

UNIVERSITATEA BABEȘ-BOLYAI CLUJ-NAPOCA
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ
SPECIALIZAREA INFORMATICĂ

LUCRARE DE LICENȚĂ

Descentralizarea rețelelor de socializare folosind metode de comunicare peer-to-peer la nivelul browserului web

Conducător științific

Drd. Limboi Sergiu

Prof. Dr. Dioșan Laura

Absolvent

Mărginean Tiberius

2021

Abstract

Online social networking services have seen a major increase in popularity over the last decade, with statistics showing people spending on average an increasing amount of time daily using them. Nowadays they have become one of the main methods for sharing personal ideas and information, and are being regarded as a free public communication channel. However, the centralized architecture behind all major social networks has led to criticism and questions about the power of their owners, who might purposely alter or block content they do not agree with on their platforms.

The solution proposed by this document is a decentralized social network named EccentricPeer, which relies on peer-to-peer communication between users to ensure a free flow of content, and uses cryptographical methods to ensure its consistency. The entire system is accessed using a simple web application in the web browser.

The first part of this document creates an overview of the modern technologies that allow the conception of such a peer-to-peer network where browsers may participate as nodes. Then, the focus is shifted to the actual architecture of the network, the involved actors and the necessary communications between them, as well as the means of securing the shared data and user's information. Finally, the document presents the features and operation of the proposed solution.

Cuprins

1. Introducere	5
1.1. Motivație. Context	5
1.2. Scop și elemente de originalitate	5
2. Arhitecturi distribuite utilizând browserul web – aspecte teoretice	7
2.1. Cercetări în domeniu.....	7
2.2. Arhitectura aplicațiilor web	8
2.3. Particularități ale rețelelor de socializare online	11
2.4. Comunicarea prin WebSockets	13
2.5. Tehnologia WebRTC	14
2.6. Rețele peer-to-peer în browserul web	18
2.7. Utilitatea criptografiei într-o rețea peer-to-peer.....	21
3. Proiectarea și implementarea unei rețele de socializare peer-to-peer	25
3.1. Structura aplicației	25
3.1.1. Modulul client.....	25
3.1.2. Modulul server	28
3.1.3. Alte aspecte arhitecturale	29
3.2. Date gestionate de aplicație	30
3.2.1. Profiluri de utilizator.....	30
3.2.2. Postări	32
3.2.3. Elemente de criptografie	33
3.2.4. Alte date salvate local	34
3.3. Rețeaua peer-to-peer	35
3.3.1. Topologia rețelei. Canale de comunicare.....	35
3.3.2. Protocolul de comunicare în rețea.....	37

3.3.3. Înregistrarea și autentificarea în rețea	39
3.3.4. Stabilirea conexiunilor	40
3.3.5. Tratarea solicitărilor între modulele client.....	42
3.3.6. Tratarea solicitărilor de către modulele server.....	43
3.4. Persistența datelor	44
3.4.1. Date salvate în modulele client	44
3.4.2. Date salvate în modulele server	45
3.5. Beneficii și limitări ale arhitecturii utilizate	46
4. EccentricPeer – rețea de socializare descentralizată	48
4.1. Funcționalități și diagrama cazurilor de utilizare.....	48
4.2. Model conceptual.....	50
4.3. Tehnologii folosite	51
4.4. Mockups.....	52
5. Concluzii	56
6. Bibliografie	57

1. Introducere

1.1. Motivație. Context

Rețelele de socializare prin intermediul internetului se bucură de o continuă creștere a popularității, ajungând astăzi să reprezinte o bună parte dintre cele mai accesate site-uri web. Acest număr mare de persoane a dus la o transformare a acestor platforme online, ducând la depășirea scopului inițial de a comunica și cunoaște alți oameni și creând un mediu care facilitează transmiterea ușoară a informațiilor și comunicarea pe scară largă.

Din punctul de vedere al arhitecturii acestor rețele, creșterea numărului de utilizatori a necesitat o continuă extindere a capacității de procesare a datelor. Astfel companiile deținătoare au fost nevoite să construiască sisteme foarte mari de servere care să poată face față noilor cerințe. Acest model de dezvoltare centralizată ridică semne de întrebare în legătură cu asigurarea libertății de exprimare și cu siguranța datelor utilizatorilor mediului online format. Practic a fost creat un nou spațiu public aflat sub guvernarea companiilor private din spatele rețelilor de socializare, fapt care a deschis discuții referitoare chiar la posibilitatea practicării cenzurii [28].

În tot acest timp, modalitățile de transmitere a datelor prin internet au progresat. Tehnologii de comunicare peer-to-peer care au văzut un progres relativ lent, fiind văzute ca o alternativă bună doar pentru transmiterea de fișiere mari sau pentru diverse utilizări în rețele mai mici, au fost adoptate direct în browsere moderne pentru a facilita transmiterea mai rapidă a informațiilor și relaxarea presiunii asupra serverelor. Totuși, în cadrul rețelilor de socializare progresul acestor facilități s-a rezumat doar la aplicații minore.

1.2. Scop și elemente de originalitate

Această lucrare propune realizarea unei rețele de socializare online urmând principiul descentralizării, în cadrul căreia transmiterea datelor către utilizatori să fie realizată prin conexiuni peer-to-peer. Prin structura acestei rețele se urmărește prevenirea existenței oricărei entități cu puterea și potențialul de blocare a unui conținut distribuit de un utilizator, înainte ca acesta să ajungă la persoanele pentru care se adresează. Rolul de filtrare al informațiilor

primite va reveni exclusiv fiecărui utilizator în parte, acesta având libertatea de a decide să blocheze sau nu conținutul publicat de oricare altă persoană.

De asemenea, se dorește folosirea tehnologiilor moderne de comunicare peer-to-peer pentru a crea facilități de utilizare similare cu cele oferite de rețelele de socializare actuale. Utilizatorii finali vor beneficia de o interfață familiară care nu va necesita o schimbare a modului de interacțiune datorată modului de implementare și nici nu va suferi diminuări ale performanței cauzate de acesta. Totodată, se vor putea oferi funcționalități asemănătoare cu cele oferite de majoritatea alternativelor actuale, printre care: posibilitatea de a crea un cont și definirea în baza acestuia a unui profil public, postarea în cadrul lui a oricărui tip de conținut dorit, accesarea profilurilor corespunzătoare altor persoane, posibilitatea de a urmări sau bloca unele conturi.

În locul marilor sisteme centralizate de servere, se va profita de evoluția semnificativă a dispozitivelor folosite de utilizatori pentru accesarea internetului și se vor salva direct pe acestea datele necesare funcționării rețelei. Pentru asigurarea consistenței și disponibilității datelor, se vor folosi servere puse la dispoziție voluntar de către orice utilizator sau entitate privată doritoare. Rolul principal al acestor servere va fi de a stoca și de a oferi la nevoie datele publicate de utilizatori. Pe de altă parte, dispozitivele persoanelor care accesează rețeaua vor acorda prioritate salvării datelor profilurilor urmărite sau accesate des.

Capitolul 2 descrie aspectele teoretice care fac posibilă funcționarea unei rețele de socializare peer-to-peer accesată prin browserul web, studiind structura acestui tip de arhitectură, canalele de comunicare a informației și securitatea unui astfel de sistem. În capitolul 3 se prezintă modul de funcționare a implementării unei rețele de socializare proiectată urmând o arhitectură distribuită, evidențiind beneficiile și limitările acesteia. Capitolul 4 prezintă EccentricPeer, o rețea de socializare descentralizată, bazată pe comunicarea peer-to-peer între utilizatori, accesată prin intermediul unei aplicații web. Capitolul 5 conține concluziile identificate prin această lucrare.

2. Arhitecturi distribuite utilizând browserul web – aspecte teoretice

2.1. Cercetări în domeniu

Încă din anul 2004 Gary Sivek et al. au introdus conceptul de WebTorrent [21], o tehnologie care permite decongestionarea serverelor intens utilizate prin partajarea resurselor direct între clienți. Aceștia au propus folosirea protocolului BitTorrent pentru a forma o rețea peer-to-peer prin intermediul căreia să fie transmis conținutul paginilor web, lăsând în sarcina serverului web doar transmiterea unui mic fișier torrent și comunicarea pentru tracking. Au obținut astfel o scădere semnificativă a timpului de răspuns pentru accesarea paginilor disponibile prin intermediul acestei tehnologii și au creat perspectiva rețelelor peer-to-peer la nivel de browser web.

Enrico Franchi et al. au propus în lucrarea lor [15] rețeaua de socializare Blogracy care folosește o tabelă de dispersie distribuită pentru a păstra referințe către activitățile publicate de utilizatori, acestea fiind mai apoi distribuite printr-un client BitTorrent extern browserului web. Rețeaua peer-to-peer obținută de aceștia a reușit să obțină un timp foarte bun de acces al paginilor, cu condiția ca în rețea să existe suficienți urmăritori care să joace rol de seed-eri.

Ames Bielenberg et al. au studiat rețeaua de socializare Diaspora [2], o rețea distribuită care se remarcă prin faptul că fiecare utilizator trebuie să aleagă un server numit „pod”, care poate fi găzduit de un terț sau chiar de utilizator, în cadrul căruia urmează să fie salvate toate datele personale distribuite de acesta. Studiul a descoperit o popularitate sporită a rețelei de socializare datorată descentralizării și garanției libertății de exprimare. Totuși, s-a observat și tendința utilizatorilor de a migra către servere cât mai populare, evitând găzduirea propriului pod, ducând astfel la o diminuare a beneficiilor descentralizării prin mutarea responsabilității de a gestiona datele de la dezvoltatorii rețelei la entitățile terțe care găzduiesc aceste pod-uri.

2.2. Arhitectura aplicațiilor web

Termenul „World Wide Web” [43], folosit adesea și în forma scurtă „web”, se referă la un sistem informațional care identifică prin adrese unice (Uniform Resource Locators, prescurtat URL) documente, fișiere și alte tipuri de resurse accesibile prin internet [29]. Principalii actori în această rețea sunt serverele web, sisteme care găzduiesc și facilitează accesul la resurse web, și clienții, în general browsere web care accesează aceste resurse și le prezintă unui utilizator.

Resursele web, pe lângă date statice, pot avea un conținut dinamic, actualizat de serverul web în funcție de anumiți factori, și pot conține fragmente de cod (scripturi) care pot fi rulate de către clienți, având astfel posibilitatea de a executa diverse operații similar cu o aplicație software locală. O astfel de aplicație, accesibilă prin World Wide Web de către clienți web, este numită aplicație web [29].

Introducerea documentului ECMA-262 [18], aprobat drept standard internațional [24], care prevede caracteristicile scripturilor ce pot fi rulate la nivel de browser, a dus la posibilitatea de preluare de către client a execuției unor părți din logica aplicațiilor. Astfel, există diverse tipuri de aplicații web, de la aplicații care rulează integral pe serverele web, acestea fiind responsabile de pregătirea conținutului actualizat dinamic care e trimis spre client, la aplicații care lasă unele părți de procesare în seama clientului și până la aplicații care folosesc serverul web doar pentru a fi distribuite către clienți urmând ca toată partea efectivă de logică a aplicației să fie rulată de browserul utilizatorului.

O astfel de aplicație, care respectă standardele web putând rula în browser și care odată încărcată nu mai are nevoie să comunice cu serverul web, poartă numele de Progressive Web Application, prescurtat PWA. Acest concept a fost introdus în anul 2015 de Alex Russell [6], care a evidențiat principalele avantaje ale acestei tehnologii, cum ar fi capacitatea acestor aplicații de a rula asemănător cu aplicațiile instalate local, putând accesa unele funcții ale sistemului de operare, siguranța și transparența, independența de conexiunea la internet după ce au fost încărcate, ușurința distribuirii folosind adrese URL, aplicația prezentându-se ca un site web, dar și adaptivitatea la mediul în care rulează, funcționând pe orice dispozitiv care permite folosirea unui browser web modern. Toate aceste beneficii au dus astăzi la o popularitate sporită a dezvoltării de aplicații folosind tehnologii și standarde web.

Browserele moderne sunt într-o continuă adaptare la noile standarde web și includ mai multe facilități care permit dezvoltarea de PWA. Cea mai importantă dintre acestea este mediul de rulare pentru JavaScript, cel mai popular limbaj de programare bazat standardul

descriș de ECMA-262 [18]. Astfel, codul JavaScript poate fi integrat cu ușurință în orice pagină web și rulat de către fiecare client. Profitând de acest mecanism, un PWA poate fi transmis ca orice resursă web în format text de pe un server web, printr-o solicitare HTTP. După acest pas, întreaga aplicație este încărcată și rulează în browser, având atât posibilitatea de a accesa resurse din cadrul sistemului clientului cât și de a deschide noi conexiuni și de a accesa resurse din internet.

Spre deosebire de conexiunile HTTP clasice stabilite de browser cu serverele web, conexiunile stabilite de aplicațiile web pot respecta orice protocol de comunicare, putând accesa chiar și resurse și servicii care nu sunt accesibile prin web. Deci, dacă într-o aplicație clasică comunicarea cu un server terț ar fi fost realizată în fundal de serverul web, în cadrul unui PWA apelul poate fi realizat direct, scăzând astfel numărul de solicitări procesate de serverul web și putând chiar oferi clientului un timp mai bun de răspuns. Totodată, aceste noi posibilități de comunicare au dus la introducerea în standardele web a posibilității de stabilire a conexiunilor directe între clienți prin tehnologia WebRTC [48].

