului din cadrul conversaţiei. | Utilizatorul B va fi direcţionat în zona mediului de stocare unde va putea accesa fişierul trimis în cadrul conversaţiei din lista de elemente afişate. | Sistemul prezintă toate fişierele trimise în cadrul conversaţiei sub forma unei lista, conform specificaţiilor cerinţei. |
| T6 | SRS 6 | Crearea unui grup cu mai mult de 2 utilizatori. | Se accesează butonul pentru crearea unui grup nou şi se completează cu numele grupului. | Grupul nou creat apare în lista conversaţiilor. | Grupul poate fi accesat, se pot trimite mesaje şi poate fi accesată zona de storage. |
| T7 | SRS 7 | Adăugarea de participanţi noi. | Se accesează opţiunea de adăgare participanţi noi, apoi se alege o persoană din lista afişată. | Utilizatorul nou adăugat poate accesa grupul din lista de conversaţii împreună cu mediul de stocare partajat. | Utilizatorul poate primi şi trimite mesaje. Acesta are acces la toate fişierele trimise până atunci în cadrul grupului. |
| T8 | SRS 8/  SRS 9 | Trimiterea şi descărcarea fişierelor. | Din cadrul conversaţiei, utilizatorul trimite un fişier folosind butonul de ataşament. | La nivelul converaţiei va fi trimis un mesaj ce conţine denumirea fişierului şi o previzualizare în cazul fişierelor de tip imagine. | Sistemul se comportă conform aşteptărilor, fiind posibilă descărcarea fişierelor prin accesarea link-ului de acces.. |
| T9 | SRS 10 | Efectuarea apelurilor video. | Doi utilizatori diferiţi accesează butonul de pornire al apelului. | Utilizatorii sunt direcţionaţi în fereastra dedicată, iar după cateva secunde vor avea posibilitatea de a se comunica video şi audio (pe langă imaginea proprie, pe ecran pot vedea şi imaginea celuilalt utilizator). | Sistemul are întârzieri de conectare, iar utilizatorul nu primeşte primeşte nici un fel de notificare că există o conexiune în curs de realizare. După conectare comunicaţia decurge normal, cu unele întârzieri. |
| T10 | SRS 11/  SRS 12/  SRS 13 | Crearea şi editarea fişierelor text în mod colaborativ. | Este accesat butonul pentru crearea fişierelor text din secţiunea de storage, apoi adresa URL a site-ului este trimisă utilizatorilor cu care se doreşte colaborarea. Utilizatorii vor accesa adresa trimisă şi vor fi redirecţionaţi spre editorul de text. | Este accesat mediul de editare reprezentat prin editorul de text., iar utilizatorul are posibilitatea de a formata textul. Restul participanţilor sunt afişaţi sub formă de listă şi vor avea un cursor colorat. | Sistemul respectă specificaţiile din cerinţe, dar nu sunt afişaţi mai mult de un pointer simultan şi poziţia acestora este decalată faţă de poziţia reală. De asemnea după deconectara utilizatorului cursorul specific acestuia nu este eliminat. |

## Diagrame UML

### Diagrama cazurilor de utilizare

În figura 6.14 este reprezentată diagrama cazurilor de utilizare ale aplicaţiei din perspectiva utilizatorului. Folosind aplicaţia pusă la dispoziţie, utilizatorii pot căuta persoane pentru a începe noi conversaţii pentru ca mai apoi să trimită mesaje şi fişiere în cadrul acestora. Utilizatorii pot participa la conversaţii de tip apel şi pot utiliza mediul de stocare pus la dispoziţie. Un administrator va putea controla activitatea utilizatorilor prin crearea sau ştergerea conturilor de utilizator normal.

