Cuprins

[1. Introducere 2](#_Toc144701310)

[2. Concepte teoretice ale aplicațiilor de mesagerie în timp real 7](#_Toc144701311)

[2.1 Scurtă descriere ale aplicațiilor de mesagerie în timp real 7](#_Toc144701312)

[2.2 Aplicații utilizate în prezent 7](#_Toc144701313)

[2.3 Comparație între Facebook Messenger, Whatsapp și Signal 10](#_Toc144701314)

[3. Tehnologiile software utilizate 11](#_Toc144701315)

[3.1 Visual Studio Code 11](#_Toc144701316)

[3.2 JavaScript 11](#_Toc144701317)

[3.3 React.js 11](#_Toc144701318)

[3.4 Node.js 12](#_Toc144701319)

[3.5 Express.js 12](#_Toc144701320)

[3.6 Git 12](#_Toc144701321)

[3.7 Sass 12](#_Toc144701322)

[3.8 MongoDB 13](#_Toc144701323)

[4. Baza de date 14](#_Toc144701332)

[4.1 Modelul utilizatorilor 14](#_Toc144701333)

[4.2 Modelul mesajelor 15](#_Toc144701334)

[5. Aplicația Chitchat 17](#_Toc144701324)

[5.1 Introducere 17](#_Toc144701325)

[5.2 Cerințe funcționale ale aplicației 17](#_Toc144701326)

[5.3 Dezvoltarea aplicației 18](#_Toc144701327)

[5.3.1 Pagina de conectare 18](#_Toc144701328)

[5.3.2 Pagina de înregistrare 21](#_Toc144701329)

[5.3.3 Pagina de alegere a pozei de profil 25](#_Toc144701330)

[5.3.4 Pagina principală 28](#_Toc144701331)

[5.3.5.1 Componenta Navbar 18](#_Toc144701328)

[5.3.5.2 Componenta SettingsMenu 21](#_Toc144701329)

[5.3.5.3 Componenta Chats 25](#_Toc144701330)

[5.3.5.4 Componenta Messages 21](#_Toc144701329)

[5.3.5.5 Componenta Input 21](#_Toc144701329)

[6. Concluzii 47](#_Toc144701335)

[7. Bibliografie 48](#_Toc144701337)

Listă figuri

[Figură 1 (Aplicația WhatsApp) 8](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950365)

[Figură 2 (Aplicația Signal) 9](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950366)

[Figură 3 (Aplicația Facebook Messenger) 10](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950367)

[Figură 4 (Modelul User) 15](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950368)

[Figură 5 (Modelul Message) 16](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950369)

[Figură 6 (Pagina de conectare) 19](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950370)

[Figură 7 (Notificare Eroare Conectare 1) 19](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950371)

[Figură 8 (Notificare Eroare Conectare 2) 19](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950372)

[Figură 9 (Notificare Eroare Conectare 3) 19](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950373)

[Figură 10 (Notificare Eroare Conectare 4) 20](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950374)

[Figură 11 (Funcția handleChange() din pagina de conectare) 20](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950375)

[Figură 12 (Funcția handleValidation() din pagina de conectare) 20](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950376)

[Figură 13 (Funcția handleSubmit() din pagina de conectare) 21](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950377)

[Figură 14 (Controler pentru conectarea în aplicație) 22](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950378)

[Figură 15 (Pagina de înregistrare) 23](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950379)

[Figură 16 (Notificare eroare la înregistrare 1) 23](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950380)

[Figură 17 (Notificare eroare la înregistrare 2) 23](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950381)

[Figură 18 (Notificare eroare la înregistrare 3) 23](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950382)

[Figură 19 (Notificare eroare la înregistrare 4) 24](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950383)

[Figură 20 (Notificare eroare la înregistrare 5) 24](#_Toc144950384)

[Figură 21 (Notificare eroare la înregistrare 6) 24](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950385)

[Figură 22 (Funcția handleChange() din pagina de înregistrare) 24](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950386)

[Figură 23 (Funcția handleValidation() din pagina de înregistrare) 24](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950387)

[Figură 24 (Funcția handleSubmit() din pagina de înregistrare) 25](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950388)

[Figură 25 (Controlerul de înregistrare) 25](#_Toc144950389)

[Figură 26 (Pagina de alegere a pozei de profil) 26](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950390)

[Figură 27 (Funcția de prelucrare a pozei de profil) 27](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950391)

[Figură 28 (Funcția de deschidere al managerului de fișiere) 27](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950392)

[Figură 29 (Funcția de selectare a pozei standard) 28](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950393)

[Figură 30 (Funcția de trimitere a pozei de profil selectate în baza de date) 28](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950394)

[Figură 31 (Controlerul de prelucrare a pozei de profil) 29](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950395)

[Figură 32 (Pagina principală atunci când nu este selectat un contact) 30](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950396)

[Figură 33 (Pagina principală atunci când unul din contacte este selectat) 30](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950397)

[Figură 34 (Funcția de selectare a contactului) 31](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950398)

[Figură 35 (Funcția de ștergere a conversației curente) 31](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950399)

[Figură 36 (Funcțiile pentru instanțierea socket-ului în client și pentru returnarea contactelor din listă) 32](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950400)

[Figură 37 (Controler pentru ștergerea conversației curente) 32](#_Toc144950401)

[Figură 38 (Funcția de conectare a utilizatorului la socket) 33](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950402)

[Figură 39 (Componenta Navbar) 33](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950403)

[Figură 40 (Funcțiile de deschidere a meniului și pentru mărirea pozei de profil) 33](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950404)

[Figură 41 (JSX-ul pentru componenta Navbar) 34](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950405)

[Figură 42 (Meniul de setări) 35](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950406)

[Figură 43 (Funcția de schimbare a pozei de profil) 36](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950407)

[Figură 44 (Prompt pentru modificarea numelui de utilizator) 36](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950408)

[Figură 45 (Funcția de modificare a numelui de utilizator) 37](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950409)

[Figură 46 (Funcția de deconectare din aplicație) 38](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950410)

[Figură 47 (Fereastră de confirmare pentru ștergerea contului) 38](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950411)

[Figură 48 (Funcția de ștergere a contului) 38](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950412)

[Figură 49 (Controler pentru modificarea pozei de profil) 39](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950413)

[Figură 50 (Controler pentru modificarea numelui de utilizator) 39](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950414)

[Figură 51 (Controler pentru ștergerea contului de utilizator) 40](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950415)

[Figură 52 (Componenta Chats) 41](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950416)

[Figură 53 (Funcția de schimbare a contactului selectat și JSX-ul componentei) 42](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950417)

[Figură 54 (Componenta auxiliară pentru definirea fiecărui contact) 42](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950418)

[Figură 55 (Controler pentru returnarea tuturor contactelor) 43](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950419)

[Figură 56 (Componenta Messages) 44](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950420)

[Figură 57 (Componenta auxiliară afișată când niciun contact nu este selectat) 45](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950421)

[Figură 58 (Componenta auxiliară pentru conținutul mesajului) 45](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950422)

[Figură 59 (Funcția pentru trimiterea mesajelor) 45](#_Toc144950423)

[Figură 60 (Funcția pentru returnarea mesajelor dintre 2 utilizatori) 46](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950424)

[Figură 61 (Funcția pentru afișarea mesajelor primite) 46](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950425)

[Figură 62 (Funcțiile pentru actualizarea vectorului de mesaje și respectiv schimbarea focusului la ultimul mesaj primit) 46](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950426)

[Figură 63 (Controler pentru trimiterea mesajelor în baza de date) 47](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950427)

[Figură 64 (Controler pentru returnarea mesajelor din baza de date) 48](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950428)

[Figură 65 (Componenta Input) 48](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950429)

[Figură 66 (Funcțiile componentei Input) 49](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950430)

[Figură 67 (Funcția pentru trimiterea pozelor între utilizatori) 49](https://d.docs.live.net/e3942b5e49621312/Documents/ChitChat_Pop_Cristian_Mihail_Stabila.docx#_Toc144950431)

# Introducere

Comunicarea este prezentă în activitățile noastre zilnice. Indiferent de tipul de limbaj utilizat cum ar fi limbajul corpului, vocalizarea, semnele sau cuvintele scrise, comunicarea este o parte fundamentală a ființelor umane. Comunicarea a început încă din evul mediu, de la transmiterea mesajelor la distanțe mari prin intermediul porumbeilor voiajori, până în prezent unde transmiterea mesajelor între oameni se realizează indiferent de distanța care îi desparte, prin intermediul aplicațiilor web si mobile.

Aplicațiile de mesagerie datează încă din anii 80’, perioadă în care a fost lansat primul serviciu de comunicare online disponibil publicului, numit CB Simulator, creat de compania CompuServe. Primul „Short Message Service” a fost creat de către Vodafone GSM in anul 1992, iar spre finalul anilor 90’, au luat amploare MSN Messenger (22 Iulie 1999) și YAHOO Messenger (9 Martie 1998). Odată cu începutul anilor 2000, au apărut renumitele aplicații desktop cum ar fi Skype si Google Talk, ce au permis utilizatorilor sa comunice la distanță prin apeluri audio sau video.