De regulă, în cadrul unei aplicații web se dorește accesarea unor date, efectuarea unor operații cu acestea și gestionarea lor. O caracteristică a PWA care le permite să exceleze în acest domeniu este arhitectura de tip Application Shell [5], care facilitează încărcarea în fundal a entităților, doar în măsura în care sunt necesare. Practic, aceste aplicații pot fi împărțite în două componente: „carcasa” aplicației, constând în codul HTML, CSS și JavaScript care generează interfața aplicației și care trebuie să fie încărcat rapid în browser, și conținutul, obținut în mod dinamic urmând principiul lazy-loading. Astfel se obțin timpi de răspuns foarte buni, aplicația încărcându-se chiar dacă datele încă sunt colectate în fundal, și se evită aglomerarea inutilă a rețelei cu date care nu sunt necesare utilizatorului. Mai mult, ”carcasa” aplicației poate fi salvată în cache în cadrul browserului, reducând astfel la minimum solicitarea serverului web după prima accesare și permițând funcționarea fără conexiune la internet [39].

Totuși, pentru a rula aplicația offline nu este suficientă doar interfața, așa că browserele web moderne au introdus facilități și pentru salvarea pe dispozitiv a conținutului aplicației, cum ar fi API-urile WebStorage [47] și IndexedDB [45], gândite pentru salvarea cu ușurință a unor seturi mici de date, respectiv pentru salvarea datelor complexe într-o bază de date care rulează la nivelul clientului. Folosirea acestor facilități permite atât afișarea conținutului în cadrul aplicației web chiar dacă nu se poate realiza conexiunea la internet, cât și decongestionarea suplimentară a rețelei, nefiind nevoie ca entitățile să fie transmise din nou clienților dacă nu au suferit între timp modificări de la ultima accesare.

Profitând de beneficiile arhitecturii Application Shell, au fost dezvoltate aplicații web cu o singură pagină (Single Page Application). Conținutul acestei pagini, incluzând elementele de interfață, este actualizat în funcție de acțiunile utilizatorului lăsând impresia navigării pe pagini diferite. Beneficiile folosirii acestei arhitecturi includ posibilitatea unei mai bune gestionări a solicitărilor utilizatorului, schimbarea paginii făcându-se tot în cadrul aplicației fără a depinde de modul clasic de încărcare a paginilor web în browser, cât și de modularitatea aplicației rezultate, cuprinzând mai multe pagini în cadrul aceluiași cod sursă.

Pentru a facilita dezvoltarea acestui tip de aplicații au apărut framework-uri și librării precum Angular [7]. De asemenea, dezvoltarea la scară largă a dus la apariția unor aplicații foarte complexe în cadrul cărora limbajul de programare JavaScript [18], conceput inițial pentru scripturi relativ mici inserate în pagini web clasice, s-a dovedit dificil de utilizat. Lipsa tipurilor de date asociate variabilelor a fost identificată drept un mare impediment în dezvoltarea și menținerea aplicațiilor complexe. Acesta a fost motivul care a dus la apariția limbajului de programare TypeScript [19], o extensie a JavaScript menită să rezolve o mare parte din problemele identificate: variabilelor li se poate asocia un tip în momentul declarării, s-a introdus un sistem de module, există posibilitatea definirii de interfețe, precum și alte beneficii. O caracteristică importantă este aceea că în urma compilării codului TypeScript rezultă codul echivalent JavaScript, care poate fi rulat de browserul web. Astfel, programatorii au la dispoziție niște unelte mai clare de a defini comportamentul aplicațiilor dezvoltate, cu posibilitatea unei întrețineri mult mai ușoare, iar rezultatul final este cod simplu JavaScript.

Cuprinzând toate aceste tehnologii și evoluții la nivelul browserelor, aplicațiile web moderne au toate uneltele pentru a implementa funcționalități similare cu aplicațiile locale clasice [39], putând gestiona și stoca date, stabili conexiuni prin internet și chiar rula offline. Framework-urile apărute permit dezvoltarea cu ușurință a aplicațiilor complexe, lăsând flexibilitate în alegerea arhitecturii necesare și oferind beneficiile date de PWA.

2.3. Particularități ale rețelelor de socializare online

O rețea de socializare oarecare poate fi definită ca o colecție de noduri conectate prin una sau mai multe relații, de obicei aceste noduri fiind persoane sau organizații [3]. În cadrul mai restrâns al comunicațiilor digitale prin internet, Boyd și Ellison definesc [13] o rețea de socializare online drept o platformă web care permite persoanelor crearea unui profil public sau semi-public, sintetizarea unei liste de utilizatori cu care împărtășesc o conexiune și vizualizarea și parcurgerea acestei liste precum și a conexiunilor create de alți utilizatori. Obar și Wildman introduc în cadrul definiției [25] caracterul conținutului distribuit prin rețea, acesta fiind generat de utilizatori, încadrând rețelele de socializare online în conceptul Web 2.0 [20]. Așadar, se evidențiază accentul pe transmiterea de idei și creații ale fiecărei persoane care participă în rețea, nu doar pe distribuirea de informații concrete.

Conform unor studii recente [32], peste 50% din populația țărilor dezvoltate accesează zilnic rețele de socializare online, în cadrul acestora petrecând majoritatea timpului de navigare pe internet. De asemenea, utilizatorii petrec în medie peste 2 ore zilnic [36] folosind aceste platforme web, iar numărul de persoane care le accesează regulat este într-o continuă creștere [36]. Practic, s-a creat un mediu virtual în permanență activ, existând un număr foarte mare de persoane care generează constant nou conținut pe care îl distribuie altor utilizatori cu care au realizat o conexiune.

Ca structură a acestui conținut, majoritatea platformelor au adoptat, sub o formă sau alta, conceptul de postare. Acestea pot fi scrise și publicate în orice moment de către utilizatori, platforma rețelei de socializare fiind responsabilă cu distribuirea lor către alte persoane, precum și cu salvarea acestora pentru a fi citite, modificate sau șterse ulterior.

Profilul public pe care fiecare persoană îl definește este o pagină dedicată care conține date de bază pe care utilizatorul vrea să le împărtășească cu alți membri ai rețelei [13], acestea variind în funcție de platformă de la un simplu nume de utilizator până la informații cu mai mult caracter personal, precum numele, o fotografie, date de contact sau detalii despre educație. Principalele funcții pe care trebuie să le îndeplinească această pagină sunt de a prezenta imaginea fiecărui utilizator în cadrul rețelei, de a păstra toate postările făcute de acesta și de a afișa conexiunile cu alți utilizatori. De regulă, fiecare profil are și un identificator unic atribuit automat la creare, care ia forma unui număr sau a unei secvențe de caractere fără un înțeles pentru persoana înregistrată, pe care rețeaua îl folosește în fundal pentru a gestiona datele aferente fiecărui cont de utilizator.

Platformele de socializare oferă utilizatorilor și posibilitatea de a-și reînnoi sau schimba imaginea publică creată. Oamenii sunt într-un continuu proces de acumulare de informații, proces în urma căruia anumite idei și concepții personale suferă modificări. Rețelele de socializare trebuie să permită asemenea modificări, oferind oricând posibilitatea utilizatorilor de a actualiza informațiile publice din profil sau de a elimina postări făcute în trecut cu care nu mai sunt de acord.

S-au observat în studiile lui Lampe et al. [11] și Joinson [4] două tipuri de comportamente ale persoanelor care navighează în cadrul unei rețele de socializare, ambele facilitate de modul de funcționare al paginilor de profil ale utilizatorilor: căutarea persoanelor și descoperirea acestora. Căutarea persoanelor se referă la identificarea oamenilor cunoscuți în viața de zi cu zi pe baza datelor personale completate în profil, interacțiunea cu aceștia și urmărirea conținutului publicat de ei. Pe de altă parte, descoperirea persoanelor se referă la utilizarea platformei pentru a intra în contact cu alți oameni necunoscuți, lucru facilitat de expunerea publică în profil a conexiunilor. Se creează astfel un graf [13] în care muchiile sunt materializate de aceste conexiuni, iar parcurgerea liberă a acestora duce la întâlnirea unor utilizatori noi ai rețelei. Ca și practică uzuală, persoanele cu care se realizează o astfel de conexiune sunt afișate drept „prieteni” sau „urmăritori”.

Ambele cazuri de utilizare a acestor platforme prezentate [4][11] ajută la înțelegerea tipului de trafic de internet în cadrul rețelelor de socializare. S-a concluzionat tendința utilizatorilor de a revizita des anumite profiluri cu scopul de a se păstra la curent cu noul conținut publicat de deținătorii acestora. De asemenea, structura de graf [13] a acestor rețele face ca majoritatea persoanelor descoperite prin intermediul conexiunilor să fie la un număr mic de muchii distanță față de restul utilizatorilor cu care s-a stabilit anterior o legătură.

Un alt aspect al traficului în cadrul rețelelor de socializare online este dat de valorile extreme, momentele în care un număr foarte mare de utilizatori accesează platforma. Ratkiewicz et al. [26] au studiat asemenea creșteri neașteptate ale valorilor de trafic, remarcând că anumite evenimente externe produc „explozii” ale numărului de accesări ale conținutului online. Ca și caracteristică a acestui trafic excepțional se evidențiază direcția lui preponderentă spre acele pagini web care cuprind informații din sfera evenimentului extern care a cauzat creșterea.

Așadar, o rețea de socializare online se bazează pe conținut generat liber de un număr cât mai mare de utilizatori, care folosesc conexiunile puse de dispoziție de platformă pentru distribuirea acestor idei spre alte persoane. Esența unei rețele este capacitatea de a accesa idei

publicate de diverși oameni [25], deci rolul principal al sistemului din spatele unei platforme web de socializare este de a garanta consistența, durabilitatea și accesibilitatea acestora.

2.4. Comunicarea prin WebSockets

Sistemul informațional World Wide Web face posibilă accesarea unor resurse pe baza adreselor URL [43]. Modul clasic de accesare a acestora se bazează pe protocolul HTTP [34]. Acesta se bazează pe mecanismul cerere-răspuns: când se dorește accesarea unei resurse, clientul trimite o cerere către un server, acesta procesează cererea și trimite înapoi un răspuns. Acest mecanism s-a dovedit insuficient pentru nevoile de comunicare apărute odată cu aplicațiile web moderne, în cadrul cărora se dorește adesea un mod optim de a trimite cereri repetate către același server, dar și transmiterea de informații către clienți fără a aștepta o cerere din partea acestora.

În sprijinul acestor noi cerințe a fost dezvoltată tehnologia WebSocket, introdusă în protocolul HTML5 [44] și standardizată în anul 2011 [23]. Prin intermediul WebSocket se facilitează comunicarea bidirecțională între client și server, folosind o singură conexiune TCP menținută pe toată durata comunicării între cei doi actori.

Stabilirea unei conexiuni care urmează protocolul WebSocket se folosește de o cerere HTTP care cuprinde o ofertă de upgrade [23]. Astfel, pentru a putea comunica prin această metodă este nevoie de un server HTTP care rulează pe același port, de regulă 80 sau 443, rolul acestuia fiind de a interpreta cererea inițială și de a trimite un răspuns favorabil clientului. După acest pas, folosirea protocolului HTTP pentru comunicare nu mai este necesară. Pentru trimiterea și primirea mesajelor prin această conexiune, WebSocket pune la dispoziție un API [46] bazat pe evenimente, oferind astfel o cale simplă de procesare a interacțiunilor prin acest canal de comunicație. La fel ca trimiterea de mesaje, și închiderea conexiunii poate fi semnalată de oricare din cei doi participanți.

O caracteristică importantă a protocolului este structura mult mai simplă a mesajelor. Practic se renunță la overhead-ul implicat de transmiterea antetelor HTML atașate fiecărei cereri și la timpul necesar restabilirii conexiunii pentru fiecare apel. În schimb, mesajele sunt cuprinse într-o „ramă” [23] de dimensiune cât mai mică, în funcție de lungimea mesajului. În acest fel în unele aplicații se reușește transmiterea de până la 1000 de ori mai puține date nefolositoare și o latență de până la 3 ori mai mică față de utilizarea tehnologiilor de HTTP

polling și long-polling care îndeplinesc aceleași funcții [33]. De asemenea, Skvorc et al. [14] au obținut cu WebSocket performanțe comparabile cu folosirea directă a protocolului TCP și au concluzionat faptul că, în cazul conexiunilor care rămân deschise mult timp, overhead-ul creat de cele două mesaje HTTP inițiale asupra volumului de date transmis este nesemnificativ. În aceste condiții, tehnologia WebSocket își arată potențialul atât pentru comunicarea client-server cât și pentru interacțiuni server-server simple, în cadrul unor rețele de dimensiuni mari, în aplicații unde flexibilitatea și simplitatea oferite de acest protocol se dovedesc foarte benefice.

Din punct de vedere al securității, WebSocket implementează mai multe măsuri [23] care să prevină atacurile. Una dintre acestea este posibilitatea de a stabili conexiuni în cadrul cărora mesajele sunt criptate prin TLS, numite WebSocket Secure (WSS), prevenind astfel posibilitatea ca datele transmise să fie interceptate și preluate de un intermediar. Alte măsuri includ mecanismul de masking, care previne ocuparea eventualelor cache-uri intermediare cu date eronate, precum și header-ul „Origin”, primit în cererea de stabilire a conexiunii, pe care serverul îl poate verifica pentru a preveni conexiuni inițiate de scripturi malițioase. În esență, canalul de comunicare stabilit de această tehnologie este considerat sigur, probabilitatea mai mare a unui atac afectând aplicațiile de la capetele acestuia.

Mulțumită faptului că WebSocket a fost adoptat ca standard internațional de către IETF [23], toate browserele web populare au implementat API-ul [46] care facilitează folosirea acestui protocol. În aceste condiții, această tehnologie a reușit să devină una din cele mai comune metode de comunicare pentru aplicații moderne precum Progressive Web Applications [6] în cadrul cărora se dorește transmiterea bidirecțională de date în timp real, cu o latență cât mai scăzută.