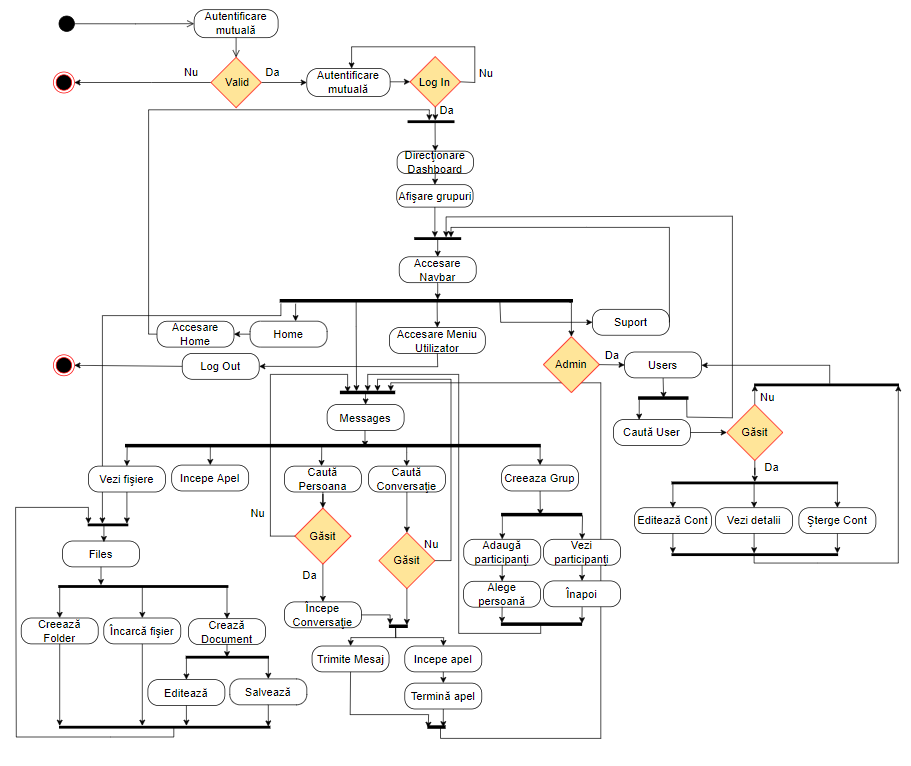


Figură 6.14 Diagrama cazurilor de utilizare

### Diagrama de activităţi

Pentru a explica funcţionalităţile şi stările aplicaţiei am întocmit diagrama de activităţi reprezentată în figura 6.14. Starea de start a sistemului este reprezentată de momentul în care utilizatorul accesează adresa aplicaţiei, iar starea de final reprezintă finalul sesiunii de lucru a utilizatorului.

După autentificare, utilizatorul poate naviga la nivelul fiecărui modul prin accesarea componentei *Navbar*. De menţionat că, pentru începerea unui apel video este necesar ca utilizatorul să acceseze starea de căutarea a unei conversaţii noi sau de a selecta o conversaţie deja creată. Un document de tip text poate fi creat doar de la nivelul ferestrei de ***Storage***, în figură fiind reprezentat prin starea ***Files***.



Figură 6.15 Diagrama de activităţi

# Concluzii

## Probleme întâmpinate

În timpul dezvoltării aplicaţiei au fost întâmpinate câteva probleme ce ar trebui luate în considerare în dezvoltarea unor astfel de sisteme.

Printre problemele întâmpinate se numără distrugerea obiectelor de tip *Peer()*, din cadrul bibliotecii „*simple-peer*”, folosite la nivelul componentei *VideoRoom*. Un obiect *Peer()* defapt un obiect ce menţine datele corespondente unei conexiuni ce se stabileşte între browser-ele utilizatorilor, prin care este transmis fluxul video-audio. Iniţial lista cu obiectele respective era salvată cu ajutorul hook-ului *useState()* şi chiar dacă era apelată metoda *destroy()* pentru obiectul respectiv, lista nu suferea nici o modificare, iar conexiunile rămâneau deschise. În timpul testelor acest aspect a fost mai greu de detectat din cauză că obiectul ce se dorea distrus era eliminat din cadrul listei prin setarea uneia noi, iar vizual, caseta video corespunzătoare obiectului nu mai era afişată. În momentul în care se produceau mai multe intrări şi ieşiri dintr-un apel, sistemul de comunicaţie prezenta întârzieri, finalizând cu un blocaj, din cauza unui număr prea mare de conexiuni active. Pentru rezolvare am utilizat *hook*-ul *useRef()*. Utilizarea acestuia a permis salvarea referinţei corespondente obiectelor din *useState()* şi astfel obiectele pot fi prelucrate prin folosirea referinţei.