Apariția aplicațiilor de comunicare ne-a oferit o mulțime de avantaje care au transformat felul in care oamenii comunică în prezent. În primul rând, acestea oferă utilizatorilor o comunicare eficientă si rapidă, cu o latență minimă în transmiterea mesajelor sau în convorbirile audio/video. În al doilea rând, acestea sunt foarte convenabile și accesibile, accesarea lor fiind la doar câteva click-uri sau apăsări de ecran distanță și necesită doar o conexiune constanta la internet. În al treilea rând, utilizatorii pot crea în cadrul aplicațiilor camere de grup, astfel comunicarea se face mai ușor între membrii familiilor, între elevi/studenți sau la locul de muncă. Odată cu creșterea masivă a utilizatorilor și a cerințelor acestora, aplicațiile au evoluat de la simpla transmitere de mesaje text la trimiterea de poze, documente, videoclip-uri, înregistrări audio sau chiar locații în timp real ale utilizatorilor.

Din păcate, aplicațiile de mesagerie în timp real au venit și cu un set de dezavantaje majore. În primul rând, multe dintre aplicațiile disponibile pe piață nu au implementat criptare end-to-end, astfel mesajele trimise de emițător sunt criptate pe dispozitivul acestuia și decriptate pe dispozitivul receptorului. Această practică duce la o diminuare a interceptării mesajelor de către persoanele neautorizate. În al doilea rând, aceste aplicații sunt vizate de atacuri de tip Phising, unde un utilizator poate primi fișiere corupte sau link-uri către website-uri false pentru a îi compromite datele personale. Pentru a evita problemele de securitate expuse anterior, s-au implementat mai multe metode pentru îmbunătățirea securității transmiterii datelor în cadrul acestor aplicații, cum ar fi:

* Autentificarea prin 2 factori (2FA): în autentificarea clasică, utilizatorul trebuie în general să introducă numele de utilizator sau adresa de e-mail și o parolă, iar odată validate aceste date utilizatorul este conectat în aplicație. Autentificarea prin 2 factori, adaugă un strat suplimentar la nivelul securității, astfel odată ce utilizatorul a introdus adresa de e-mail și parola, acestuia i se cere o dată de validare suplimentară setată de el la momentul creării contului. Un exemplu din viața reală poate fi conectarea la platforma Facebook Messenger, unde utilizatorului i se cere adresa de e-mail și parola contului, iar dacă aceste date sunt valide, va primi un SMS pe numărul de telefon salvat în momentul creării contului cu un cod care trebuie introdus în platformă.
* Actualizări de securitate: firmele care se ocupă cu dezvoltarea acestor aplicații, lucrează în permanență la rezolvarea vulnerabilităților, și încurajează utilizatorii să mențină versiunile aplicațiilor la zi.
* Informarea utilizatorilor: firmele anunță utilizatorii să nu trimită date confidențiale, asigurându-i că aceasta nu le va trimite mesaje în care să ceară astfel de informații.
* Urmărirea activităților suspecte: aplicațiile utilizează algoritmi de urmărire și detectare a activităților suspecte, cum ar fi încercări repetate de conectare la aplicație eșuate si un număr prea mare de cereri către server într-un timp scurt.

In prezent, găsim o multitudine de aplicații de mesagerie în timp real, cum ar fi:

* Facebook: această aplicație a fost creată în anul 2004, a primint un update pentru a permite utilizatorilor să comunice în timp real în anul 2008 și ulterior au fost create aplicații dedicate pentru Android si IOS în anul 2011.
* WhatsApp: cea mai populară aplicație de mesagerie, creată în anul 2009, cu un sistem de criptare end-to-end, ce permite transmiterea de mesaje text, poze, documente, videoclip-uri, înregistrări audio etc.
* Instagram: această aplicație a fost creată în anul 2010 cu scopul împărtășirii imaginilor prin postări între utilizatori și ulterior a introdus un sistem de mesagerie în timp real în anul 2013 numit Instagram Direct.
* Discord: aplicație creată în anul 2015, a avut o creștere în popularitate în anul 2018 datorită creșterii jucătorilor online.
* Telegram: aplicație creată în anul 2013, este renumită pentru nivelul ridicat de securitate oferit de către aplicație prin criptarea end-to-end.

Odată cu creșterea accelerată a numărului de utilizatori in cadrul aplicațiilor de comunicare, ideea de intimitate digitală a dispărut, deoarece principiile morale generale ale conducătorilor au fost mereu acaparate de avere, sacrificând astfel intimitatea mesajelor și datelor personale pentru anumite sume de bani.

În crearea aplicației ChitChat am avut următoarele obiective:

* Facilitarea comunicării între utilizatori printr-un sistem eficient și rapid
* Crearea unei aplicații ușor de extins, astfel încât se vor putea adăuga funcționalități noi în dezvoltarea ulterioară

Această lucrare este structurată în 6 capitole:

* 1. Introducere: în acest capitol voi prezenta o scurtă istorie a primelor aplicații de mesagerie în timp real, avantajele si dezavantajele aplicațiilor de mesagerie, scurtă descriere a aplicațiilor din prezent și obiectivele aplicației ChitChat.
  2. Concepte teoretice ale aplicațiilor de mesagerie în timp real: în acest capitol voi prezenta teoria acestor tipuri de aplicații și un scurt istoric, câteva aplicații utilizate în prezent și o comparație între acestea.
  3. Tehnologiile software utilizate: aici voi descrie tehnologiile alese pentru dezvoltarea acestei aplicații și motivul pentru care le-am ales
  4. Baza de date: voi descrie modelele create în MongoDB
  5. Aplicația Chitchat: aici voi descrie componentele și paginile create, controlerele folosite pentru validarea datelor și transmiterea acestora în baza de date.
  6. Concluzii: aici voi realiza o sinteză a rezultatelor obținute în cadrul creării acestei aplicații, și ulterioarele actualizări care pot fi aduse.

Acestea sunt capitolele principale ale lucrării, cu o descriere scurtă despre ce conține fiecare. Aceste capitole vor fi dezvoltate în detaliu unde voi oferi detalii si exemple relevante temei abordate.

# Concepte teoretice ale aplicațiilor de mesagerie în timp real

În acest capitol voi descrie cum funcționează în teorie aplicațiile de mesagerie în timp real, cu un mic istoric al primelor sisteme de acest tip.

## 2.1 Scurtă descriere ale aplicațiilor de mesagerie în timp real

Sistemele de mesagerie în timp real asigură schimbul cu o latență cât mai scăzută între mesajele de tip text cu unul sau mai mulți utilizatori în același timp, conectate în general printr-o conexiune la internet.

Diferența majoră față de curierul electronic (e-mail) este faptul că acest tip de comunicare afișează mesajele trimise de utilizatori aproape în timp real, permițând un dialog asemănător cu cel față în față.

Aceste aplicații de mesagerie sunt în general gratuite, deoarece aceste mesaje reprezintă puține date, iar vitezele acestor aplicații sunt mulțumitoare indiferent de distanța care desparte utilizatorii.

Conceptul de mesagerie în timp real este unul destul de vechi, pornind încă de la sistemele *UNIX* ce oferă o comunicare prin mesaje de tip text prin comanda *talk*. Ulterior *Windows* a creat un echivalent, prin comanda *WinPopUp.*

În primele sisteme de mesagerie în timp real, în momentul în care un utilizator tasta mesajul, fiecare literă introdusă de acesta apărea atunci când era tastată sau modificată. În prezent, putem vedea mesajul primit de la un utilizator doar atunci când acesta a ales să îl trimită, trecând printr-un sistem de validare înainte de trimitere.

## 2.2 Aplicații utilizate în prezent

În acest capitol mă voi axa pe cele mai cunoscute aplicații de mesagerie criptate end-to-end (datele sunt criptate pe telefonul emițătorului și decriptate pe telefonul receptorului) din prezent, acestea fiind:

* Whatsapp
* Signal
* Facebook Messenger

Aplicația *Whatsapp* [Figură 1] poate fi găsită în 80% din telefoanele oamenilor. În urma unei statistici făcute de *Statista,* s-a dovedit că această aplicație este cea mai utilizată din întreaga lume. *Whatsapp* a creat aplicații dedicate pentru toate platformele, încât daca se accesează un calculator și se trimite un mesaj pe *Whatsapp* acesta oferă două posibilități: aplicația specială pentru calculatoare și aplicația accesibilă din browser. În cazul în care, se accesează aplicația de pe telefonul mobil, acestea sunt dedicate pentru sistemul de operare Android cât și pentru sistemul de operare IOS. Asta a dus la o creștere masivă a utilizatorilor acestei aplicații, atingând cifra de 2.2 miliarde de utilizatori în anul 2021, cifră care este într-o creștere constantă.[ (Wikipedia, 2023)]

A screenshot of a phone and computer

Description automatically generatedAceastă aplicație a fost creată în anul 2009 de către Brian Acton și Jan Koum drept o alternativă pentru SMS (Short Message Service) care era destul de costisitor. *Whatsapp* are la bază limbajul de programare *Erlang,* limbaj conceput pentru programare concurentă, în care mai multe procese rulează concomitent pentru a avea o performanță mai mare.