2.5. Tehnologia WebRTC

Tehnologia Web Real-Time Communications (WebRTC) este compusă dintr-un API JavaScript și o colecție de protocole de comunicare care permit transmiterea de date peer-to-peer între browsere web [48]. Astfel, această tehnologie face posibil transferul conținutului multimedia și al resurselor web direct între dispozitivele utilizatorilor, fără a necesita un server web intermediar.

Ideea unei astfel de tehnologii își are rădăcinile la începutul anilor 2000. Odată cu sporirea popularității internetului, serverele web au avut nevoie de un continuu efort de creștere a capacității de procesare și de transfer al resurselor web. Protocoale de transfer peer-to-peer precum BitTorrent și-au arătat performanța în transferul de date generice, așa că s-a dorit aplicarea aceluiași principiu și în cazul resurselor web, urmărind diminuarea traficului de date suportat de serverele web [21]. Urmând o creștere semnificativă a popularității serviciilor de mesagerie directă prin internet, compania Google a făcut publică în 2011 prima versiune a WebRTC [22], menită să asigure transferul rapid al conținutului audio și video între doi participanți ai unui apel inițiat printr-un astfel de serviciu direct din browserul web. De atunci, tehnologia a fost adoptată într-o gamă largă de aplicații și a suferit multe actualizări, urmând ca în ianuarie 2021 să fie adoptată ca recomandare oficială a W3C și standard IETF [49]. Una din cele mai notabile actualizări a fost introducerea unui canal de comunicare pentru transferul datelor generice, nu doar al conținutului audio sau video, permițând astfel transferul oricăror resurse între browserele web.

API-ul WebRTC conține trei componente principale: `MediaStream`, `PeerConnection` și `DataChannel` [37]. `MediaStream` este o reprezentare abstractă a unui flux de date audio și/sau video, care poate fi preluat de la dispozitive fizice precum microfoane sau camere web, iar asupra lui se pot executa direct operații cum ar fi înregistrarea lui sau transmiterea către un alt peer. `PeerConnection` reprezintă o asociere cu un alt browser care, de obicei, rulează aceeași aplicație și se poate realiza cu acesta transfer direct de date. `DataChannel` este o parte a API-ului care permite comunicarea peer-to-peer de date generice, într-o manieră bidirecțională. În ceea ce privește facilitățile oferite programatorului, `DataChannel` poate fi folosit similar cu un `WebSocket` [46], privit ca și cum ar fi deschis direct cu un alt client.

Deși tehnologia WebRTC permite comunicarea directă între browsere, pentru stabilirea inițială a acestui canal este necesar un serviciu intermediar care să permită negocierea conexiunii între cele două browsere. Acest protocol de stabilire a conexiunii, numit „signaling”, nu este standardizat, tehnologia lăsând la latitudinea dezvoltatorului stabilirea metodei de implementare [37]. Deoarece protocolul HTTP [34] este deja cel mai răspândit și folosit la nivelul browserului web, se poate folosi un server intermediar pe care ambii utilizatori îl contactează inițial prin solicitări HTTP. Totuși, flexibilitatea oferită de WebRTC dă posibilitatea stabilirii acestor canale de comunicare și folosind conexiuni `WebSocket` cu un server, sau chiar folosind `DataChannel`-ul unei alte conexiuni WebRTC stabilită anterior.

Mecanismul de signaling întâmpină dificultatea dată de existența de regulă între cele două browsere a mai multor rețele cu spații de adresă ascunse prin NAT. Pentru a depăși această problemă se folosește mecanismul ICE (Interactive Connectivity Establishment), care simplifică mult complexitatea sistemului de adresare din internet [8]. Acesta folosește servere STUN (Session Traversal Utilities for NAT) care determină adresa IP publică a browserului, aceasta trebuind transmisă celuilalt peer. La nevoie, în cazul în care conexiunea directă nu poate fi stabilită din cauza setărilor de rețea, se folosesc servere TURN (Traversal Using Relays around NAT) care au rolul de a redirecționa traficul dintre cele două browsere. Astfel, comunicarea se realizează prin acest server intermediar care ar trebui să fie acceptat de orice configurație de rețea.

Procesul de stabilire a conexiunii are mai multe etape [37]. În primul rând, browserul care dorește să inițieze o conexiune WebRTC creează o „ofertă”, care conține datele referitoare la această conexiune, cum ar fi lista de servere pentru ICE, tipurile de canale care se doresc deschise (audio, video și/sau data), dar și restul informațiilor necesare pentru stabilirea canalului de comunicare. Această ofertă este scrisă urmând formatul Session Description Protocol (SDP) [1] și este trimisă prin serviciul intermediar de signaling către celălalt browser, care alege dacă o acceptă sau nu. În caz afirmativ, acesta construiește un răspuns în care include datele sale și pe care îl transmite tot în format SDP, prin serviciul de signaling, către browserul inițiator. După primirea răspunsului, cele două browsere intră într-o etapă de negociere în care folosesc din nou serviciul de signaling pentru a transmite între ele informații despre canalele disponibile pentru comunicarea prin rețea, sub numele de „candidați ICE”, până la identificarea unui canal care să asigure conexiunea. Odată cu găsirea acestui canal acceptat de ambele părți, conexiunea WebRTC este deschisă și se poate realiza comunicarea directă între cele două browsere. Figura 1 ilustrează desfășurarea acestui întreg proces.

Din punct de vedere al securității, imediat după stabilirea oricărei conexiuni mai intervine un pas de negociere al DTLS (Datagram Transport Layer Security) [17]. Conținutul transmis prin WebRTC este obligatoriu criptat în etapa de transport între clienți. Pentru fiecare canal deschis (audio, video sau data) se schimbă între cele două browsere informații despre protocolul de criptare a datelor, iar la final se exportă cheile care vor fi folosite ulterior pentru securizarea mesajelor. Astfel, niciun mesaj transmis prin conexiunea stabilită nu va putea fi interceptat și citit de un intermediar. Procesul de securizare a canalului de comunicare este unul automat, inclus în standardul acceptat de W3C și IETF [49], și nu necesită nicio intervenție specială din partea programatorului. Deci, tehnologia WebRTC

poate fi considerată o metodă sigură de comunicare între browsere web, în cazul oricărui tip de canal utilizat.

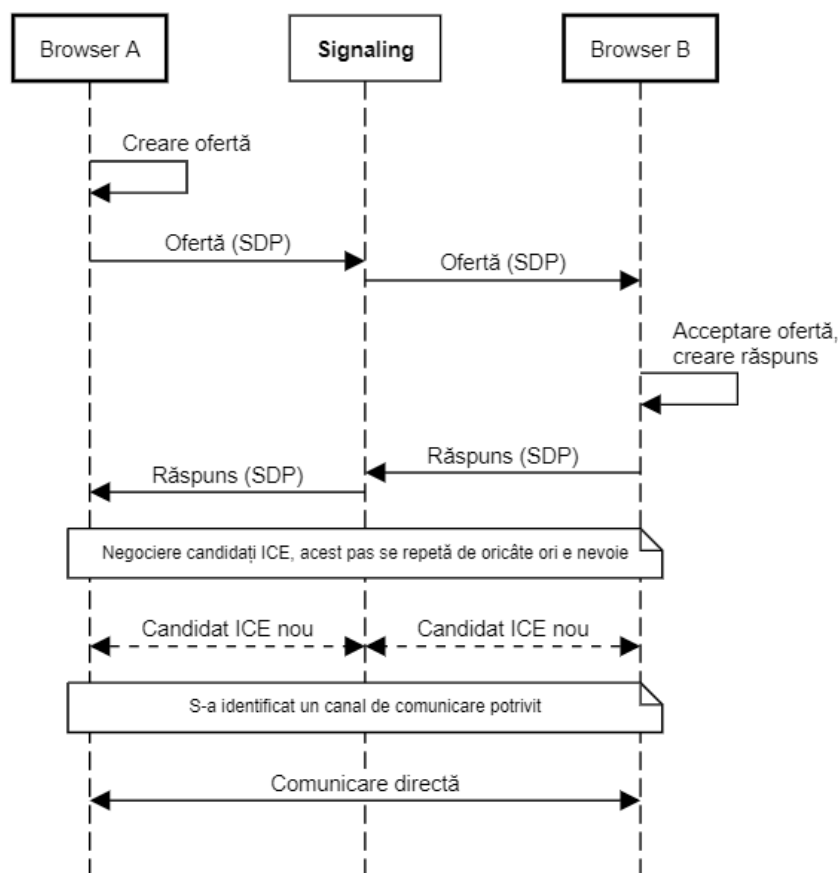


Fig. 1: Mesajele transmise între două browsere prin intermediul unui serviciu intermediar, numit generic „Signaling”, pentru stabilirea unei conexiuni directe prin WebRTC

Așadar, WebRTC este o tehnologie nouă în cadrul browserelor web care oferă modalități noi de transmitere a informațiilor prin World Wide Web. Mulțumită numărului mare de funcționalități oferite și a siguranței asigurate, WebRTC a fost deja integrat în multe aplicații pentru a prelua transmiterea mesajelor și realizarea apelurilor directe între clienți. Totodată, posibilitatea deschiderii unui canal pentru date generice atribuie tehnologiei un potențial ridicat și pentru alte domenii de utilizare, inclusiv pentru preluarea unor responsabilități care ar fi revenit serverelor web.

Totuși, un aspect de luat în considerare în momentul deciziei de a folosi tehnologia WebRTC este dat de overhead-ul produs de procesul de stabilire a conexiunii. Chiar dacă administratorul aplicației web trebuie să pună la dispoziție doar serviciul de signaling, asupra

căruia impactul realizării conexiunilor este unul mic, procesul de negociere este suportat de clienți și uneori poate fi relativ costisitor din punct de vedere al timpului de identificare a canalului de comunicare potrivit. Mai mult, în unele cazuri configurația rețelei ar putea necesita prezența și utilizarea unui server TURN. Tehnologia se potrivește în cazul apelurilor audio/video, unde volumul de date transmis este relativ mare și durata comunicării este lungă, dar pentru transmiterea de date generice este necesară studierea și stabilirea rentabilității ei, întrucât pentru un volum mic de date transmis într-o perioadă scurtă ar putea fi mai potrivită utilizarea unui server intermediar.

2.6. Rețele peer-to-peer în browserul web

Rețelele peer-to-peer sunt sisteme distribuite în care noduri cu roluri și capacități egale schimbă informații și servicii în mod direct între ele [9]. Acestea au devenit populare mulțumită potențialului lor de transfer al volumelor mari de date. De asemenea, în perioada mai recentă acestea au fost utilizate și datorită siguranței date de descentralizarea datelor în domenii unde inexistența unei autorități centrale cu influență asupra întregii rețele este un factor pozitiv. Un bun exemplu în acest sens este dat de monedele digitate precum Bitcoin [38], o valută electronică bazată pe o rețea peer-to-peer și pe folosirea unor funcții criptografice, care poate fi utilizată fără a necesita prezența unei bănci centrale sau a unui mecanism centralizat pe care toți participanții să îl considere de încredere.

În lucrarea sa, R. Schollmeier [35] a dat o definiție mai riguroasă pentru aceste arhitecturi. Acesta a subliniat diferența față de sistemele client-server, în cazul cărora un server central oferă toate serviciile și resursele clienților fără a solicita partajarea resurselor hardware deținute de aceștia. El a introdus conceptul de „servent”, adică noduri ale rețelei care servesc în același timp atât drept client, cât și server. În definiția oferită, pentru a fi considerată arhitectură peer-to-peer, o rețea distribuită trebuie să fie formată din participanți care partajează o parte din propriile resurse hardware (putere de procesare, spațiu de stocare, conexiune la rețea, etc.). Aceste resurse sunt necesare pentru a oferi conținutul și serviciile date de rețea și sunt accesibile de alți participanți direct, fără intervenția unui intermediar. Așadar, nodurile unui astfel de sistem găzduiesc și solicită concomitent resurse.

Tot în cadrul acestei lucrări s-au distins două tipuri de arhitecturi peer-to-peer, arhitecturi „pure” și arhitecturi hibride [35]. Pentru a avea un sistem peer-to-peer pur, pe

lângă definiția generală dată anterior, acesta mai trebuie să permită eliminarea oricărui nod ales arbitrar fără ca rețeaua să sufere vreo întrerupere a oricărui serviciu oferit. Dacă se identifică noduri care sunt absolut necesare pentru a putea oferi întregile servicii, rețeaua este considerată una hibridă.

În domeniul World Wide Web, accentul a fost pus mai mult pe potențialul rețelelor peer-to-peer de a transfera cu ușurință volume mari de date. În această categorie se află sistemul WebTorrent [21], care se folosește de acest potențial pentru decongestionarea serverelor web. Totuși, rețele precum Blogracy [15] se axează mai mult pe descentralizarea oferită de aceste arhitecturi, dar suferă de pe urma imposibilității la momentul respectiv de a integra întregul sistem în browserul web, oferind astfel clienților aceeași convenabilitate a utilizării dată de arhitecturile client-server.

Pe baza definițiilor menționate se pot identifica funcțiile necesare pentru ca un browser web să participe într-o rețea peer-to-peer. Un prim aspect este dat de partajarea resurselor hardware, lucru pe care se bazează funcționarea unei astfel de rețele. Am notat în capitolul 2.2 progresul făcut de browsere în sensul rulării directe a aplicațiilor prin intermediul scripturilor JavaScript [18]. Totodată, am menționat existența unor API-uri [45][47] care să permită salvarea locală a conținutului. Aceste aspecte dau posibilitatea browserului de a partaja atât putere de procesare cât și spațiu de stocare și creează un cadru care permite acestuia însușirea rolului de „servent” [35].