O altă problemă întâmpinată în implementarea software a fost necesitatea transmiterii variabilelor între componente ce nu aveau o relaţie de legătură directă. Această necesitate a devenit o problemă odată cu dezvoltarea modulelor în care era nevoie de folosirea unor variabile comune precum datele utilizatorului conectat sau date ce aveau legătură cu starea aplicaţiei. Rezolvarea a venit prin utilizarea bibliotecii React Redux. Acest sistem permite modificarea şi accesarea datelor de la nivelul oricărei componente prin folosirea funcţiei *useSelector()*.

O problemă apărută în momentul trecerii din mediul de dezvoltare în mediul de producţie este incompatibilitatea serverului de tip websocket din cadrul aplicaţiei server cu conexiunea pe care încearcă să o stabilească clientul prin intermediul browser-ului (browser-ul încearcă să creeze conexiune HTTPS, iar serverul Socket.IO este de tip HTTP). Pentru rezolvare, serverul trebuie creat de tipul HTTPS.

## Rezultate obţinute

Prezenta lucrare prezintă procedeul de implementare al unei soluţii software colaborative, având ca scop documentarea tipurior de sisteme similare şi implementarea unei soluţii proprii. În cadrul proiectului au fost descris stadiul actual, s-au detaliat tehnologiile folosite şi a fost descrisă implementarea unei aplicaţii proprii, construite cu scopul de a cuprinde conceptele şi tehnologiile studiate. Aplicaţia propusă îşi atinge obiectivul principal de a asigura munca colaborativă la nivelul unui grup restrâns de persoane. Deşi sistemul nu este optimizat reuşeşte să asigure funcţionalităţile implementate fără probleme de ordin tehnic şi cu performanţe decente.

## Dezvoltări ulterioare

Aplicaţia implementată necesită optimizări care ar duce la o stabilitatea ridicată pentru sistem. Pentru a acoperi în totalitate necesitatea de instrumente colaborative se pot propune spre implementare module precum aplicaţii calendar sau organizator de task-uri.

Capacitatea numărului de participanţi la un apel video poate fi crescută prin adăugarea unui server dedicat de prelucrare video. Scopul unui astfel de server este de a prelua fluxul video de la fiecare participant, să-l unifice prin procesare, iar apoi să direcţioneze rezultatul fiecărui client înrolat în convorbire. Un astfel de server este destul de complex pentru a fi implementat intern, dar performanţele ar fi cu mult îmbunătăţite.

O altă îmbunătăţire ce ar putea fi adusă este instalarea şi folosirea unui server extern dedicat pentru formatarea fişierelor în format Word. Aceste tipuri de fişiere sunt prea complexe pentru a putea fi convertite direct în format HTML şi reprezentate cu ajutorul un editor normal, aşa că au nevoie de a fi procesate de către sisteme dedicate.

# 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] 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).

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

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

[6] 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).

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

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

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

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

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

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

[13] 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

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

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

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

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

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

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

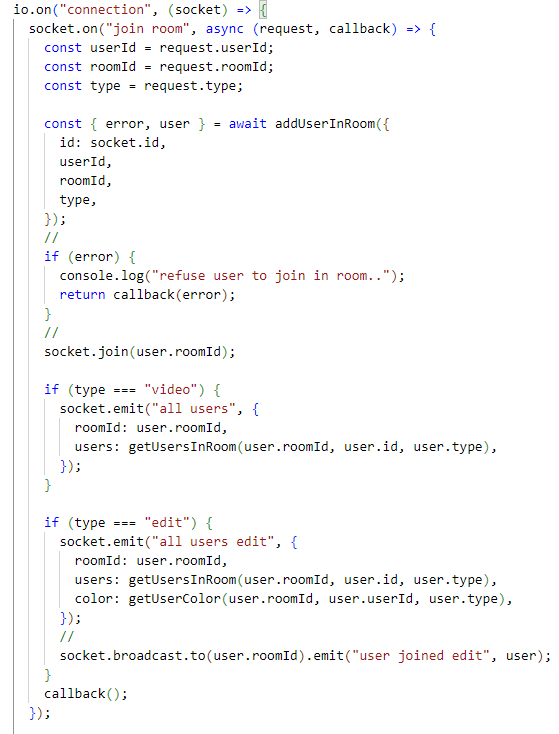
# Anexe

## Anexa A

Codul sursă al proiectului poate fi vizualizat prin accesarea link-ului către repository-ul de Github de mai jos:

<https://github.com/ionut99/icomV2>

## Anexa B









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