Figură 1 (Aplicația WhatsApp)

A doua aplicație de mesagerie în timp real este Signal [Figură 2], cea mai securizată aplicație de pe piață în acest moment. Aceasta este o aplicație cu sursă deschisă, astfel oricine are acces la codul sursă al aplicației, dovedind că nu are absolut nimic de ascuns față de utilizatorii săi și în plus este o aplicație complet gratuită.

Creată în anul 2018 de către Moxie Marlinspike și Brian Acton (cofondatorul aplicației *Whatsapp*), aceasta a fost construită specială să nu colecteze sau să stocheze date personale ale utilizatorilor, astfel aducând un nivel ridicat de securitate aplicației.

Din păcate, aceasta are un număr de 40 de milioane de utilizatori activi, relativ mic în comparație cu giganții *Whatsapp* sau *Messenger*. Numărul scăzut de utilizatori se datorează faptului că nu a fost investită o sumă mare de bani în partea de creare a publicității, dar oameni care chiar doresc o aplicație ce de la început a fost creată cu scopul de a menține intimitatea digitală a utilizatorului, cu o simplă căutare pe *Google* cum ar fi : „cea mai securizată aplicație de mesagerie”, vor găsi link-uri care îi va duce către aplicația Signal.[ (Wikipedia, 2023)]

A screenshot of a chat

Description automatically generated

Figură 2 (Aplicația Signal)

Cea de-a treia aplicație de mesagerie în timp real este *Messenger* [Figură 3], o aplicație dezvoltată de către *Meta,* în anul 2011. Această aplicație nu a avut nevoie de reclamă, deoarece în momentul apariției, *Facebook* avea deja 800 de milioane de participanți ai platformei activi, astfel *Messenger* a avut un start promițător.

În prezent, numărul lor nu a crescut foarte mult, de la 800 de milioane de utilizatori acum a ajuns în jurul cifrei de un miliard. Acest lucru se datorează proceselor în care Mark Zuckerberg (fondatorul *Facebook* și ulterior fondatorul *Meta*) a dat datele utilizatorilor către *Cambridge Analytica* fără acordul acestora, cu scopul de a fi folosite în publicitate politică.[ (Wikipedia, 2023)]

A screenshot of a computer

Description automatically generatedAceastă aplicație are un sistem de criptare end-to-end, dar nu este implicit activ pentru fiecare conversație. Pentru a activa acest lucru utilizatorii trebuie să selecteze conversația pe care ar dori să o cripteze. Acesta este un mare dezavantaj față de competitorii săi *Signal* și *Whatsapp.*

Figură 3 (Aplicația Facebook Messenger)

## 2.3 Comparație între Facebook Messenger, Whatsapp și Signal

Aceste trei aplicații prezintă atât avantaje cât și dezavantaje cum ar fi:

* *Whatsapp* este o aplicație foarte populară, încât este foarte improbabil să nu găsim un prieten căruia să îi dăm mesaj, este o aplicație securizată, în schimb acum este sub tutela *Meta* care de-a lungul timpului a demonstrat că nu este interesată de datele private ale utilizatorilor, iar aplicația nu are sursă deschisă astfel nu puteam afla ce este în spatele acestei interfețe.
* *Signal* este o aplicație mult mai puțin populară față de *Facebook Messenger* și *Whatsapp,* dar este o aplicație concepută exact cu scopul păstrării intimității digitale a utilizatorilor săi și cu un nivel de securitate ireproșabil. Un mare plus al acestei aplicații este faptul că nu are absolut nimic de ascuns, codul sursă al aplicației fiind public.
* *Facebook Messenger*, chiar dacă are un număr mare de utilizatori, a reușit să dezamăgească de multe ori, atât în performanța propriu-zisă a aplicației cât și în păstrarea datelor utilizatorilor în siguranță.

O scurtă concluzie ar fi următoarea, dacă o persoană dorește o simplă aplicație de mesagerie fară a fi interesat de securitatea datelor personale, aceasta poate alege aplicația *Facebook Messenger.*

În schimb, dacă acesta dorește o aplicație în care probabilitatea ca grupul său de muncă, de prieteni sau chiar familia să fie ușor accesibilă, iar datele sale să fie într-o oarecare siguranță, acesta poate alege aplicația *Whatsapp*.

Dacă persoana dorește totuși un nivel mai ridicat de securitate față de *Whatsapp* sau *Facebook Messenger* și o asigurare a faptului că datele sale nu vor fi utilizate în alt scop decât acela de a facilita comunicarea între oameni, aplicația perfectă pentru această persoană va fi *Signal.*

# Tehnologiile software utilizate

În acest capitol voi descrie toate tehnologiile utilizate pentru dezvoltarea aplicației ChitChat și motivul alegerii acestora.

## 3.1 Visual Studio Code

Pentru proiectarea aplicației, am folosit editorul de cod Visual Studio Code deoarece am avut acces la o multitudine de plugin-uri dedicate dezvoltării web, cum ar fi: *Prettier* pentru formatarea codului HTML, *Auto Close Tag* pentru a închide automat tag-urile folosite în HTML și pentru schimbarea în paralel a acestora, *ES7+ React/Redux/React-Native snippets* deoarece oferă prefixe scurte pentru o dezvoltare mai rapidă a proiectului și *Image Preview,* extensie ce afișează imaginile.

În acest editor de cod, avem extensia *Github Pull Request and Issues* pe care am folosit-o pentru gestionarea versiunilor aplicației. Această extensie s-a dovedit a fi foarte utilă în implementarea aplicației ChitChat.

## 3.2 JavaScript

JavaScript este principalul limbaj de programare utilizat în dezvoltarea de aplicații software, conceput inițial pentru a adăuga interactivitate paginilor Web și devenind ulterior unul dintre cele mai folosite si versatile limbaje de programare. Am ales să folosesc acest limbaj deoarece cu ajutorul mediului de execuție *Node.js* pot rula cod JavaScript și în afara unui browser, mai exact în partea de back-end a aplicației, iar cu ajutorul framework-ului *Express.js* dezvoltarea server-ului este mai rapidă și mai eficientă pentru transmiterea datelor în timp real.

## 3.3 React.js

React.js, dezvoltată și întreținută de *Facebook* și de către o comunitate de dezvoltatori care contribuie constant cu actualizări ale problemelor existente sau cu funcții noi, este una dintre cele mai populare biblioteci pentru dezvoltarea front-end-ului unei aplicații web.

Am ales să folosesc această bibliotecă deoarece pot împărți interfața utilizatorului în componente reutilizabile și autonome, având propria logică și stare, astfel facilitând mentenanța aplicației.

Totodată, biblioteca *React.js* este perfectă pentru aplicația ChitChat, deoarece mecanismele de redare ale acestei biblioteci sunt create pentru a face interfața utilizatorului reactivă și dinamică, în care mesajele trimise de utilizatori să fie afișate în timp real.

## 3.4 Node.js

Node.js este un mediu de execuție cu sursă deschisă, multiplatformă și cu un singur thread, creat pentru a face dezvoltarea aplicațiilor *Web* mult mai rapidă și mai scalabilă. Acesta rulează pe motorul de execuție creat de către *Google* numit: *JavaScript V8.*

Am ales să folosesc Node.js datorită modelului *I/O non-blocant,* bazat pe evenimente, astfel încât face dezvoltarea unei aplicații de comunicare în timp real mult mai facilă. Un alt motiv pentru care folosesc Node.js a fost managerul de pachete (*Node Package Manager*), de unde am putut adăuga biblioteci pentru urmatoarele caracteristici: criptarea parolelor utilizatorilor (*bcrypt*), afișarea câmpului de emoticoane (*emoji-picker-react*), compresia imaginilor de profil si a celor trimise de către utilizatori (*browser-image-compression*), trimiterea mesajelor în timp real (*Socket.io*), etc. Un alt aspect important al folosirii acestui mediu de execuție este faptul că pot dezvolta aplicația *Chitchat* fără a fi nevoie să folosesc un alt limbaj de programare înafară de *JavaScript.*

## Express.js

Express.js este un framework creat pentru mediul de execuție Node.js. Am ales să folosesc acest framework deoarece simplifică procesul de dezvoltare al aplicațiilor *Web* prin intermediul unui set de caracteristici și instrumente pentru rutare, middleware și gestionarea cererilor de tip *HTTP.*

## Git

Git este cel mai utilizat sistem modern de control al versiunilor unei aplicații din lume, creat în anul 2005 de către faimosul Linus Torvalds. Am ales să utilizez acest sistem de control al versiunilor pentru a ține sub control evoluția aplicației *Chitchat.* Acesta îmi permite să salvez modificările aduse aplicației și în cazul în care aplicația prezintă probleme după adaugarea unor noi caracteristici mă pot întoarce la o versiune anterioară a aplicației. În timpul dezvoltării aplicației Chitchat am creat 3 ramuri: *master* (unde aveam versiunea stabilă a aplicației), *features* (unde adăugam și testam noi caracteristici ale aplicației) și *refactoring* (unde am refactorizat codul pentru a aduce o performanță îmbunatățită la finalul aplicației).