Un alt aspect necesar pentru integrarea într-o rețea peer-to-peer este dat de capacitatea de a stabili conexiuni directe între participanți și comunicarea cu aceștia fără intervenția unui intermediar. În cazul unei rețele hibride [35] există noduri speciale, care asigură o parte din servicii. Întrucât aceste noduri trebuie să rămână active în permanență, nu ar fi fezabilă utilizarea browserelor web pentru preluarea funcționalităților acestora, ci ar fi nevoie de un alt tip de aplicație independentă. În acest caz, nodurile simple reprezentate de browsere au mai multe opțiuni pentru a stabili conexiuni directe, una din acestea fiind WebSocket [23]. Totuși, variantele posibile sunt mai limitate în cazul conexiunilor browser-browser, necesare pentru ca acestea să participe atât în rețele peer-to-peer hibride cât și pure [35].

Tehnologia WebRTC [48] vine în sprijinul depășirii acestei probleme, deoarece introduce un API care permite comunicarea directă a oricăror date prin DataChannel [37] direct între browsere. Există însă o limitare dată de procesul de stabilire a conexiunii prin WebRTC care nu este standardizat și necesită un serviciu de signaling. Prin existența acestui serviciu se introduce un intermediar care ar putea să nu convină în anumite implementări peer-to-peer. Deci, este nevoie de stabilirea unui protocol convenabil de conectare a nodurilor

în cadrul rețelei prin care să se diminueze până la un nivel acceptabil impactul prezenței acestui intermediar. O posibilă soluție în acest sens ar fi ca alte noduri din rețea să preia responsabilitățile serviciului de signaling.

Încă un aspect de luat în calcul este volumul de resurse hardware pe care browserul îl poate oferi. Întrucât funcționarea serviciilor oferite de rețeaua peer-to-peer este asigurată de noduri, acestea s-ar putea dovedi insuficient de performante pentru anumite aplicații. Puterea de procesare oferită de browser nu este de regulă limitată de procesorul fizic al dispozitivului unde rulează, ci mai degrabă de formatul scripturilor JavaScript [18] utilizate. Acestea sunt primite în format text și interpretate linie cu linie de browser, proces care ar putea fi prea lent pentru aplicații distribuite unde este necesară o putere computațională mare, dar ar putea asigura suficientă putere de calcul pentru aplicații uzuale mai simple. În ceea ce privește spațiul de stocare, limitările depind de API-ul folosit și de browserul clientului. Nu există o anumită capacitate maximă impusă de toate browserele, însă de regulă API-ul IndexedDB [45] oferă suficient spațiu de stocare pentru aplicații uzuale, în condițiile în care acesta este disponibil pe dispozitivul utilizatorului.

O problemă importantă în orice rețea peer-to-peer este încrederea în alte noduri. Fără existența unui sistem central, care ar putea garanta orice informație primită de client, nodurile din rețea sunt libere să trimită informații eronate sau malițioase. O metodă de a asigura încrederea, utilizată și în rețeaua descentralizată Bitcoin [38], este folosirea unor funcții criptografice, mai exact a semnăturilor digitale, care pot garanta sursa oricărui conținut distribuit în rețea. Devine astfel responsabilitatea nodurilor să verifice aceste semnături și să respingă informațiile incorecte sau false. De asemenea, nodurile trebuie să genereze o semnătură digitală în momentul în care doresc introducerea de nou conținut în rețea, pentru ca acesta să fie acceptat de celelalte noduri.

Căutarea informațiilor în rețelele peer-to-peer este și ea o sarcină mult mai dificilă față de o arhitectură client-server. Dacă în cazul acestora toate datele erau centralizate și accesate prin intermediul serverului, într-o rețea peer-to-peer datele pot fi împărțite pe oricare din nodurile rețelei. Mai mult, datele care primesc actualizări ar putea rămâne pe unele dintre noduri într-o versiune veche, îngreunând și mai mult identificarea informațiilor corecte. Din această cauză, o operațiune simplă de căutare ar putea ajunge să necesite într-o rețea peer-to-peer pură [35] un mecanism foarte complex și relativ costisitor [9], prin intermediul căruia să fie interogate toate nodurile rețelei în cazul fiecărei căutări. În cazul sistemelor hibride [35] există posibilitatea folosirii nodurilor speciale din rețea pentru a optimiza prin diverse

mecanisme căutarea, cum ar fi salvarea unor tabele care să rețină nodurile unde se găsesc anumite informații.

În domeniul rețelelor de socializare, unde s-a evidențiat structura conexiunilor între persoane sub formă de graf [13], este posibilă folosirea unei arhitecturi peer-to-peer care să permită conectarea directă între browserele utilizatorilor. Descentralizarea acestui tip de sistem prezintă un beneficiu pentru libera circulație a ideilor publicate de clienți. Totuși, chiar dacă numărul de utilizatori este mare și este probabil ca aceștia să petreacă mult timp conectați [32][36], nodurile rețelei sunt browsere care nu oferă nicio garanție referitor la momentul deconectării, deci există posibilitatea ca niciun alt client să nu fie conectat la rețea la un moment dat. Publicarea unui conținut nou în acel timp nu ar avea astfel niciun efect. Întrucât o rețea de socializare trebuie să asigure consistența acestui conținut, în acest caz de utilizare ar fi mai potrivită o arhitectură peer-to-peer hibridă, cu noduri speciale care să asigure la nevoie salvarea conținutului publicat. Utilizarea acestui tip de arhitectură atrage și beneficiul posibilității eficientizării căutării utilizatorilor în sistem, o activitate foarte comună într-o rețea de socializare online [4][11].

2.7. Utilitatea criptografiei într-o rețea peer-to-peer

După cum am menționat și în capitolul anterior, într-o rețea peer-to-peer informațiile primite de la noduri nu pot fi considerate automat de încredere, în lipsa unui mecanism central care să le verifice. Algoritmii criptografici oferă o soluție la această problemă, fără a necesita un mecanism centralizat și asigurând un grad mare de siguranță. Aceștia au fost folosiți cu succes pe scară largă în diverse aplicații descentralizate, una dintre acestea fiind Bitcoin [38].

Criptografia reprezintă studiul sistemelor matematice care rezolvă două problematice de securitate: confidențialitatea și autentificarea [42]. Un sistem care asigură confidențialitatea previne extracția informației de către un terț neautorizat din mesaje transmise printr-un canal public, oferind garanția expeditorului unui mesaj că acesta va fi citit doar de către destinatarul dorit. Un sistem care asigură autentificarea previne inserarea neautorizată a mesajelor într-un canal de comunicare public, oferind garanția destinatarului că mesajul vine cu adevărat de la expeditorul pretins.

Utilizarea criptografiei are o istorie foarte lungă. Diffie și Hellman menționează în lucrarea lor [42] dificultatea metodelor vechi de codificare a mesajelor, datorată nevoii de a ține secret întreg procesul prin care mesajul securizat a fost obținut. Spre exemplu, în cazul cifrului Cezar, care presupune înlocuirea fiecărei litere cu o altă literă aflată la o distanță fixă în alfabet, descoperirea de către un terț a algoritmului prin care literele sunt schimbate i-ar permite să extragă informații din toate mesajele transmise folosind acest cifru, dar și din eventualele mesaje viitoare. După invenția telegrafului au apărut sisteme care făceau distincția între procesul de codificare și o anumită cheie folosită pentru securizarea fiecărui mesaj. Aceste sisteme nu mai depindeau de păstrarea secretă a întregului algoritm prin care mesajul a fost codificat, ci doar de siguranța cheii folosite. Chiar dacă aceasta ar fi descoperită, ea ar putea fi înlocuită cu ușurință obținând din nou mesaje sigure urmând același proces de codificare.

Acest tip de algoritmi, care folosesc aceeași cheie atât pentru codificarea cât și pentru decodificarea mesajelor poartă numele de algoritmi de criptare simetrici, sau algoritmi cu cheie partajată [27]. Deficiența acestora identificată de Diffie și Hellman [42] este dată tocmai de această partajare a cheilor. Pentru a transmite un mesaj securizat este necesară identificarea unui mijloc sigur de transmitere către destinatar a cheii pentru decodificare, cum ar fi predarea personală a acesteia într-un format fizic, proces de lungă durată, costisitor și de cele mai multe ori nefezabil. Această problemă a fost soluționată prin introducerea algoritmilor de criptare asimetrice, care folosesc două chei diferite pentru codificarea și respectiv decodificarea mesajelor.

Unul dintre cei mai utilizați algoritmi asimetrice este RSA [16], creat în anul 1978 de R. Rivest, A. Shamir și L. Adleman, motivația acestuia bazându-se pe lucrarea teoretică publicată de Diffie și Hellman [42]. Acest algoritm implementează două aspecte: criptografia folosind chei publice și semnăturile digitale.

Utilizarea cheilor publice se referă la faptul că, în algoritmul RSA cheile folosite la codificare sunt cunoscute de oricine, dar cheile necesare pentru decodificare sunt secrete [16]. Astfel, doar persoana cu acces la cheia de decodificare va putea extrage informațiile dintr-un mesaj. Fiecare expeditor și destinatar are propria pereche de chei publice și private, iar ele sunt construite în așa fel încât este extrem de greu de descoperit cheia privată pe baza celei publice. Pentru a expedia un mesaj confidențial, expeditorul îl va codifica utilizând cheia publică a destinatarului, acesta fiind singurul cu acces la cheia privată necesară pentru decodificare. Prin faptul că expeditorul nu are nevoie de nicio informație secretă a

destinatarului, se elimină nevoia de comunicare prin alte mijloace sigure a cheilor, pas care era necesar în cazul algoritmilor simetrici.

Semnăturile digitale sunt folosite atunci când destinatarul are nevoie să confirme identitatea expeditorului mesajului [16]. Acestea constau în atașarea în mesaj a unui cod generat de expeditor folosind cheia privată, autenticitatea acestui cod putând fi verificată cu ușurință de oricine folosind cheia publică a expeditorului. Codul semnăturii digitale depinde atât de cheia privată cât și de conținutul mesajului, deci orice semnătură este unică pentru un anumit mesaj. Astfel, expeditorul nu are posibilitatea de a nega ulterior semnarea unui mesaj. Mai mult, prin utilizarea cheii private se exclude posibilitatea falsificării semnăturii de către un terț neautorizat.

Deci, criptografia folosind chei publice se ocupă de confidențialitatea mesajelor, iar semnăturile digitale asigură autentificarea expeditorului. Un mesaj poate fi semnat utilizând cheia privată a expeditorului, apoi codificat folosind cheia publică a destinatarului, profitând astfel de ambele beneficii oferite de cele două tehnici [16].

Algoritmul RSA se bazează pe folosirea în procesul de codificare a funcțiilor „trapă unidirecțională” [16]. Aceste funcții trebuie să îndeplinească anumite criterii: decodificarea unui mesaj codificat rezultă mesajul original, atât codificarea cât și decodificarea sunt procese ușor de calculat, iar expunerea publică a procedurii de codificare nu compromite secretul necesar pentru decodificare. Aceste funcții sunt numite „trapă” deoarece procedura inversă de decodificare este ușoară doar dacă se cunoaște o anumită informație (cheia secretă), iar în lipsa acesteia este aproape imposibilă. De asemenea, sunt numite „unidirecționale” deoarece sunt ușor de calculat într-o singură direcție, procesul invers fiind extrem de dificil.

Funcțiile folosite de RSA mai îndeplinesc o cerință: procedura de decodificare poate fi aplicată direct unui mesaj necodificat, iar aplicarea pe mesajul rezultat a procedurii de codificare va rezulta în mesajul inițial. Această caracteristică este necesară pentru a putea oferi semnături digitale într-o manieră simplă, folosind aceleași proceduri și aceleași perechi de chei publice și private [16].

Un alt tip de funcții utilizate adesea în criptografie este reprezentat de funcțiile hash. Acestea sunt funcții care comprimă datele de intrare primite la o mulțime mai mică de valori posibile, deci datele de ieșire sunt de regulă mai mici ca dimensiune decât cele de intrare [27]. De obicei, aceste funcții primesc date de lungimi arbitrare și le transformă în secvențe de dimensiuni fixe. Și acestea sunt funcții unidirecționale, ele sunt foarte ușor de calculat într-o direcție, dar având datele de ieșire este imposibil de determinat în sens invers ce set de date

de intrare le-au generat. De asemenea, aceste funcții sunt deterministe, deci aceleași date de intrare vor rezulta de fiecare dată în aceleași date de ieșire.

În practică, funcțiile hash sunt utilizate adesea în generarea semnăturilor digitale [27]. Conținutul care trebuie semnat digital este de regulă un șir de caractere de dimensiune arbitrară, iar algoritmul RSA ar putea necesita un efort computațional prea mare în cazul datelor de intrare foarte lungi. Din această cauză este mai eficientă generarea unei semnături digitale pentru rezultatul unei funcții hash aplicate acestor date. La verificarea semnăturii, destinatarul va aplica din nou aceeași funcție hash pentru a putea determina autenticitatea mesajului, abordare bazată pe determinismul acestor funcții. Totuși, se introduce astfel și posibilitatea unor atacuri: semnătura va fi validă pentru orice conținut care generează același hash ca și datele inițiale. Din acest motiv, este obligatorie folosirea unor funcții hash sigure, pentru care este nefezabilă computațional descoperirea altor date de intrare care generează același rezultat.

Din punct de vedere al eficienței, există mulți algoritmi de criptare simetrici mai rapizi decât algoritmul RSA. Din această cauză, o practică uzuală este utilizarea algoritmului RSA pe post de canal sigur de transmitere a cheii unui astfel de algoritm simetric [16]. Astfel, conținutul mesajului este codificat folosind un algoritm mai eficient, iar cheia utilizată este securizată folosind RSA. Destinatarul va putea decodifica în primă fază această cheie, iar mai apoi o va folosi pentru accesarea conținutului mesajului.

Așadar, algoritmul RSA permite crearea unui sistem sigur de transmitere a informațiilor prin intermediul unor canale publice de comunicare [16]. În cadrul acestuia, expeditorii și destinatarii mesajelor sunt identificați prin cheile publice pe care le dețin, iar siguranța informațiilor comunicate depinde de păstrarea secretă a cheilor private asociate.

3. Proiectarea și implementarea unei rețele de socializare peer-to-peer

3.1. Structura aplicației

Soluția aleasă urmărește realizarea unei rețele de socializare descentralizate, accesibilă și utilizabilă prin World Wide Web folosind un browser web, în cadrul căreia comunicațiile între clienți se realizează într-o manieră peer-to-peer. Pentru atingerea acestui scop, au fost implementate două module distincte: un modul client, cu care interacționează direct persoanele care vor beneficia de funcționalitățile rețelei, și un modul server, proiectat și el într-o manieră descentralizată în sensul că este capabil să coexiste alături de oricâte module similare, administrate de entități diferite, situate în locații și rețele diferite. De asemenea, este necesar un server web generic prin intermediul căruia modulul client să fie distribuit către browserele web ale persoanelor care accesează aplicația pentru prima dată. Deci, cerințele minimale pentru funcționarea aplicației se rezumă la posibilitatea utilizatorului de a accesa modulul client, fie prin intermediul acestui server web, fie prin alte metode precum preluarea acestuia dintr-un cache la accesări repetate, și la funcționarea a cel puțin un modul server în rețeaua peer-to-peer.

Funcționalitățile puse la dispoziție de această aplicație web acoperă principalele caracteristici ale rețelelor de socializare moderne realizate în cadrul paradigmei Web 2.0 [20]. Acestea cuprind posibilitatea creării unui cont de utilizator și autentificarea ulterioară pe baza acestuia, crearea automată a unui profil pentru fiecare cont, conținutul acestuia putând fi modificat oricând, posibilitatea publicării în cadrul profilului a unor postări, căutarea și vizualizarea altor profiluri, posibilitatea de a realiza conexiuni cu acestea, mai exact este posibilă urmărirea sau blocarea acestora, precum și posibilitatea de a șterge unele postări sau chiar întregul cont de utilizator.

3.1.1. Modulul client

Modulul client a fost realizat sub forma unui Progressive Web Application [6], având o structură de tip Application Shell [5]. Acest modul este proiectat să ruleze exclusiv în cadrul browserului utilizatorului, folosindu-se de capacitatea acestuia de a executa scripturi JavaScript [18]. El este capabil să stabilească toate conexiunile necesare pentru integrarea în

rețeaua peer-to-peer și să gestioneze datele transmise prin această rețea, prezentând utilizatorului o interfață grafică pentru accesarea funcționalităților disponibile. Ca și limbaj de programare, a fost folosit în cea mai mare parte TypeScript [19], datorită caracteristicilor sale care duc la o dezvoltare mult mai ușoară a aplicațiilor complexe față de JavaScript, dar păstrând în același timp compatibilitatea cu acesta, deci permițând rularea cu ușurință în browserul web.

În cadrul acestui modul au fost implementate mai multe componente cu roluri specifice. În primul rând, există pachetul „connection”, care are ca principal obiectiv transmiterea și primirea datelor din rețeaua peer-to-peer. Prin intermediul lui, se realizează două tipuri de conexiuni: o conexiune cu un modul server și multiple conexiuni cu alte module client.

Conexiunea cu serverul este realizată la pornirea aplicației. Aceasta se stabilește în urma unui protocol de autentificare care va fi discutat în capitolul 3.3.3. După finalizarea acestuia, conexiunea rămâne deschisă pe toată perioada rulării aplicației, timp în care se realizează o comunicare bidirecțională între client și server. În cazul în care apare o problemă cu această conexiune, fie o problemă de rețea, fie serverul nu mai răspunde, se va stabili automat o nouă conexiune cu un alt server din rețea. Pachetul connection reține în permanență o listă de adrese ale altor servere din rețea pentru evenimente de acest tip.

Celălalt tip de conexiune este aceea cu alte module client. Aceste conexiuni sunt mult mai instabile deoarece utilizatorii pot închide oricând aplicația, pot pierde conexiunea la internet sau pot suferi alte dificultăți tehnice cu dispozitivul folosit, toate acestea fiind situații care produc terminarea neașteptată a comunicării. Din acest motiv, pachetul connection încearcă realizarea cât mai multor conexiuni directe cu alți clienți, acordând prioritate utilizatorilor urmăriți, dacă aceștia sunt conectați la rețea, sau utilizatorilor ale căror pagini sunt frecvent accesate. Totodată, această prioritate are și rolul de a optimiza transferurile de date în rețea deoarece se presupune că în medie utilizatorii vor fi interesați să reviziteze paginile create de persoane apropiate, sau paginile pe care le-au frecventat în trecut.

Odată cu stabilirea acestor conexiuni, pachetul connection preia responsabilitatea de a trimite solicitări în rețea în funcție de acțiunile utilizatorului. Prin obiectul „ConnectionManager” se analizează aceste solicitări și se stabilește cel mai potrivit canal de comunicare prin care să fie trimise. De asemenea, răspunsurile primite sunt trecute mai întâi printr-un proces de validare și abia apoi trimise către interfața utilizatorului. Acest pas este necesar deoarece nu se poate garanta că peer-ul care a răspuns este de încredere. În orice punct al rulării aplicației se presupune că se poate primi un mesaj malițios, incomplet sau

greșit din rețea, fiecare modul client având responsabilitatea de a verifica înainte de a accepta un astfel de mesaj.

Tot în responsabilitatea pachetului connection revine tratarea notificărilor care pot ajunge la client din rețea, precum și tratarea solicitărilor venite de la alte module din rețea. Arhitectura rețelei se bazează pe faptul că majoritatea clienților conectați sunt dispuși să salveze date pe propriul dispozitiv și să le transfere la nevoie către alți utilizatori.

Procesul de validare cade în responsabilitatea obiectului „Validator”, care pune la dispoziție funcții capabile să analizeze conținutul unui mesaj primit din rețea și să confirme că respectă caracteristicile așteptate ale acestuia. În acest scop, au fost definite interfețe pentru toate obiectele așteptate de la alte entități din rețea, iar fiecare modul client are responsabilitatea să le aplice înainte de a trimite obiectele spre alte module. În afară de conținutul efectiv al obiectului primit, funcțiile de validare mai verifică și dimensiunea acestuia, respingând astfel obiectele nejustificat de mari care ar putea avea un scop malițios, cum ar fi umplerea cu date inutile a spațiului de stocare alocat de client.

Spațiul local de stocare este administrat de pachetul „storage”. Acesta se folosește de API-ul IndexedDB [45] pentru a salva, accesa, actualiza sau șterge datele salvate local. Totuși, alegerea și favorizarea unor date pentru a fi salvate revine modulului de control, pachetul storage intervenind doar pentru eliberarea spațiului de stocare dacă s-a atins o anumită limită predefinită.

Tot în cadrul modulului client există și un pachet care se ocupă cu operațiunile criptografice, numit „encryption”. Acesta are multiple utilizări, fiind apelat în momentul înregistrării în rețea pentru generarea unei perechi noi de chei publice/private RSA, la autentificare pentru obținerea cheilor publice/private RSA pe baza parolei contului, la efectuarea unei noi postări pentru aplicarea unei semnături digitale dar și la validarea datelor primite din rețea, pentru verificarea semnăturilor aplicate de alți utilizatori. Tot acest modul este capabil să creeze datele de stabilire a conexiunii cu un alt peer, evitând astfel posibilitatea interceptării acestor date de către un intermediar.

Ultimul element al acestui modul este interfața grafică. Prin această componentă utilizatorul interacționează cu aplicația și poate utiliza funcționalitățile puse la dispoziție de aceasta. Se disting două tipuri de pagini afișate: pagina de autentificare, prin intermediul căreia o persoană poate înregistra un cont nou sau se poate autentifica în rețea cu un cont existent, și pagina de vizualizare a unui profil, prin intermediul căreia se accesează restul funcționalităților, inclusiv cele care țin de administrarea profilului personal.

Așadar, la pornirea unui modul client utilizatorul trece prin procesul de autentificare. După acest pas, în memoria locală rămân salvate profilul și postările acestuia, asigurându-le disponibilitatea în rețea pe perioada în care aplicația este deschisă. În spațiul de stocare local se păstrează aceste date și după închiderea aplicației, asigurând disponibilitatea acestora în rețea de oricâte ori aplicația este deschisă de pe acel dispozitiv. Totuși, autentificarea este necesară la fiecare utilizare deoarece în urma ei se generează cheia privată necesară modului client pentru a realiza postări noi, iar salvarea acestuia în spațiul de stocare local nu este considerată sigură.

3.1.2. Modulul server

Modulul server este proiectat ca o aplicație JavaScript care se integrează în rețeaua peer-to-peer prin conexiuni WebSocket [23] cu modulele client. Serverul poate rula în rețea alături de alte module server realizând și cu acestea conexiuni de același tip, clienții putând transmite date către unul din oricare aceste module. Rolul principal care le revine este acela de a asigura consistența și durabilitatea conținutului distribuit în cadrul întregii rețele, dar servesc și rolul de a facilita autentificarea utilizatorilor și conectarea directă a acestora cu alți clienți, precum și de a realiza căutarea unor profiluri în întregul sistem. Fiecare modul client, pe parcursul funcționării, păstrează o conexiune către un modul server din rețea.

Controlul funcțiilor modului server este preluat de cele două componente de comunicare distincte: comunicarea cu modulele client și comunicarea cu alte module server. Acestea funcționează similar, așteptând și tratând cereri primite din rețea, diferența fiind dată de tipurile de cereri care sunt diferite în fiecare din cele două componente. Încă o diferență este dată de gestiunea conexiunilor, în cazul serverelor încercându-se conectarea la fiecare modul nou descoperit, iar în cazul clienților conectarea realizându-se la cererea acestora, pentru perioada dorită de ei, impunându-se condiția ca numărul maxim de clienți conectați pe modulul server curent să nu depășească o anumită limită predefinită.

Pentru a trata cererile primite, modulul server apelează la componenta internă „storage”, responsabilă cu salvarea locală a datelor, și încearcă să formeze un răspuns pe baza acestor date locale. Modulele server salvează de regulă toate datele pe care le primesc din rețea, în încercarea de a garanta că toate profilurile și postările sunt în permanență disponibile, chiar dacă în rețea există un număr mic de clienți. Totuși, în cazul unor solicitări s-ar putea ca datele să nu existe local, sau să necesite o verificare pentru actualizări, moment în care modulul server apelează la celelalte module server din rețea.

Similar cu modul de operare al modulelor client, și serverele verifică corectitudinea datelor primite din rețea, mai exact a profilurilor și a postărilor, folosind componenta internă „validator”. În cadrul acesteia se realizează și singura operație criptografică necesară pe server, verificarea semnăturilor digitale. În urma unei validări reușite, modulul server are garanția că profilul de utilizator și eventuala secvență de postări primită odată cu acesta sunt nemodificate față de momentul când au fost publicate, conțin toate câmpurile necesare și nu includ conținut malițios, deci pot fi salvate local în siguranță și pot fi trimise mai departe când sunt solicitate.

În ansamblu, modulul server poate fi văzut ca un modul client lipsit de un profil de utilizator propriu și de interfață grafică. Acesta nu realizează conținut nou în rețea, dar în schimb salvează toate datele primite. Iar legat de conexiuni, la fel cum modulul client favorizează conexiunile cu utilizatori cunoscuți, și modulul server încearcă să rămână în permanență conectat cu alte servere.

3.1.3. Alte aspecte arhitecturale

Partea de distribuire a aplicației, mai exact a modulului client, a fost realizată prin intermediul unui server web. Singura funcționalitate a acestuia este de a primi o cerere HTTP de la browserul persoanei care dorește să acceseze aplicația, cerere la care răspunde cu Application Shell-ul necesar rulării modulului client. Acest server web este total independent de rețeaua peer-to-peer și de conținutul transmis prin aceasta. El trimite odată cu această „carcasă” a aplicației și o listă de adrese la care modulul client poate apela pentru a găsi un modul server care să îi permită autentificarea în rețea. După această unică cerere HTTP, serverul web nu mai este contactat. Dacă în rețeaua utilizatorului, sau chiar în cadrul browserului său, există un modul cache care reține răspunsul primit, accesările ulterioare ale aplicației vor putea prelua modulul client direct din acest cache fără nevoia intervenției serverului web.

Întregul sistem al aplicației a fost structurat în așa fel încât să poată face față unor valori de vârf ale traficului, adesea identificat în cazul rețelelor de socializare [26]. Având o arhitectură distribuită, clienții pot comunica între ei independent de starea vreunui sistem central. De asemenea, modulele server comunică între ele pentru distribuirea la nevoie a clienților astfel încât să se prevină supraîncărcarea. Totodată, solicitările spre ele, oricum minimale în ceea ce privește funcționalitățile puse la dispoziția utilizatorului, devin din ce în ce mai rare pe măsură ce clienții reușesc să stabilească mai multe conexiuni directe între ei.

3.2. Date gestionate de aplicație

Conținutul gestionat prin intermediul aplicației a fost împărțit în două tipuri de entități: profiluri de utilizator și postări. Acestea încapsulează toate informațiile distribuite de fiecare utilizator al rețelei și sunt transmise peer-to-peer între modulele conectate.

3.2.1. Profiluri de utilizator

Orice rețea de socializare trebuie să ofere utilizatorilor posibilitatea creării unui spațiu personal care să conțină datele pe care aceștia doresc să le facă vizibile celorlalte persoane care folosesc rețeaua. Această aplicație încapsulează aceste informații în obiecte de tip „UserProfile”. Acestea conțin anumite proprietăți standardizate, pentru a permite distribuirea ușoară către modulele server și către browserele celorlalți utilizatori.