## Sass

Sass (*Syntactically Awesome Style Sheet*) este un preprocesor interpretat și compilat în *CSS*. A fost creat pentru a face codul de *CSS* mai ușor de întreținut. Acesta oferă caracteristici suplimentare față de *CSS* prin: crearea unor variabile unde putem stoca valori utilizate cum ar fi culori, font-uri și dimensiuni. Cu ajutorul acestuia am creat design-ul aplicației, mai exact așezarea în pagină a elementelor *HTML*, culoarea acestora și animațiile pentru a face experiența utilizatorului mai plăcută.

## MongoDB

MongoDB este o bază de date NoSQL cu sursă deschisă, concepută pentru a stoca documente într-un format asemănător cu cel al *JSON*-ului (*JavaScript Object Notation*)*.* Am folosit această bază de date pentru aplicație deoarece este ușor de integrat cu *Express.js* și *Socket.io,* și suportă actualizările de date în timp real.

1. Baza de date

Ca bază de date am ales să folosesc *MongoDb*, care este o bază de date de tip *NoSQL*. Pentru aplicații cu funcționalități în timp real cum este aplicația *Chitchat,* bazele de date *NoSQL* sunt mai potrivite decât bazele de date relaționale, deoarece pot fi scalate orizontal (mai multe instanțe pot fi create în funcție de cererea computațională a aplicației) și pot stoca date cu ușurință pentru că datele sunt nestructurate.[Figură 4]

4.1 Modelul utilizatorilor

Pentru a salva datele utilizatorilor, am creat un model în baza de date unde am următoarele câmpuri:

* Username : unde salvez numele introdus de utilizator, setez tipul de dată pe care să îl accepte și fac acest cămp obligatoriu și unic pentru crearea unui cont.
* Email : unde salvez email-ul introdus de utilizator, setez tipul de dată pe care să îl accepte și ca la câmpul *Username,* fac acest câmp obligatoriu și unic pentru crearea contului.
* Password : unde salvez parola *hashed* introdusă de utilizator, îi definesc tipul de dată pe care să îl accepte și fac acest cămp obligatoriu.
* ProfilePicture : unde salvez poza de profil a utilizatorului, tipul de dată acceptat este *String* deoarece *FileReader()* returnează un string în bază 64 cu conținutul imaginii, iar acest câmp este predefinit cu un șir de caractere nul.
* IsProfilePictureSet : unde salvez dacă utilizatorul a ales o poză de profil diferită față de cea standard, această variabilă atribuindu-se valoarea *true* după introducerea unei poze de profil în baza de date.

A screen shot of a computer program

Description automatically generated

Figură 4 (Modelul User)

Am introdus suplimentar opțiunea *timestamps* care adaugă 2 automat două câmpuri, acelea fiind *createdAt* și *updatedAt* pentru a putea vedea cănd a fost creat contul sau actualizat ultima oară.

4.2 Modelul mesajelor

Pentru a salva mesajele trimise de utilizatori, am creat un alt model [Figură 5] în baza de date cu următoarele câmpuri:

* Message : aici salvez proprietatea text în care introduc mesajul trimis de utilizator
* Users : aici salvez un vector ce conține numarul de identificare al celor 2 utilizatori care poartă o conversație și creez o referință către modelul utilizatorului.
* Sender : aici salvez numărul de identificare al celui care trimite mesajul și creez o referință către modelul utilizatorului.

Am introdus aici opțiunea *timestamps* pentru a putea ordona mesajele schimbate între utilizatori în sens crescător, astfel mesajele cele mai recente apar în partea de jos a componentei *Messages.*

A screen shot of a computer program

Description automatically generated

Figură 5 (Modelul Message)

# Aplicația Chitchat

În acest capitol voi prezenta cum a fost creată aplicația *Chitchat*, din punct de vedere al dezvoltării și al arhitecturii acesteia.

## 5.1 Introducere

Aplicația Chitchat a pornit de la ideea creării unei unelte simple, un spațiu de comunicare și interschimbare de informații, cu o interfață care asigură o navigare eficientă și plăcută. Aceasta este compusă din 2 părți:

* Aplicația de tip client
* Aplicația de tip server

*Aplicația de tip client* reprezintă interfața grafică prin care utilizatorii vor putea comunica între ei. Aici găsim 4 pagini cu care utilizatorul poate interacționa, acestea fiind:

* pagina de conectare unde se poate conecta daca are deja un cont creat,
* pagina de înregistrare unde poate crea un cont,
* pagina de selectare a pozei de profil unde își poate alege poza de profil după ce s-a înregistrat
* pagina principală unde se află contactele acestuia, conversațiile, un buton de ștergere a conversației și un meniu de setări (unde acesta își poate schimba poza de profil sau numele său de utilizator, își poate șterge contul în cazul în care nu mai dorește să fie activ în cadrul acestei aplicații și un buton de deconectare).

*Aplicația de tip server* reprezintă partea din spate, ce primește date prin intermediul modulelor de rutare și de validare create în *Express.js* pentru a putea utiliza caracteristicile descrise mai sus în aplicația de tip client.

## 5.2 Cerințe funcționale ale aplicației

Principalele cerințe funcționale ale aplicației *Chitchat*:

* Facilitarea creării unui cont de utilizator sau conectarea la un cont deja existent
* Primirea și transmiterea datelor între client și server în timp real
* Stocarea datelor într-o bază de date pentru accesarea ulterioară a acestora
* Eficiența în rulaj pentru a asigura o navigare plăcută și rapidă

## 5.3 Dezvoltarea aplicației

În acest capitol voi discuta cele 4 pagini ale aplicației și anume: pagina de conectare, pagina de înregistrare, pagina de alegere a pozei de profil și pagina principală unde utilzatorii vor putea comunica. Fiecare pagină va avea 2 subcapitole:

1. Front-end
2. Back-end

În aceste subcapitole voi descrie design-ul și controlerele fiecărei pagini în parte.

### 5.3.1 Pagina de conectare

Pagina de conectare este pagina cu care utilizatorul va avea contact în primă fază, unde în cazul în care are deja un cont creat în cadrul aplicației acesta trebuie să introducă adresa de email cu care s-a înregistrat și parola aferentă contului. [Figură 6]

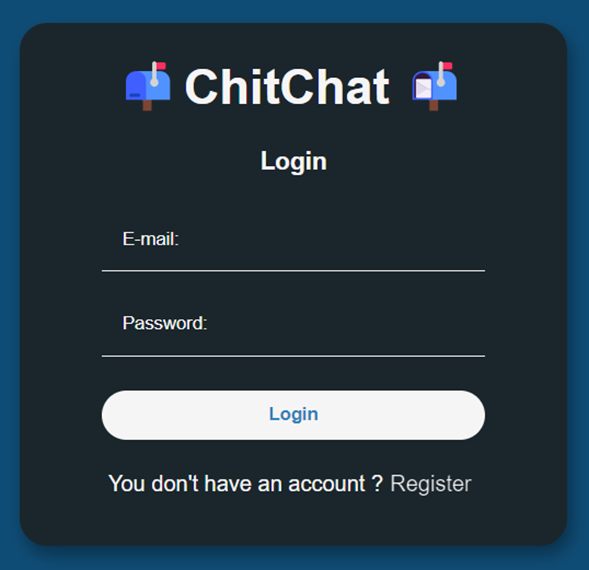
#### Front-end

Pentru formularul de conectare, am creat un chenar gri închis în centrul ecranului în care utilizatorul poate introduce email-ul și parola contului pentru a accesa pagina principală a aplicației.

În această pagină se găsește un design simplu și intuitiv. Cu două câmpuri în care utilizatorul trebuie să introducă datele, iar fiecare câmp are un *placeholder* care descrie ce anume trebuie introdus în acesta.

Pentru a prelua datele introduse de utilizator, pentru fiecare câmp este introdus evenimentul *onChange* care se activează în momentul în care cămpul este modificat. Datele din câmp sunt după adăugate într-un obiect cu ajutorul funcției *useState()*.[Figură 11]

Datele introduse sunt ulterior validate în funcția *handleValidation()* unde se verifică dacă utilizatorul a respectat următoarele restricții: ambele câmpuri să fie completate, email-ul să aibă o lungime cuprinsă între 8 și 50 de caractere, iar parola să aibă o lungime cuprinsă între 8 și 30 de caractere.[Figură 12]



Figură 6 (Pagina de conectare)

În cazul în care utilizatorul nu respectă restricțiile impuse acesta va primi o notificare în colțul din dreapta-sus al paginii cu condițiile care trebuiesc îndeplinite, iar în cazul în care nu există un cont cu datele introduse, acesta va primi o eroare în care i se va spune că email-ul sau parola sunt incorecte.[Figură 7, Figură 8, Figură 9, Figură 10]

A black and white sign with white text

Description automatically generated

Figură 7 (Notificare Eroare Conectare 1)

A black and white sign with white text

Description automatically generated

Figură 8 (Notificare Eroare Conectare 2)

A black and orange rectangle with white text

Description automatically generated

Figură 9 (Notificare Eroare Conectare 3)

A close-up of a logo

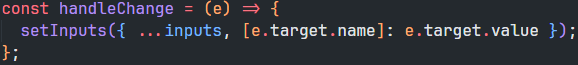
Description automatically generated

Figură 10 (Notificare Eroare Conectare 4)

Dacă utilizatorul are un cont existent în baza de date și datele introduse de acesta sunt conforme cu restricțiile impuse, atunci utilizatorul prin intermediul funcției *handleSubmit()* va trimite datele introduse pe ruta [*/server/authentication/login*](http://localhost:4000/server/authentication/login)și vor ajunge în controlerul creat pentru conectare. În cele din urmă datele acestuia vor fi salvate în *sessinStorage*.[Figură 13]

Dacă utilizatorul nu are un cont existent, în partea de jos a formularului am creat un link, care odată accesat trimite utilizatorul pe pagina de înregistrare unde acesta își poate crea un cont.