În primul rând, profilurile conțin unele elemente de identificare. Fiecare are asociat un identificator unic, un nume de utilizator, o cheie publică și o altă cheie publică specială, numită cheie de recuperare. Toate aceste câmpuri sunt asociate contului la crearea acestuia.

Cheia publică de recuperare permite efectuarea celor mai importante operații de administrare a contului: schimbarea parolei și ștergerea acestuia. La înregistrare, utilizatorul primește o cheie privată asociată cheii publice de recuperare, iar în baza acesteia are accesul să semnaleze în rețea aceste operații administrative. Este numită cheie de recuperare deoarece prin intermediul ei utilizatorul are posibilitatea să își schimbe parola și să recupereze astfel accesul la cont, în cazul în care acesta a fost pierdut. Cheia privată asociată este vizibilă doar la înregistrarea contului, moment după care utilizatorul trebuie să o mențină secretă, dar accesibilă în cazul în care este nevoit să o folosească.

Identificatorul unic este un șir de caractere de o dimensiune fixă, generat aleatoriu la înregistrare. El nu poate fi schimbat pe toată durata utilizării rețelei, fiind elementul principal pe baza căruia conturile sunt distinse. Chiar și după ștergerea unui profil utilizând cheia de recuperare, identificatorul unic rămâne salvat în rețea și nu poate fi refolosit.

Numele de utilizator este ales de fiecare persoană în momentul înregistrării. El servește ca mijloc facil de căutare a unor profiluri, fiind de regulă într-un format ușor de înțeles și recunoscut de utilizatori umani. Pentru a ușura sincronizarea datelor între nodurile rețelei, el nu poate fi schimbat, dar prin actualizarea la nevoie a mecanismului de identificare a profilurilor s-ar putea permite schimbarea acestuia folosind cheia de recuperare.

Cheia publică asociată profilului este structurată asemănător cheii publice de recuperare, dar ea depinde de numele de utilizator și de parola contului, deci poate fi modificată la nevoie prin schimbarea datelor de autentificare. Celelalte noduri ale rețelei verifică folosind această cheie publică validitatea semnăturilor digitale realizate de acel utilizator. Această cheie privată este generată pe baza numelui de utilizator și a parolei în momentul autentificării, oferind astfel și un mecanism de verificare a datelor de autentificare introduse: introducerea unui nume de utilizator sau a unei parole greșite vor duce la generarea unei chei private greșite, lucru care va conduce la respingerea autentificării. Procesul de autentificare este explicat pe larg în capitolul 3.3.3.

Tot în sfera datelor de identificare se află și numărul de postări. Acesta este salvat în UserProfile pentru a servi pe post de mecanism de stabilire a versiunii: nodurile rețelei recunosc mereu profilurile cu un număr mai mare de postări ca fiind cea mai actualizată versiune a datelor. Astfel, la detectarea a două obiecte UserProfile cu același identificator unic, dar cu număr diferit de postări, obiectul cu numărul mai mare de postări va fi păstrat. Pentru a preveni inserarea malițioasă în rețea a unor obiecte cu un număr mai mare de postări decât cel real, la fiecare actualizare a profilului se introduce în acesta o semnătură digitală realizată folosind cheia privată asociată cheii publice curente a utilizatorului.

Pe lângă datele de identificare, în UserProfile sunt salvate și date opționale, completate la dorința utilizatorului. Spre deosebire de datele de identificare, acestea nu vor fi distribuite oriunde în rețea, fiecare persoană putând opta să blocheze anumiți utilizatori atenționând celelalte noduri ale rețelei să nu mai transmită aceste informații spre aceștia. Datele opționale cuprind câmpuri specifice rețelelor de socializare, cum ar fi numele real, o descriere personală scrisă de utilizator, o imagine de profil, data nașterii, nivelul de studii sau localitatea de domiciliu. Aceste date pot fi completate, modificate sau șterse oricând de către utilizator cu ușurință.

Tot în cadrul UserProfile sunt salvate listele de utilizatori urmăriți și de utilizatori blocați. Având acces la utilizatorii urmăriți, nodurile rețelei pot sugera modulului client conexiuni directe cu alte module folosite de acești utilizatori, în cazul în care observă că ambii sunt conectați în același timp. Lista de utilizatori blocați este accesată pentru a limita corespunzător datele opționale și postările care sunt distribuite unor persoane.

Toate aceste date care nu sunt esențiale la identificarea profilurilor pot primi des actualizări. Din această cauză, a fost introdus un mecanism de stabilire a versiunii datelor, care permite eficientizarea transferurilor de date în rețea. Câmpurile au fost împărțite în mai multe părți, fiecare având propriul număr de versiune reprezentat prin momentul ultimei

modificări, în milisecunde. Astfel, la transferul profilurilor în rețea pot fi omise acele date care sunt deja în cea mai actualizată formă pe nodul unde sunt trimise. Totodată, în cazul postărilor noi este adesea mai avantajoasă trimiterea unui obiect UserProfile care conține doar elementele de identificare și semnătura digitală care atestă noul număr de postări.

Un alt element important al UserProfile este o variabilă de tip boolean numită „deleted”, care este setată ca adevărată doar dacă acel profil a fost șters folosind cheia de recuperare. În urma identificării acestei proprietăți de către orice nod al rețelei, toate datele opționale și postările sunt înlăturate, păstrând doar datele de identificare. Salvând aceste date, orice nod interesat la un moment ulterior de acest profil poate fi notificat despre stergerea acestuia.

3.2.2. Postări

Postările sunt entitățile prin intermediul cărora sunt introduse informații noi în rețea. Ele pot reprezenta simplu conținut publicat de utilizatori, dar pot conține și informații speciale generate la orice acțiune de actualizare a datelor din profil. Astfel, orice noutate apărută în cadrul rețelei trebuie să fie generată de o postare. Postările alături de toate câmpurile necesare transmiterii lor în rețea sunt încapsulate în obiecte de tip „Post”.

Fiecare postare trebuie să conțină un identificator unic reprezentând numărul de postări ale profilului de la momentul publicării acesteia. Întrucât acest număr este incrementat la orice postare nouă, identificatorii unici ai postărilor vor fi numere naturale consecutive, unice pentru fiecare profil. Prima postare are identificatorul 1 și este publicată la crearea profilului. În cazul ștergerii acestuia, postarea cu cel mai mare identificator va fi realizată la ștergerea contului. Nodurile rețelei nu acceptă postări noi cu același identificator ca o postare mai veche, pentru același profil.

Ca și element de securitate, orice postare conține obligatoriu o semnătură digitală care atestă faptul că ea a fost publicată de deținătorul acelui profil.

Pentru a determina tipul postării se folosește un câmp numeric în cadrul fiecărui obiect numit „postType”. Acesta distinge între următoarele tipuri:

- postări de conținut – sunt fragmente de text scrise și publicate de fiecare utilizator, vizibile în cadrul paginii personale aferente profilului personal,
- postări de actualizare a profilului – sunt realizate automat în momentul efectuării unei operații de actualizare a datelor din profil, sau în momentul creării contului,

- postări de ștergere a unei alte postări – sunt realizate în momentul în care utilizatorul dorește ștergerea unei postări de conținut publicate anterior și au rolul de a notifica nodurile rețelei în vederea ștergerii conținutului acesteia,
- postări de schimbare a cheii publice – sunt realizate în momentul schimbării datelor de autentificare și creării în acest fel a unei noi chei publice, aceasta fiind utilizată în continuare de nodurile din rețea,
- postări de ștergere a contului – apariția unei astfel de postări semnalează ștergerea definitivă a contului utilizatorului.

La publicarea unei noi postări, modulul client trimite automat și varianta actualizată a obiectului UserProfile, cu noul număr de postări și noua semnătură digitală corespunzătoare.

Singurul tip de postări vizibile în pagina asociată profilului utilizatorului sunt cele de conținut. Celelalte tipuri de postări au un efect funcțional, procesat în fundal de modulul client. Totuși, chiar dacă nu sunt vizibile în interfața grafică, ele sunt salvate în lista de postări a fiecărui utilizator și sunt distribuite la nevoie către alte noduri ale rețelei.

Datorită nevoii de a incrementa în permanență numărul de postări, deci și identificatorul unic al postărilor noi, procesul de ștergere necesită crearea unui nou obiect Post care include identificatorul postării care se dorește ștersă. În urma operației de ștergere, nodurile păstrează identificatorul postării șterse împreună cu un câmp numit „deleted” setat ca adevărat. În acest fel există un obiect chiar și pentru conținutul șters, deci nu se creează impresia lipsei unor postări și nevoia de a le căuta în rețea, iar identificatorul unic al obiectului vechi nu va putea fi refolosit. Singurul tip de postare care poate fi ștersă este cea de conținut.

În cazul postărilor de schimbare a cheii publice, acestea trebuie să includă vechea cheie publică a profilului. În acest fel, la nevoie, semnăturile digitale mai vechi realizate cu această cheie vor putea fi verificate.

3.2.3. Elemente de criptografie

Comunicând într-un context peer-to-peer, modulele din cadrul rețelei trebuie să poată verifica autenticitatea obiectelor primite, atât a profilurilor de utilizator cât și a postărilor. În acest sens au fost introduse cele două chei publice disponibile în profil, cu ajutorul cărora se poate valida orice postare sau actualizare a profilului utilizatorului.

Astfel, cele două tipuri de entități transmise în rețea conțin obligatoriu câte o semnătură digitală. Aceste semnături digitale trebuie să acopere câmpurile obiectelor pe care

le validează, inclusiv identificatorul unic. În cazul profilurilor, validarea este împărțită în mai multe părți în funcție de mecanismul de stabilire a versiunilor, pentru a permite transmiterea separată a unor fragmente din profil. Astfel, anumite părți ale UserProfile pot fi actualizate în siguranță, fără a necesita transmiterea întregului obiect. Fiecare modul din rețea, la primirea entităților, are responsabilitatea să verifice corectitudinea semnăturilor digitale atașate și să respingă obiectele semnate incorect. În acest fel, odată semnat un obiect nu va putea fi modificat de către un nod malițios al rețelei.

Toate tipurile de postări, cu excepția postărilor de schimbare a cheii publice și a celor de ștergere a contului, sunt semnate utilizând cheia privată asociată cheii publice a profilului. De asemenea, fiecare versiune nouă a UserProfile, adică având un număr mai mare de postări, va necesita o semnătură digitală realizată în același mod.

Un posibil mod de atac al semnăturii digitale din profilul de utilizator ar putea fi introducerea în rețea a unei variante noi a UserProfile, cu un număr mai mare de postări, dar și cu o nouă cheie publică introdusă de atacator. În acest fel, și semnătura digitală asociată obiectului ar fi validă. Totuși, acest tip de atac este contracarat prin nevoia publicării unei noi postări care să ateste schimbarea cheii publice, iar în lipsa acesteia profilul fals va fi respins de fiecare nod al rețelei. Astfel, pentru a putea oferi o siguranță sporită a conturilor, orice acțiune de modificare a profilului necesită realizarea unei noi postări.

Deoarece cheia privată asociată cheii publice din UserProfile este necesară pentru crearea majorității postărilor noi, ea trebuie să fie reținută la nivelul modulului client pe toată durata sesiunii de utilizare a aplicației. Astfel, este creată posibilitatea ca un atacator să acceseze aceste date secrete salvate în browser și să preia accesul asupra contului. Acesta este motivul pentru care a fost introdusă cheia publică de recuperare, cheia privată asociată ei putând fi păstrată într-un format mult mai sigur, lăsat la alegerea utilizatorului. Această cheie privată este necesară doar în cazul schimbării datelor de autentificare sau a ștergerii contului, ea fiind folosită pentru a semna digital postările de tipurile corespunzătoare.

3.2.4. Alte date salvate local

Modulul client reține pe toată durata funcționării o listă de adrese ale unor module server. În cazul în care conexiunea cu modulul server contactat inițial este pierdută, se încearcă stabilirea unei noi conexiuni cu un alt modul de la una din adresele din această listă. Lista este primită odată cu „scoica” aplicației web și poate fi extinsă prin solicitări către alte module server sau client din rețea.

Tot în cadrul modulului client trebuie să existe după o autentificare cu succes cea mai recentă versiune a profilului utilizatorului acelui cont, dar și lista completă a postărilor sale. De asemenea, după autentificare va rămâne salvată și cheia privată asociată cheii publice a profilului.

Chiar dacă ar fi posibilă salvarea în memoria locală a acestei chei private, ea este ștearsă intenționat la terminarea sesiunii de utilizare a aplicației. Salvarea ei ar face posibilă accesarea aplicației fără nevoia reintroducerii datelor de autentificare, dar ar face mai ușoară accesarea ei de către un atacator. Considerând riscul de securitate implicat de lipsa unor metode de salvare sigură a datelor în browser, s-a decis ca această cheie privată să nu fie reținută în spațiul de stocare al dispozitivului.

3.3. Rețeaua peer-to-peer

În cadrul implementării, modulele client și server sunt conectate direct între ele și transferă informații sub forma profilurilor de utilizator și a postărilor. În acest sens, ele formează o rețea peer-to-peer care stă la baza serviciilor oferite de aplicație.

3.3.1. Topologia rețelei. Canale de comunicare

Rețeaua cuprinde două tipuri de noduri: modulele client și modulele server. Cerința minimală pentru asigurarea funcționării rețelei este existența a cel puțin un modul server. Astfel, rețeaua peer-to-peer formată nu este una „pură”, cu noduri având capacități egale, ci una hibridă [35].

Conexiunile realizate în rețea sunt de două tipuri: conexiuni prin WebSocket între modulele server sau între un modul client și unul server, și conexiuni prin WebRTC între două module client.

Modulele server încearcă să păstreze în permanență câte o conexiune cu fiecare alt modul server din rețea. În acest fel se obține o comunicare cât mai rapidă și eficientă între ele. Această bună integrare între servere face necesară conectarea modulelor client la câte un singur modul server, acesta putând transmite cererile sale celorlalte servere.

În ceea ce privește conexiunile între modulele client, acestea încearcă să stabilească oricât de multe conexiuni, preferând comunicarea cu nodurile corespunzătoare unor profiluri care se găsesc în lista de persoane urmărite a utilizatorului. Această favorizare se bazează pe

probabilitatea ridicată ca profilurile accesate de utilizator să fie disponibile pe aceste noduri. În lipsa unor astfel de noduri favorizate, modulul client se va conecta la oricare alte noduri. Numărul total de module client la care se va realiza conectarea poate fi stabilit de utilizator în funcție de resursele de sistem și rețea disponibile.

Întrucât modulele client rulează în browsere web, conexiunile client-client nu oferă siguranță în legătură cu timpul cât vor rămâne online și pot fi întrerupte neașteptat în orice moment. Pe de altă parte, modulele server se presupune că vor rămâne conectate mai mult timp, deci conexiunile cu acestea ar trebui să fie mai stabile. Astfel, conexiunile WebSocket vor rămâne mai mult timp deschise și organizate uniform în rețea, în timp ce conexiunile WebRTC creează o subrețea cu o topologie imprevizibilă.

Considerând aceste proprietăți ale conexiunilor, rețeaua peer-to-peer creată ia forma unui graf cu două tipuri distincte de noduri, și două tipuri distincte de muchii.

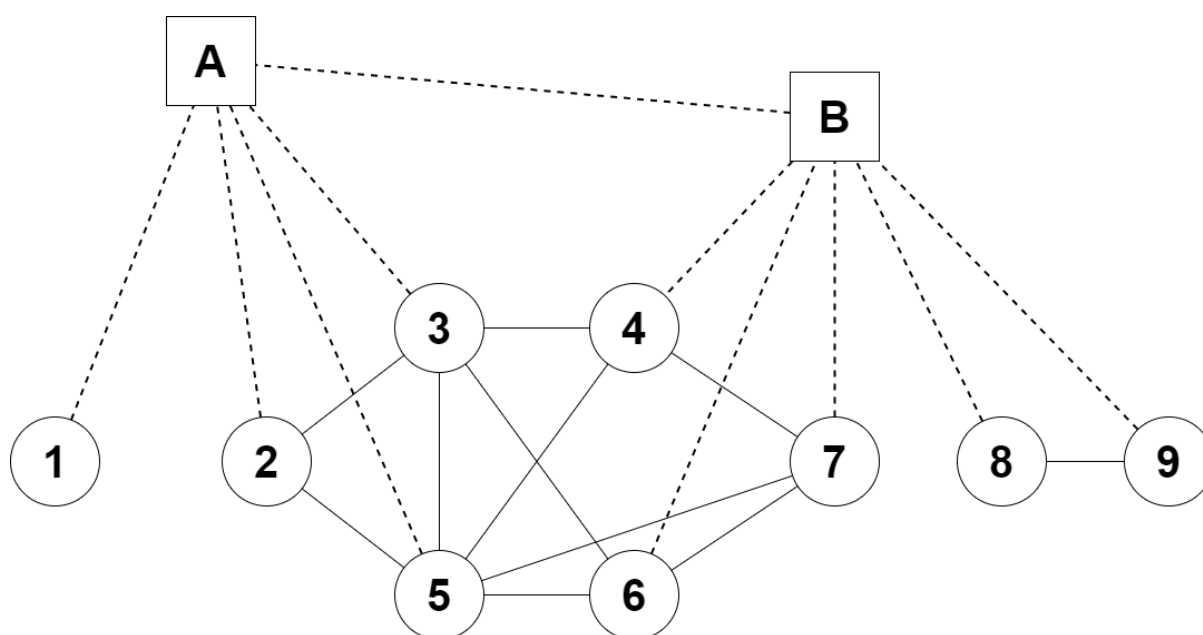


Fig. 2: Exemplu de topologie a unei rețele formate din două module server (A și B) și nouă module client (1-9)

În exemplul din figura 2 au fost evidențiate cele două tipuri de conexiuni: WebSocket, între cele două servere și între fiecare client și un server, și WebRTC între modulele client. De asemenea, au fost surprinse câteva cazuri particulare: modulul client 1 nu are nicio conexiune stabilită cu alte module client deoarece tocmai a încheiat procesul de autentificare

și urmează să primească o sugestie de conectare de la serverul A, iar modulele client 8 și 9 sunt conectate între ele, dar separate de restul clienților, urmând să primească de la serverul B noi sugestii de conectare și să se integreze mai bine în rețea. Noduri precum 3 sau 5 au stabilit un număr mare de conexiuni cu module client, conectate atât la serverul A cât și la serverul B, și au acces mai facil la resursele rețelei. În baza numărului mare de legături stabilite, este probabil ca unul din aceste două noduri să sugereze următoarea nouă conexiune pentru nodul 2.

În cazul în care un client stabilește o conexiune cu un modul server, dar acesta nu îi poate oferi sugestii pentru conectarea la alți clienți sau oferă un număr mic de opțiuni de conectare, se va încerca stabilirea conexiunii cu un alt modul server. În acest fel se evită izolarea unui modul client conectat doar la un modul server și se face posibilă conectarea între clienți ce au stabilit conexiuni cu servere distincte, prin păstrarea conexiunilor WebRTC deja stabilite în momentul schimbării serverului.

Alte situații când serverul poate fi schimbat sunt atunci când acesta este prea aglomerat, se închide sau sugerează intenționat acest lucru.

3.3.2. Protocolul de comunicare în rețea

Toate conexiunile stabilite în rețeaua peer-to-peer rămân deschise pentru a permite transferuri repetate de informații între noduri pe toată perioada cât sunt conectate. Astfel, a fost stabilit un protocol bazat pe cereri și răspunsuri prin intermediul căruia modulele pot solicita la nevoie anumite profiluri de utilizator sau postări.

Fiecare solicitare are atașat un identificator unic, creat aleatoriu de nodul care lansează cererea. De asemenea, au fost stabilite niște tipuri de cereri în funcție de care variază atât modul de interpretare a cererii cât și conținutul acesteia.

Primul tip de cerere este cererea de înregistrare în rețea. Aceste cereri sunt trimise doar de la un modul client la un modul server și conțin profilul inițial al utilizatorului, prima postare care verifică înregistrarea, și alte date necesare în procesul de înregistrare. Asemănător cu acest tip de cereri funcționează și cererile de autentificare, trimise tot de la un client spre un server, dar acestea conțin doar numele de utilizator și cheia publică a profilului, alături de o semnătură digitală necesară în procesul de autentificare.

Un alt tip de cerere este cea de căutare a unui profil de utilizator. Aceasta poate fi trimisă de la un modul client la un modul server, sau de la un server către alt server. Ea conține datele cunoscute despre un anumit profil: identificatorul unic, numele de utilizator,

cheia publică sau cheia publică de recuperare. Răspunsul așteptat va conține datele corecte ale unui profil care respectă criteriile date, garantând totodată și existența în rețea a unui astfel de cont de utilizator.

Spre deosebire de cererea de căutare, dacă se dorește doar verificarea actualizărilor unui profil de utilizator sau obținerea detaliilor opționale ale profilului, se poate trimite o cerere de informații, atât la server cât și la client. Această cerere conține numărul versiunii curente în nodul emițător pentru fiecare grup de câmpuri al profilului căutat, și va primi drept răspuns doar acele grupuri pentru care există versiuni mai noi.

Există două tipuri de cereri care servesc la stabilirea noilor conexiuni: recomandarea unui nod și cererea de stabilire a conexiunilor. Acestea au ca destinație un modul client, respectiv un modul client sau server care servește drept intermediar în stabilirea conexiunii, proces explicat în capitolul 3.3.4.

Un alt tip de cerere este cea de postări, care include un interval de identificatori unici ai unor postări și identificatorul profilului de utilizator care le-a realizat. Este trimisă de client atât către alte module client, dar la nevoie și către module server, iar răspunsul trebuie să conțină postările cerute, alături de toate elementele individuale de securitate. Și modulele server pot trimite astfel de cereri direct către alte module server.

Un ultim tip de mesaj trimis este notificarea unei actualizări de profil, care conține atât postarea nouă cât și profilul de utilizator al persoanei care a realizat-o, acesta cuprinzând doar detaliile de identificare, noul număr de postări și datele opționale care au fost modificate. Acest tip de mesaj este folosit și pentru semnalarea unei postări noi de conținut, și aceasta presupunând o actualizare de profil, mai exact modificarea câmpului numărului de postări. Așadar, datorită implicării actualizării profilului la fiecare postare nouă, nu există un tip separat de notificare pentru conținut nou publicat. Mesajul de actualizare a profilului este trimis de modulul client către toate nodurile conectate direct, inclusiv către modulul server care îl retrimite către toate celelalte module server, garantând astfel că informația va fi disponibilă de oriunde din rețea.

Răspunsurile primite în urma diverselor solicitări au atașat identificatorul solicitării, o anumită stare de succes sau de eroare, iar în cazul în care cererea a reușit, va fi atașat în funcție de nevoie și conținutul solicitat.

Toate nodurile din rețea primesc astfel de cereri de la oricare din modulele conectate direct, le analizează și la nevoie transmit răspunsuri sau contactează și alte noduri. Astfel, cererile sunt așteptate și tratate pe toată durata funcționării nodului în cadrul rețelei și pot fi transmise pe oricare din canalele de comunicare ale rețelei.

3.3.3. Înregistrarea și autentificarea în rețea

Procesul de înregistrare are rolul de a crea un profil nou de utilizator pentru orice persoană care dorește să folosească rețeaua de socializare pentru prima dată. După acest pas, la fiecare refolosire a aplicației se trece prin procesul de autentificare. Ambele necesită comunicarea modulului client cu un modul server.

Datele care permit accesul la un cont sunt numele de utilizator și parola, similar cu datele cerute de un site web obișnuit. Diferența vizibilă pentru utilizator, dată de modul de securizare al mesajelor în arhitectura peer-to-peer, se rezumă la primirea după înregistrare a cheii private asociate cheii de recuperare a profilului.

Autentificarea în rețea implică următorii pași:

1. Utilizatorul accesează aplicația și introduce numele de utilizator și parola,
2. Modulul client stabilește o conexiune WebSocket cu un modul server din rețea, și primește automat de la acesta un text generat aleatoriu,
3. Pe baza numelui de utilizator și a parolei introduse, modulul client generează o pereche de chei RSA și o folosește pe cea privată pentru a semna digital textul primit de la server,
4. Se trimite către server numele de utilizator, cheia publică și semnătura digitală generată,
5. Serverul verifică semnătura digitală și caută profilul cu numele de utilizator și cheia publică dată, trimițând cereri de căutare la toate celelalte module server din rețea pentru a se asigura că are disponibilă cea mai actualizată versiune a profilului,
6. Dacă toate verificările reușesc, modulul client primește un răspuns de succes alături de profilul de utilizator găsit de server,
7. Modulul client salvează profilul primit și cheia privată generată anterior pentru a putea semna digital și eventuale noi postări, și încheie procesul de autentificare prezentând clientului pagina profilului personal.

Dacă se dorește înregistrarea unui cont nou, procesul funcționează similar cu cel de autentificare, dar intervin niște pași suplimentari. După pasul 3, modulul client generează și un profil de utilizator nou cu numele de utilizator și cheia publică rezultată. De asemenea, este generată aleatoriu, folosind funcții criptografice sigure, o nouă pereche de chei RSA, partea publică a acesteia devenind cheia publică de recuperare a profilului. Componenta

privată a acesteia este folosită pentru a genera încă o semnătură digitală pentru textul aleatoriu primit de la server. Identificatorul unic al profilului este generat și el aleatoriu.

Noul profil și o postare inițială de tip actualizare de profil, alături de ambele semnături digitale realizate, sunt trimise către server. Acesta caută din nou profilul după datele de identificare, de data aceasta căutând să respingă înregistrarea dacă se găsește un profil deja existent cu acestea.

Dacă nu se găsește un alt profil cu aceleași date de identificare și toate validările efectuate de server se încheie cu succes, modulul client este notificat similar despre succesul înregistrării, iar serverul transmite către celelalte servere din rețea noul profil și prima postare printr-o notificare de tip actualizare de profil.

Ca ultimă etapă, modulul client afișează pe ecranul utilizatorului cheia privată corespunzătoare cheii publice de recuperare generate, cu scopul ca persoana care a creat contul să o salveze într-un format sigur, la alegerea sa.

Aceste procese de autentificare și înregistrare asigură modulul server contactat de corectitudinea datelor de autentificare introduse prin generarea cheii publice și prin folosirea ei la realizarea unei semnături digitale.

Din motive de securitate, cheia privată generată cu ajutorul datelor introduse nu este păstrată în memoria browserului, deci procesul de autentificare trebuie repetat la fiecare redeschidere a aplicației. Dacă modulul server este schimbat în timpul utilizării aplicației, va fi necesară o nouă autentificare pentru acest modul, dar întregul proces va fi rulat automat de modulul client fără intervenția utilizatorului.

Există un caz particular ca rețeaua de servere să nu fi primit ultimele actualizări de la client, iar în memoria locală a acestuia să existe o versiune mai actualizată a profilului decât cea primită de la server în urma autentificării. Acest scenariu este puțin probabil, dar dacă se constată această situație modulul client va trimite o cerere de actualizare a profilului cu noile date imediat după terminarea autentificării. În cazul în care o actualizare a cheii publice nu a fost înregistrată de rețea, utilizatorul va fi nevoit să folosească vechea parolă a contului la autentificare, iar informațiile vor fi actualizate automat ulterior.