A screen shot of a computer program

Description automatically generated

Figură 11 (Funcția handleChange() din pagina de conectare)

Figură 12 (Funcția handleValidation() din pagina de conectare)

A screen shot of a computer program

Description automatically generated

Figură 13 (Funcția handleSubmit() din pagina de conectare)

#### Back-end

Pentru a valida datele introduse de către utilizator, am creat un controler numit *login* unde caut în baza de date dacă există email-ul introdus de utilizator. Dacă nu există email-ul în baza de date, atunci trimit un *JSON* către front-end cu mesajul erorii și statusul *false* pentru a opri conectarea la aplicație, iar în caz contrar verific parola introdusă de utilizator cu cea aferentă contului creat. Dacă parola introdusă de utilizator este greșită voi trimite din nou un *JSON* către utilizator cu mesajul erorii și statusul *false.*[Figură 14]

Dacă și email-ul și parola sunt valide, atunci trimit un *JSON* cu statusul *true* și cu datele acestuia, astfel încât datele sunt introduse în *sessionStorage* și utilizatorul este trimis către pagina principală a aplicației. Pentru fiecare câmp avem câte un *placeholder* pentru a informa utilizatorul cu privire la ce date trebuiesc introduse în fiecare câmp.

A screen shot of a computer code

Description automatically generated

Figură 14 (Controler pentru conectarea în aplicație)

### 5.3.2 Pagina de înregistrare

Pagina de înregistrare este pagina unde oamenii își pot face un cont pentru a accesa aplicația *Chitchat.* Dacă aceștia au deja un cont activ, atunci pot accesa pagina de conectare prin link-ul atașat in partea de jos a chenarului.

#### Front-end

Pentru formularul de înregistrare, am creat un chenar gri închis în centrul ecranului în care utilizatorul trebuie să introducă un nume, un email, o parolă și confirmarea parolei introduse.[Figură 15]

Design-ul acestei pagini este asemănător cu design-ul paginii de conectare descrisă în capitolul anterior. Spre deosebire de pagina de conectare, avem prezente 2 câmpuri suplimentare, acelea fiind: numele și confirmarea parolei.

Pentru a prelua datele introduse de utilizator, pentru fiecare câmp am introdus evenimentul *onChange* care se activează în momentul în care cămpul este modificat. Datele din câmp sunt după adăugate într-un obiect cu ajutorul funcției *useState()*.

Datele introduse sunt ulterior validate în funcția *handleValidation()* unde verificăm dacă utilizatorul a respectat următoarele restricții: toate câmpurile să fie completate, email-ul sa aibă o lungime cuprinsă între 8 și 50 de caractere, parola să aibă o lungime cuprinsă între 8 și 30 de caractere și textul introdus în câmpul de confirmare al parolei să fie același cu textul introdus în câmpul parolă.A screenshot of a login form

Description automatically generated[Figură 23]

Figură 15 (Pagina de înregistrare)

În cazul în care utilizatorul nu respectă restricțiile impuse, acesta va primi o notificare în colțul din dreapta-sus al paginii cu condițiile care trebuiesc îndeplinite.

A black and white sign with white text

Description automatically generated

Figură 16 (Notificare eroare la înregistrare 1)

A black and orange sign with white text

Description automatically generated

Figură 17 (Notificare eroare la înregistrare 2)

A close-up of a black background

Description automatically generated

Figură 18 (Notificare eroare la înregistrare 3)

A black and white text

Description automatically generated

Figură 19 (Notificare eroare la înregistrare 4)

A black and orange rectangle with white text

Description automatically generated

Figură 20 (Notificare eroare la înregistrare 5)

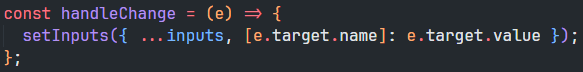
*A black and white sign with white text

Description automatically generated*

Figură 21 (Notificare eroare la înregistrare 6)

Dacă utilizatorul introduce datele conform restricțiilor impuse, atunci utilizatorul prin intermediul funcției *handleSubmit()* va trimite datele introduse pe ruta [*/server/authentication/register*](http://localhost:4000/server/authentication/register)și vor ajunge în controlerul creat pentru înregistrare. În cele din urmă datele acestuia vor fi salvate în *sessionStorage* și va fi trimis către pagina de alegere a pozei de profil. [Figură 24]

A screen shot of a computer program

Description automatically generated

Figură 23 (Funcția handleValidation() din pagina de înregistrare)

Figură 22 (Funcția handleChange() din pagina de înregistrare)

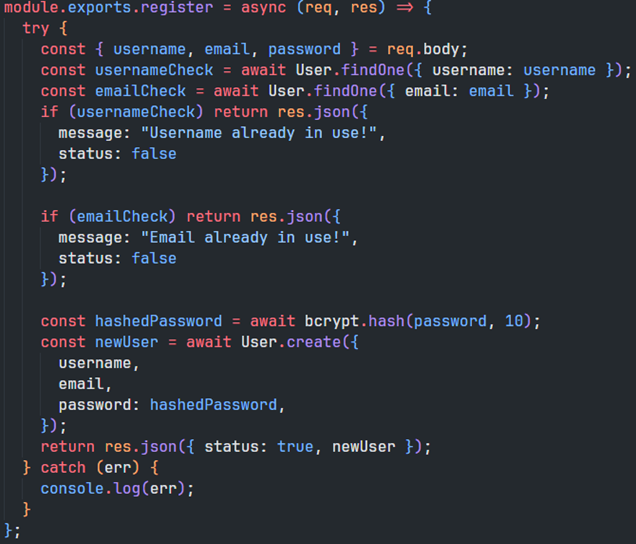
A screen shot of a computer code

Description automatically generated

Figură 24 (Funcția handleSubmit() din pagina de înregistrare)

#### Back-end

Pentru a valida datele introduse de către utilizator, am creat un controler numit *register* unde caut în baza de date dacă există email-ul sau numele introdus de utilizator. Dacă există email-ul sau numele în baza de date, atunci trimit un *JSON* către front-end cu mesajul erorii și statusul *false* pentru a opri crearea contului, iar în caz contrar, creez un nou cont cu datele introduse de utilizator. Cu ajutorul bibliotecii *bcrypt*, fac un *hash* parolei introduse de utilizator pentru a spori siguranța datelor din cadrul aplicației.[Figură 25]



Figură 25 (Controlerul de înregistrare)

### 5.3.3 Pagina de alegere a pozei de profil

Utilizatorii își pot alege o poza de profil pentru a putea fi recunoscuți mai ușor de ceilalți utilizatori ai aplicației. Aici pot face două alegeri, ori să păstreze poza predefinită a aplicației de profil, ori să aleagă una din calculatorul lor.

#### Front-end

După parcurgerea de înregistrare a contului, utilizatorul este trimis către această pagină, unde este întâmpinat de un chenar mai mare față de paginile de înregistrare și conectare, cu 3 butoane, unul în partea din stânga-jos a chenarului cu textul „*Skip*”, unul în partea din dreapta-jos cu textul „*Finish*” și unul în partea de jos-centru cu textul „*Choose a profile picture*”.

Dacă utilizatorul nu dorește adăugarea unei poze de profil specifice, atunci acesta poate apăsa butonul *Skip* pentru a își pune poza standard a aplicației drept poză de profil. În caz contrar, acesta poate apăsa butonul *Choose a profile picture* și i se va deschide managerul de fișiere pentru a selecta o poză de profil.[Figură 26]

Odată selectată o poză de profil, aceasta va apărea în locul pozei standard din centrul ecranului, astfel utilizatorul poate hotărâ dacă păstrează sau nu acea poză de profil.

Utilizatorul poate apăsa butonul *Finish* dacă a selectat o poză de profil, astfel este trimis către pagina principală a aplicației.

A screenshot of a profile picture

Description automatically generatedÎn cazul în care utilizatorul nu a selectat o poză de profil și apasă butonul *Finish*, acesta va primi o notificare în colțul din dreapta-sus al paginii unde va fi informat că trebuie selectată o poză de profil pentru a apăsa butonul *Finish,* în caz contrar poate apăsa butonul *Skip.*

Figură 26 (Pagina de alegere a pozei de profil)

Pentru a deschide managerul de fișiere în momentul apăsării butonului *Choose a profile picture,* am introdus evenimentul *onClick.*

Pentru a afișa poza selectată de utilizator, am creat funcția *handleFileChange(),* care verifică dacă managerul de fișiere a fost închis fără a selecta o poză. În caz contrar, cu ajutorul bibliotecii *browser-image-compression* reduc dimensiunea pozei selectate de utilizator la un megabyte pentru a nu încetini performanța aplicației. Definesc un now *FileReader()* pentru a citi conținutul fișierului într-un mod asincron și pentru a actualiza variabila *photo* cu ajutorul funcției *useState()* în momentul în care am terminat de citit datele pozei.[Figură 27]

Butonul *Finish* poate fi acum utilizat pentru a trimite poza de profil către controlerul *profilePicture.* Odată apăsat acest buton, trimit datele pe ruta [*/server/authentication/profilePicture*](http://localhost:4000/server/authentication/profilePicture)*.* [Figură 30]

A screen shot of a computer program

Description automatically generated

Figură 27 (Funcția de prelucrare a pozei de profil)

A screen shot of a computer screen

Description automatically generated

Figură 28 (Funcția de deschidere al managerului de fișiere)

A screenshot of a computer program

Description automatically generatedA computer screen shot of a program code

Description automatically generated

Figură 30 (Funcția de trimitere a pozei de profil selectate în baza de date)

Figură 29 (Funcția de selectare a pozei standard)

#### Back-end

În partea din *back-end* a acestei pagini, preiau imaginea selectată de utilizator și *id-*ul acestuia, caut contul cu id-ul specificat și actualizez datele contului adăugând poza de profil în baza de date și modificând câmpul *isProfilePictureSet* prezent în baza de date din *false* în *true.*

După executarea acestor operații, trimit un *JSON* înapoi către partea de *front-end* a aplicației cu datele imaginii și modificarea câmpului *isSet,* pentru a putea trimite utilizatorul în pagina principală a aplicației.[Figură 31]

A screen shot of a computer program

Description automatically generated

Figură 31 (Controlerul de prelucrare a pozei de profil)

### 5.3.4 Pagina principală

Pagina principală a aplicației este construită din 5 componente:

* *Navbar* : aici sunt afișate poza de profil a utilizatorului, numele acestuia și meniul de setări al aplicației.
* *Chats* : aici sunt afișate toate contactele utilizatorului prin poza de profil și numele acestora.
* *Messages* : aici sunt afișate mesajele și pozele schimbate de utilizatori
* *Input* : aici este câmpul de introducere al mesajului, butonul pentru selecția pictogramelor de expresie, butonul pentru selecția pozei pe care utilizatorul dorește să o trimită și butonul de trimitere al mesajelor.
* *SettingsMenu* : aici sunt afișate câteva setări pe care utilizatorul le poate accesa (cum ar fi: schimbarea pozei de profil, schimbarea numelui de utilizator, deconectarea din aplicație și ștergerea definitivă a contului).

Odată ce utilizatorul ajunge pe această pagină este informat să aleagă un contact pentru a începe o conversație, iar în momentul apăsării unui contact, componenta *Messages* va reda mesajele schimbate de aceștia în cazul în care există.[Figură 32]

A screenshot of a chat

Description automatically generatedA screenshot of a computer

Description automatically generated

Figură 32 (Pagina principală atunci când nu este selectat un contact)

Figură 33 (Pagina principală atunci când unul din contacte este selectat)

#### Front-end

Pentru a face posibilă selecția contactelor de către utilizator, am creat o funcție numită *handleChange()* care setează datele obiectului *selectedContact* cu datele contactului selectat de utilizator.[Figură 34 (Funcția de selectare a contactului)]

În partea din dreapta-sus a paginii, am creat un buton pentru ștergerea conversației dintre utilizator și contactul selectat. În momentul apăsării acestui buton, utilizatorului îi va apărea o fereastră de confirmare a deciziei luate. În cazul în care utilizatorul are un răspuns afirmativ, atunci voi trimite o cerere pe ruta */server/messages/deleteConversation* cu numărul de identificare al utilizatorului curent si al contactului selectat.[Figură 35]

A black background with colorful text

Description automatically generated

Figură 34 (Funcția de selectare a contactului)

A screen shot of a computer code

Description automatically generated

Figură 35 (Funcția de ștergere a conversației curente)

Cu ajutorul funcției *io* din biblioteca *socket.io-client,* creez o conexiune de tip *WebSocket* atribuită variabilei *socket.current,* această variabilă fiind o referință catre conexiunea WebSocket ce persistă și în cazul rerandărilor*.* Înainte de stabilirea conexiunii verific dacă există un utilizator curent în cazul apariției unei erori la conectarea în cadrul aplicației. După stabilirea conexiunii WebSocket, clientul emite evenimentul numit *join* către server*,* permițând utilizatorului să trimită și să primească mesaje sau imagini în timp real.[Figură 36]

A screen shot of a computer program

Description automatically generated

Figură 36 (Funcțiile pentru instanțierea socket-ului în client și pentru returnarea contactelor din listă)

#### Back-end

Pentru butonul *deleteConversation*, primesc datele în controlerul *deleteConversation* și șterg toate mesajele schimbate de cei doi utilizatori în funcție de numerele de identificare ale acestora. Odată terminată operația de ștergere a mesajelor, trimit un *JSON* înapoi în partea de *front-end* cu statusul *true.*[Figură 37]

*A screen shot of a computer code

Description automatically generated*

Figură 37 (Controler pentru ștergerea conversației curente)

Dacă există un utilizator curent, am creat evenimentul *join* prin care adaug într-un nou obiect *Map()* numărul de identificare al utilizatorului și numărul de identificare al *socket*-ului acestuia.[Figură 38]

A screenshot of a computer screen

Description automatically generated

Figură 38 (Funcția de conectare a utilizatorului la socket)

#### 5.3.5.1 Componenta Navbar

Prima componentă creată este bara de navigare din colțul din stânga-sus al paginii, unde sunt prezente poza de profil a utilizatorului, numele acestuia și meniul de setări.[Figură 39]

Poza de profil și numele utilizatorului sunt selectate din *sessionStorage,* din cadrul *browser*-ului. Dacă utilizatorul apasă poza de profil a sa, aceasta se mărește în centrul ecranului pentru a o vizualiza mai bine.

A blue background with white text

Description automatically generated

Figură 39 (Componenta Navbar)

Meniul de setări este o componentă separată, dar pentru a îl deschide și închide, am creat o varibila *isOpen,* de tip *boolean*,prin care verific dacă utilizatorul a apăsat butonul de setări. În cazul în care utilizatorul apasă acest buton, variabila va lua valoarea sa negată, astfel dacă are valoarea *false* aceasta va deveni *true* pentru a deschide meniul, iar la a doua apăsare se va întoarce înapoi la valoarea *false* pentru a închide meniul.[Figură 40]

Valoarea variabilei *isOpen* este trimisă prin sistemul de proprietăți disponibil în *React.js*, pentru a face posibilă modularizarea aplicației.

A screen shot of a computer code

Description automatically generated

Figură 40 (Funcțiile de deschidere a meniului și pentru mărirea pozei de profil)

A screen shot of a computer program

Description automatically generated

Figură 41 (JSX-ul pentru componenta Navbar)

#### 5.3.5.2 Componenta SettingsMenu

În acestă componentă avem setările disponibile utilizatorilor, acestea fiind: schimbarea pozei de profil, schimbarea numelui, deconectarea din aplicație și ștergerea permanentă a contului de utilizator.[Figură 42]

##### Front-end

Pentru meniul de setări, am creat un chenar de culoare albastră ce conține 4 butoane:

* *Change profile picture*
* *Change Username*
* *Log Out*
* *Delete Account*

A screen shot of a blue screen

Description automatically generated

Figură 42 (Meniul de setări)

Primul buton este *Change Profile Picture*. Pentru momentul în care utilizatorul apasă acest buton, am creat un eveniment numit *handleProfilePictureChange()* prin care verific dacă utilizatorul a apăsat acest buton. În cazul în care acest buton este apăsat, utilizatorului i se va deschide managerul de fișiere pentru a selecta o poză din calculatorul său.

Pentru a gestiona selectarea unei poze de către utilizator, am creat funcția *handlePictureChange()* prin care preiau poza selectată de utilizator, micșorez dimensiunea acesteia cu ajutorul bibliotecii *browser-image-compression* la un megabyte și creez un nou *FileReader()* pentru a citi datele pozei. Dupa procesarea datelor, trimit poza pe ruta [*/server/authentication/changeProfilePicture*](https://localhost:4000/server/authentication/changeProfilePicture)pentru a actualiza datele din baza de date.[Figură 43]

A screen shot of a computer program

Description automatically generated

Figură 43 (Funcția de schimbare a pozei de profil)

Al doilea buton din meniul de setări este *Change Username.* Dacă utilizatorul dorește să schimbe numele din aplicație, apăsând acest buton, în partea de sus a paginii va apărea un prompt unde acesta își poate introduce noul nume dorit.[Figură 44]

A screenshot of a computer

Description automatically generated

Figură 44 (Prompt pentru modificarea numelui de utilizator)

Noul nume introdus trebuie să respecte restricțiile impuse și la crearea contului de utilizator, acelea fiind : noul nume ales să aibă o lungime între 3 și 30 de caractere, plus o condiție suplimentară de a fi diferit de numele ales anterior.

Pentru a face toate acestea posibile, am creat o variabilă în care stochez textul din prompt introdus de utilizator. Pentru a obliga respectarea restricțiilor de către utilizatori, am creat un *while()* care va alerta utilizatorul în cazul în care nu a respectat restricțiile până când acestea sunt îndeplinite.

Dacă utilizatorul a respectat condițiile schimbării numelui, atunci voi trimite noul nume pe ruta [*/server/authentication/changeUsername*](http://localhost:4000/server/authentication/changeUsername)pentru a actualiza numele în baza de date. Totodată voi actualiza datele din *sessionStorage* și voi reîncărca pagina principală ca noile schimbări să ia efect.[Figură 45]

A computer screen shot of text

Description automatically generated

Figură 45 (Funcția de modificare a numelui de utilizator)

Al treilea buton este *Log Out.* În cazul în care utilizatorul dorește să se deconecteze din aplicația *Chitchat,* atunci poate apăsa acest buton. Pentru a deconecta utilizatorul din aplicație, am creat o funcție numită *handleLogOut()* prin care șterg datele utilizatorului curent din *sessionStorage* și reîncarc pagina, astfel utilizatorul va fi trimis înapoi pe pagina de conectare.[Figură 46]

A screen shot of a computer program

Description automatically generated

Figură 46 (Funcția de deconectare din aplicație)

Ultimul buton creat este *Delete Account.* În cazul în care utilizatorul nu mai dorește să fie activ în cadrul aplicației *Chitchat,* acesta poate selecta acest buton pentru a își șterge contul. În momentul apăsării, am creat o fereastră de confirmare în care întrb utilizatorul dacă este sigur de alegerea făcută. În cazul în care acesta este de acord, trimit o cerere către controlerul *deleteAccount* din *back-end* pentru a șterge contul acestuia,voi șterge datele acestuia din *sessionStorage* și la finalul operațiunii voi reîncărca pagina, astfel luând efect schimbările făcute.[Figură 47, Figură 48]

A screenshot of a computer

Description automatically generated

Figură 47 (Fereastră de confirmare pentru ștergerea contului)

A screen shot of a computer code

Description automatically generated

Figură 48 (Funcția de ștergere a contului)

##### Back-end

A screen shot of a computer code

Description automatically generatedControlerul *changeProfilePicture()* primește cererea de schimbare a pozei de profil a utilizatorului și îl caută în baza de date în funcție de numarul de identificare al acestuia. Odată găsit contul cu numărul de identificare asociat, modificam vechea poză de profil a utilizatorului cu ce nouă selectată de acesta și returnăm un *JSON* cu noua poză a utilizatorului și statusul variabilei *isSet(true)* pentru confirmarea succesului modificărilor.[Figură 49]

Figură 49 (Controler pentru modificarea pozei de profil)

A screen shot of a computer code

Description automatically generated Controlerul *changeUsername()* primește cererea de modificare a numelui utilizatorului și caută în baza de date un cont asociat cu numărul de identificare al acestuia. Odată găsit un cont cu numarul de înregistrare specificat, actualizăm numele vechi al utilizatorului cu cel nou introdus de acesta. Dacă operația are succes, returnăm un *JSON* cu noul nume introdus. [Figură 50]

Figură 50 (Controler pentru modificarea numelui de utilizator)

Controlerul *deleteAccount* *()* primește cererea de șterge a contului cu numarul de identificare al utilizatorului și caută un cont cu același număr de identificare. Odată găsit contul utilizatorului în baza de date, ștergem contul acestuia și returnăm un *JSON* cu statusul *true* pentru a confirma succesul operației.[Figură 51]

A screen shot of a computer code

Description automatically generated

Figură 51 (Controler pentru ștergerea contului de utilizator)

#### 5.3.5.3 Componenta Chats

În această componentă utilizatorul poate vedea toți utilizatorii disponibili, și îi poate accesa pentru a purta o conversație cu aceștia.[Figură 52]

Pentru fiecare contact din listă am afișat poza de profil și numele pentru a face identificarea acestora mai ușoară.

Când utilizatorul duce cursorul mouse-ului peste unul din contacte, acesta își va schimba culoarea print-o animație pentru a evidenția contactul pe care acesta dorește să îl acceseze și când acesta apasă pe unul din contacte, culoarea va rămâne schimbată până la selectarea altui contact din listă.

A screenshot of a phone

Description automatically generated

Figură 52 (Componenta Chats)

##### Front-end

Pentru redarea tuturor contactelor disponibile în aplicație, am folosit funcția *map()* din *JavaScript* pentru vectorul *contacts* prin care preiau datele fiecărui utilizator mai puțin cele ale utilizatorului curent pentru a afișa poza de profil și numele acestora și pentru a atribui câte un index fiecărui utilizator.[Figură 53]

Astfel, cand utilizatorul apasă unul din contactele din listă, prin intermediul index-ului atribuit, acesta își va schimba culoarea pentru a evidenția faptup că el este cel cu care se poartă conversația și va trimite datele contactului cu index-ul selectat pentru a afișa mesajele corespunzătoare conversației selectate.

A screen shot of a computer program

Description automatically generatedCând utilizatorul ajunge pe pagina principală, contactele au o întărziere în redare până cand funcția *map()* folosită returnează vectorul de contacte. Pentru această problemă, am creat o animație de încărcare a listei de contacte. În momentul în care vectorul este returnat de către funcția *map()*, animația se oprește iar contactele sunt afișate în listă.

Figură 53 (Funcția de schimbare a contactului selectat și JSX-ul componentei)

A screen shot of a computer code

Description automatically generated

Figură 54 (Componenta auxiliară pentru definirea fiecărui contact)

##### Back-end

Pentru a face posibilă randarea tuturor contactelor din baza de date, în componenta *Home* am trimis o cerere către controlerul *getUsers()* împreună cu numarul de identificare al utilizatorului curent pentru a căuta în baza de date toți utilizatorii mai puțin pe cel curent. Odată găsiți toți utilizatorii, returnez un vector cu email-ul, numele, poza de profil și numărul de identificare înapoi în componenta *Home* print-un *JSON* și după aceea, prin sistemul de proprietăți disponibil în *React.js,* trimit acest vector către componenta *Chats.*[Figură 55]

*A screen shot of a computer code

Description automatically generated*

Figură 55 (Controler pentru returnarea tuturor contactelor)

#### 5.3.5.4 Componenta Messages

În această componentă, în urma selectării unui contact, utilizatorul poate vedea mesajele și pozele schimbate cu alți utilizatori.[Figură 56]

Mesajele trimise de utilizatorul curent sunt afișate în partea din dreapta a chenarului, iar mesajele primite de acesta sunt afișate în partea din stânga.

În cazul în care utilizatorul dorește să trimită un mesaj mai lung față de lățimea maximă setată a mesajului (80%), atunci mesajul va fi împărțit în câte rânduri este nevoie.

A screenshot of a chat

Description automatically generated

Figură 56 (Componenta Messages)

##### Front-end

În această componentă, utilizatorul poate vedea mesajele schimbate cu contactele sale, cele din dreapta componentei fiind mesajele trimise de către utilizatorul curent și cele din stânga componentei fiind mesajele primite de utilizatorul curent.

Pentru a putea trimite un mesaj, am creat o funcție numită *handleSendMessage()* care ia ca parametru mesajul preluat din componenta *Input.* În variabila *currentUser* stochez datele din *sessionStorage* și ulterior creez un obiect numit *data* în care introduc numărul de identificare al utilizatorului curent, numărul de identificare al contactului selectat și conținutul mesajului trimis.

După aceea emit datele stocate în obiect către evenimentul *sendMessage*, pentru a crea transmiterea de mesaje în timp real și către ruta */server/messages/sendMessage* pentru a salva mesajul în baza de date.

Biblioteca *React.js* nu poate actualiza un obiect dacă are aceeași referință în memorie, astfel, când trebuie actualizat obiectul, creez un nou obiect pentru a avea o referință diferită față de obiectul inițial, în care copiez datele vechi și le actualizez cu cele noi.

Pentru a reda mesajele schimbate de utilizatorul curent și contactul selectat, am folosit funcția *useEffect()* disponibilă în *React.js* prin care trimit o cerere pe ruta /server/messages/getMessages către controlerul *getMessages().* Dacă avem date de returnat în urma operației, actualizăm datele din vectorul *messages* cu cele dintre utilizatorul curent și contactul selectat.[Figură 60]

Pentru a actualiza variabila *messages* de fiecare dată când unul dintre utilizatori trimite un mesaj, am creat un nou vector în care copiez datele anterioare și adaug noul mesaj. Este nevoie de o astfel de abordare deoarece, ca în cazul obiectelor, dacă referința în memorie este aceeași, atunci *React.js* nu va reranda componenta.

În cele din urmă, pentru a menține focusul în partea de jos a componentei, am folosit o funcție de tip *useEffect()* prin care verific dacă avem un nou mesaj primit sau trimis și în cazul în care avem unul, cu ajutorul funcției *scrollIntoView,* pentru a menține focusul în partea de jos a componentei unde sunt cele mai recente mesaje. Suplimentar, am adăugat și parametrul *behaviour* cu valoarea *smooth* pentru a face schimbarea focusului la mesaje noi mai lină, oferind un sentiment de fluiditate utilizatorilor.[Figură 62]

A computer screen shot of text

Description automatically generatedA screen shot of a computer

Description automatically generated

Figură 58 (Componenta auxiliară pentru conținutul mesajului)

Figură 57 (Componenta auxiliară afișată când niciun contact nu este selectat)

A screen shot of a computer code

Description automatically generated

Figură 59 (Funcția pentru trimiterea mesajelor)

A screen shot of a computer code

Description automatically generated

Figură 60 (Funcția pentru returnarea mesajelor dintre 2 utilizatori)

A screen shot of a computer code

Description automatically generated

Figură 61 (Funcția pentru afișarea mesajelor primite)

A screen shot of a computer code

Description automatically generated

Figură 62 (Funcțiile pentru actualizarea vectorului de mesaje și respectiv schimbarea focusului la ultimul mesaj primit)

##### Back-end

Pentru început, preiau datele trimise de către *front-end* în funcția *sendMessage()* cu argumentul *socket* și returnez o funcție cu parametrii următori: numărul de identificare al utilizatorului curent, numărul de identificare al contactului selectat și mesajul. După aceea, căutăm în colecția de utilizatori activi, utilizatorul căruia trebuie să îi trimitem mesajul.

Pe ruta */server/messages/sendMessage* am creat un controler pentru a salva mesajele trimise în baza de date în funcție de numărul de identificare al utilizatorului curent și numărul de identificare al contactului selectat. După aceea, creez un mesaj nou în baza de date cu următoarele informații: mesajul trimis, un vector care contine numerele de identificare ale utilizatorilor, și numarul de identificare al celui care a trimis mesajul.

A computer screen shot of text

Description automatically generated Pentru a afișa mesajele existente în momentul selectării unui contact, am creat un controler numit *getMessages()* prin care caut toate mesajele schimbate de cei doi utilizatori. Odată găsite mesajele, le ordonez crescător astfel cele mai recente mesaje vor fi în partea de jos. După găsirea mesajelor și sortarea lor, verific dacă utilizatorul curent este sau nu utilizatorul pentru a decide dacă mesajul va fi pus în partea din dreapta sau partea din stânga a conversației, iar la final returnez *JSON*-ul cu un vector de obiecte.[Figură 64]

Figură 63 (Controler pentru trimiterea mesajelor în baza de date)

A screen shot of a computer code

Description automatically generated

Figură 64 (Controler pentru returnarea mesajelor din baza de date)

#### 5.3.5.5 Componenta Input

În acestă componentă, avem un câmp unde utilizatorul poate introduce mesajul pe care dorește să-l transmită. Acesta poate adăuga și emoticoane în mesajul său prin folosirea butonului de emoticoane din dreapta câmpului de scriere.[Figură 65]

Totodată acesta poate trimite și poze prin apăsarea butonului de selecția a pozelor, unde i se va afișa managerul de fișiere de unde poate selecta poza pe care dorește să o trimită.

În partea din dreapta a componentei se află butonul de send, iar atunci cănd utilizatorul duce cursorul mouse-ului deasupra săgeții, aceasta se va înclina la 90 de grade către dreapta, printr-o animație lină. Utilizatorul poate trimite mesajul și fără a apăsa săgeata de trimitere, prin apăsarea tastei *Enter* de la tastatură.

A blue rectangular object with white border

Description automatically generated

Figură 65 (Componenta Input)

##### Front-end

Pentru câmpul de introducere al textului, am adăugat o funcție de verificare a modificărilor câmpului, și le salvez intr-o variabilă numită *message.* Pentru momentul în care utilizatorul dorește să trimită mesajul, mesajul este transmis prin sistemul de proprietăți către componenta părinte *Messages,* unde se află funcția *handleSendMessage().[*Figură 66 (Funcțiile componentei Input)*]*

În cazul în care utilizatorul dorește să trimită o poză către un alt utilizator, acesta poate apăsa butonul de slectare a imaginii. Pentru a face acest lucru posibil, am creat o funcție numită *handlePictureChange()* prin care, cu ajutorul bibliotecii *browser-image-compression*, preiau poza selectată de utilizator, o reduc la o dimensiune de sub un megabyte și creez un nou *FileReader()* pentru a transforma poza în bază 64. Când *FileReader()* își termină execuția, trimit conținutul imaginii ca mesaj.[Figură 67]

A screen shot of a computer program

Description automatically generatedPentru a diferenția mesajul normal de poză, verific dacă mesajul conține textul *data:image,* atunci nu voi mai crea un paragraf, ci un tag de imagine pentru a o reda.

Figură 66 (Funcțiile componentei Input)

A computer screen with text and symbols

Description automatically generated

Figură 67 (Funcția pentru trimiterea pozelor între utilizatori)

# Concluzii

Aplicația de comunicare în timp real *Chitchat* pe care am dezvoltat-o are ca scop facilitarea comunicării între oameni prin oferirea unei aplicații eficiente și rapide prin care utilizatorii pot păstra legătura trimițându-și mesaje și poze constant.

Obiectivele propuse inițial au fost îndeplinite deoarece utilizatorii au o interfață ușor de utilizat, își pot crea conturi în cadrul aplicației și pot comunica în timp real indiferent de distanța care îi desparte.

Am creat o interfață simplă și intuitivă pentru utilizatori, prin introducerea unui *placeholder* pentru fiecare câmp de introducere al unui text pentru a nu avea ambiguități, crearea unui meniu de setări în pagina principală pentru a nu fi nevoie ca utilizatorul să schimbe pagini pentru câteva setări simple, plus câteva animații simple și discrete pentru a oferi un sentiment de fluiditate utilizatorilor.

În perioada de dezvoltare a aplicației, am învățat biblioteca *React.js* pentru a oferi reactivitate aplicației, *MongoDb* pentru stocarea datelor într-o bază de date *NoSQL*, *Git* pentru gestionarea versiunilor aplicațiilor și pentru a implementa caracteristici suplimentare fără a risca stabilitatea aplicației principale, arhitectură pentru a gestiona datele trimise de utilizator din *front-end* în *back-end* prin intermediul unui *API* prin care validez datele trimise de utilizatori.

Potențialele îmbunătățiri care pot fi aduse aplicației sunt implementarea *token*-urilor pentru a spori securitatea autentificării, caracteristici suplimentare cum ar fi crearea unui grup pentru a comunica împreună cu mai multe persoane în același timp, crearea opțiunii de *Friends* pentru a avea afișate doar contactele pe care le avem la prieteni printr-o cerere de prietenie, trimiterea aceluiași mesaj la mai multe persoane în același timp și multe altele.

În concluzie, aplicația *Chitchat* este un proiect ce reprezintă un sumar al cunoștințelor asimilate pe parcursul anilor de facultate, cum ar fi: implementarea unei arhitecturi de tip *MVC(Model-View-Controller),* crearea unui *API*, crearea unui client utilizând o arhitectură orientată spre evenimente, lucrul cu baze de date și utilizarea *WebSocket-*urilor pentru a transmite informații între client și server în timp real.

# Bibliografie

[1] DevDocs. (2013). *DevDocs*. Preluat pe August 2022, de pe <https://devdocs.io/react/>

[2] Grider, S. (2023). *Modern React with Redux*. Preluat pe Ianuarie 2023, de pe <https://www.udemy.com/course/react-redux/>

[3] Matt Zabriskie, e. a. (2014). *Axios*. Preluat pe Martie 2023, de pe <https://axios-http.com/docs/intro>

[4] Meta. (2023). *React*. Preluat pe Martie 2023, de pe <https://react.dev/>

[5] MongoDB. (2023). *MongoDB*. Preluat pe Ianuarie 2023, de pe [https://www.mongodb.com/docs/](%20https:/www.mongodb.com/docs/)

[6] MongoDB. (2023). *NoSQL vs. SQL Databases*. Preluat pe Februarie 2023, de pe <https://www.mongodb.com/nosql-explained/nosql-vs-sql>

[7] OpenJS Foundation. (2017). *Express.js*. Preluat pe Noiembrie 2022, de pe <https://expressjs.com/en/5x/api.html>

[8] Schmedtmann, J. (2023). *Advanced CSS and Sass: Flexbox, Grid, Animations and More!* Preluat pe Ianuarie 2023, de pe <https://www.udemy.com/course/advanced-css-and-sass/>

[9] Schmedtmann, J. (2023). *Node.js, Express, MongoDB & More: The Complete Bootcamp 2023*. Preluat pe Ianuarie 2023, de pe <https://www.udemy.com/course/nodejs-express-mongodb-bootcamp/>

[10] Schmedtmann, J. (2023). *The Complete JavaScript Course 2023: From Zero to Expert!* Preluat pe Martie 2023, de pe <https://www.udemy.com/course/the-complete-javascript-course/>

[11] Socket.IO. (2023). *Socket.IO*. Preluat pe Aprilie 2023, de pe [https://socket.io/docs/v4/](%20https:/socket.io/docs/v4/)

[12] Wikipedia. (2023). *Facebook*. Preluat pe Iunie 2023, de pe <https://en.wikipedia.org/wiki/Facebook>

[13] Wikipedia. (2023). *Signal (software)*. Preluat pe Iunie 2023, de pe <https://en.wikipedia.org/wiki/Signal_(software)>

[14] Wikipedia. (2023). *WhatsApp*. Preluat pe Iunie 2023, de pe <https://en.wikipedia.org/wiki/WhatsApp>