3.3.4. Stabilirea conexiunilor

În cadrul rețelei există două tipuri de conexiuni directe între două module: conexiuni WebSocket, dacă una din părțile implicate este un modul server, și conexiuni WebRTC, exclusiv pentru comunicarea între modulele client.

Conexiunile WebSocket sunt stabilite cu ușurință, singura informație necesară fiind adresa serverului care se dorește contactat. API-ul [46] pus la dispoziție de această tehnologie se ocupă de restul detaliilor pentru stabilirea canalului de comunicare.

În momentul în care un nou modul server dorește să intre în rețea, el are nevoie de adresa unui alt modul server existent. La conectarea cu acesta se va preciza faptul că emițătorul este un alt modul server, iar astfel nu va fi necesară prezentarea unor date de autentificare. În urma conectării, noul modul server primește și o listă a adreselor tuturor celorlalte module server existente în rețea, cu scopul de a stabili conexiuni directe și cu acestea.

La conectarea unui modul client la un modul server, aplicația va dispune automat de o listă de module server cunoscute în rețea. Conectarea la oricare din aceste adrese va necesita parcurgerea procesului de înregistrare sau autentificare a utilizatorului. Există posibilitatea ca modulul server ales să nu funcționeze, caz în care se va încerca alt modul. Dacă modulul ales nu mai permite conexiuni noi, acesta va recomanda utilizatorului adresa unui alt modul server disponibil din rețea.

Conectarea între două module client prin WebRTC necesită parcurgerea unui proces mai complicat, care necesită intervenția unui intermediar care să transmită mesajele între cei doi clienți. Primul pas este reprezentat de primirea unei recomandări de conectare de la modulul server conectat sau de la un alt modul client dacă s-a stabilit anterior conexiunea cu acesta.

Recomandările de conectare se bazează pe lista de utilizatori urmăriți din cadrul profilului de utilizator accesat prin fiecare modul client. Când un server sau un client detectează doi utilizatori conectați simultan care se află pe una din listele de profiluri urmărite, se va transmite spre aceștia o sugestie de stabilire a unei conexiuni directe, întrucât este probabil ca aceștia să dețină conținut necesar amândurora. Totodată, după autentificarea cu succes la un modul server, acesta încearcă să trimită automat niște recomandări de conectare, pentru ca noul modul client să se integreze cât mai rapid în rețea.

După primirea unei recomandări, clientul analizează propunerea și dacă dorește trimite o cerere de stabilire a conexiunii către nodul care a făcut recomandarea. Această cerere conține atât identificatorul profilului curent cât și al destinației, dar și o ofertă de conectare WebRTC. Nodul intermediar are doar rolul de a transmite mai departe, către destinație, această cerere. Clientul destinatar alege dacă acceptă oferta, iar în caz afirmativ formulează un răspuns care este trimis din nou printr-o cerere de stabilire a conexiunii prin

nodul intermediar. După această etapă se trece la partea de negociere a candidaților ICE, aceștia fiind transmiși de la un client la altul tot prin nodul intermediar.

Tot procesul de conectare se realizează prin formularea de cereri de stabilire a conexiunii. Acestea conțin pe lângă identificatorii profilurilor sursă și destinație și etapa procesului, care permite destinatarului să facă diferența între tipurile de conținut care pot fi atașate cererii: ofertă, răspuns sau candidat ICE. Acest conținut efectiv al cererii este întotdeauna criptat folosind cheia publică a destinației, în acest fel asigurându-se faptul că datele de conectare ale unui client nu vor fi interceptate de un nod intermediar malițios, dar și faptul că doar nodul destinație este capabil să citească oferta și să stabilească o conexiune.

Întotdeauna după stabilirea conexiunilor directe, clienții trimit între ei solicitări de actualizare a datelor din profilul celuilalt utilizator, pentru a se asigura că au disponibilă cea mai actualizată versiune a acestuia.

Un alt aspect al procesului de stabilire a conexiunilor este acela că ofertele primite de la noduri unde sunt conectați utilizatori blocați nu vor fi niciodată acceptate. Conexiunile cu modulele server nu pot fi blocate explicit de utilizator, dar vor fi automat înlocuite de modulul client dacă serverul nu răspunde în mod corespunzător.

3.3.5. Tratarea solicitărilor între modulele client

În rețeaua peer-to-peer, modulele client sunt capabile să primească și să trateze solicitări fără a necesita intervenția vreunui modul server.

După cum am menționat în capitolul trecut, clienții conectați direct transmit și tratează cereri de actualizare a informațiilor profilurilor, dacă sunt conectați direct cu utilizatorul pentru care se caută actualizările.

Cererile de postări pot fi și ele tratate de alte module client. Pentru a solicita postări, clientul trebuie să obțină mai întâi cea mai actualizată variantă a profilului căutat. După acest pas, modulul client analizează dacă există vreo conexiune directă cu acel utilizator sau cu un alt modul care ar putea avea acces la acele postări și formulează o cerere direct către aceștia. La primirea unei astfel de cerere, modulul client destinatar caută în propriul spațiu de stocare existența postărilor căutate, iar în caz afirmativ le trimite solicitantului printr-un răspuns de succes.

Modulele client conectate direct își transmit notificări în cazul publicării unei postări noi de către utilizator. În acest fel, toate modulele conectate direct primesc automat cea mai

actualizată variantă a profilului utilizatorului, precum și noua postare, fără a fi nevoie de intervenția serverelor.

Încă o capacitate a modulelor client este aceea de a juca rol de nod intermediar în procesul de stabilire a conexiunii, având capacitatea de a formula recomandări de conectare și de a redirecționa cererile de stabilire a conexiunilor.

3.3.6. Tratarea solicitărilor de către modulele server

Modulele server prezintă un mecanism mai complex de tratare a cererilor, deoarece există unele solicitări tratate exclusiv de acestea, dar există și situații în care modulele client nu pot găsi informațiile căutate la nodurile vecine și sunt nevoite să apeleze la server.

În categoria cererilor tratate exclusiv de server intră cererile de căutare a unui profil de utilizator. Această căutare urmărește găsirea celei mai actualizate versiuni a profilului de utilizator, deci trebuie căutate obligatoriu actualizări pe toate serverele din rețea, motivul fiind că unul dintre ele trebuie să fi primit o notificare de la acel utilizator în momentul ultimei actualizări. Așadar, la primirea de la un client a unei astfel de cereri, serverul va căuta în memoria locală, dar va și transmite mai departe cereri de căutare sau, după caz, de actualizare a informațiilor către celelalte module server. Singurul caz în care cererea nu necesită redirecționare este atunci când utilizatorul căutat este conectat direct la modulul server respectiv, acesta având deja garanția celei mai actualizate versiuni a profilului. În cazul în care cererea este primită de la alt modul server ea nu este redirecționată mai departe, iar răspunsul va conține datele corespunzătoare disponibile în memoria locală.

Datorită faptului că necesită trimiterea unor cereri de căutare, și cererile de autentificare și de înregistrare pot fi tratate doar de către module server. Modul de soluționare al acestor cereri a fost descris pe larg în capitolul 3.3.3.

Cererile de actualizare a informațiilor unui profil și cele de postări funcționează similar cu cererile de căutare, prin interogarea la nevoie a tuturor serverelor din rețea.

În momentul în care un modul server primește notificarea realizării unei noi postări de către un modul client conectat, el salvează noile date local și notifică automat toate celelalte module server din rețea, făcându-le și pe acestea să își actualizeze datele locale.

La fel ca modulul client, și serverul joacă rol de nod intermediar în procesul de stabilire a conexiunii între doi clienți.

3.4. Persistența datelor

Arhitectura distribuită a rețelei de socializare a făcut necesară introducerea mai multor mecanisme care să asigure persistența profilurilor și a postărilor. Atât modulele server cât și modulele client salvează date pentru a asigura că acestea vor rămâne ușor accesibile și disponibile pe termen lung.

3.4.1. Date salvate în modulele client

Modulele client pot stoca pe dispozitivul utilizatorului atât profiluri de utilizator cât și postări. Principalul rol al acestui spațiu de stocare este ușurarea accesibilității datelor: profilurile și postările căutate pot fi disponibile direct pe client, minimizând astfel volumul de date solicitate și transferate din rețea. De asemenea, după stabilirea unei conexiuni directe cu un modul client, datele salvate în memoria acestuia pot fi solicitate de alte noduri ale rețelei prin mecanismele descrise în capitolele anterioare, scăzând astfel numărul de solicitări trimise către modulele server.

Întrucât modulele client rulează în browserul web, s-a folosit un API comun pus la dispoziție de acestea, mai exact IndexedDB [45]. Acesta a permis crearea unei baze de date locale, în cadrul căreia au fost definite două tabele, unul pentru profiluri și unul pentru postări. Pentru a eficientiza căutarea datelor în aceste tabele, au fost definiți indecși pentru elementele de identificare ale acestor entități. Un alt motiv pentru care a fost ales acest API a fost dat de faptul că acesta oferă cel mai mult spațiu de stocare ce poate fi oferit de browser.

În practică, modulul client salvează toate datele pe care utilizatorul le accesează. Adică, la vizitarea profilului unui utilizator vor fi salvate toate informațiile acelui profil precum și toate postările văzute de utilizator. Așadar, aceste date vor fi disponibile și la utilizările ulterioare ale aplicației. Deoarece utilizatorii rețelelor de socializare au tendința de a revizita anumite profiluri, datele vor fi disponibile în aceste cazuri direct din memoria locală a dispozitivului folosit.

Mai mult, profilurile disponibile pe modulele client în cazul cărora s-a realizat o conexiune directă în rețea vor fi și ele salvate, iar notificările primite de la acestea vor actualiza în fundal baza de date.

Pentru a optimiza folosirea spațiului de stocare, a fost implementat un mecanism de prioritate a datelor salvate. Modulul client va salva întotdeauna cu prioritate maximă propriul profil de utilizator și propriile postări. Următoarea treaptă pe scara priorităților este rezervată

datelor persoanelor pe care utilizatorul le urmărește. Restul spațiului disponibil va fi ocupat cu datele persoanelor cu care nu s-a stabilit nicio conexiune, dar profilurile lor au fost totuși accesate la un moment dat de utilizator. În cazul persoanelor blocate, ele nu vor beneficia deloc de stocarea datelor pe dispozitiv, iar blocarea lor va duce la ștergerea informațiilor acestora salvate local. Ștergerea din baza de date a unui profil duce la ștergerea automată și a postărilor acestuia.

Funcționarea acestui mecanism de salvare și accesare a datelor se bazează pe probabilitatea ca un utilizator să folosească pentru o perioadă relativ lungă aplicația rețelei de socializare, dar și pe așteptarea ca acesta să stabilească un număr mare de conexiuni cu alte persoane care folosesc aplicația. Astfel, timpul în care datele sunt disponibile este suficient de lung încât alte module client să aibă șansa să realizeze conexiuni directe cu utilizatorul și să solicite o parte din datele găzduite de acesta.

3.4.2. Date salvate în modulele server

Ca și în cazul modulelor client, și modulele server salvează atât profiluri cât și postări. Principalul rol al stocării de pe server este de a asigura că există cel puțin un nod în rețea care să stocheze orice informație căutată, dar și de a garanta distribuirea celor mai actualizate astfel de informații.

Deoarece modulele server nu sunt limitate la funcționarea în browserul web și pot avea la dispoziție mai multe resurse, s-a ales folosirea unei baze de date externă. Similar cu modulul client, în cadrul acesteia sunt create două tabeluri, unul pentru profiluri și unul pentru postări.

Modulele server vor salva în baza lor de date toate informațiile primite din rețea, atât de la alte servere cât și de la clienți. Aceste date sunt actualizate automat în momentul în care se găsesc versiuni mai noi în rețea. Întrucât modulele client aleg aleatoriu serverul la care se conectează, iar modulele server nu sunt conectate în permanență la rețea, vor exista servere care nu vor primi spre salvare anumite profiluri și postări. De asemenea, vor fi cazuri când anumite profiluri nu vor fi actualizate pe toate serverele rețelei. Din această cauză, solicitările care necesită găsirea celor mai actualizate date, precum căutarea profilurilor, vor implica intervenția tuturor modulelor server conectate.

În momentul în care baza de date administrată de server ajunge să conțină un număr mult prea mare de date, spațiul va fi eliberat eliminând datele opționale ale profilurilor care nu au fost actualizate de cel mai mult timp, alături de toate postările asociate lor. Folosind

această abordare, profilurile abandonate vor fi înlăturate, iar dimensiunea elementelor rămase în memorie va fi minimă. Totodată, păstrând datele de identificare se face posibilă reactivarea profilurilor chiar dacă au fost șterse de toate modulele server, în cazul în care utilizatorul se autentifică din nou în rețea folosind un dispozitiv în memoria căruia se mai găsesc toate datele asociate vechiului profil. De asemenea, se previne înregistrarea unor conturi de utilizator care să refolosească datele de identificare ale profilurilor abandonate.

Deși în cadrul modulelor server sunt disponibile toate datele din rețea, se preferă ca modulele client să încerce să obțină informații de la nodurile vecine prin conexiunile directe realizate. În acest fel se încurajează un mod de funcționare descentralizat al rețelei și se evită congestionarea modulelor server, rămânând în sarcina lor doar solicitările modulelor client care nu au putut obține informațiile pe nicio altă cale.

3.5. Beneficii și limitări ale arhitecturii utilizate

În urma implementării unei arhitecturi distribuite care să asigure funcționarea rețelei de socializare, au fost identificate atât îmbunătățiri ale performanței cât și ale calității serviciilor oferite utilizatorilor. Principala consecință a folosirii unei rețele peer-to-peer a fost eliminarea necesității unui server central foarte performant, care să asigure în totalitate serviciile disponibile prin aplicație.

În ceea ce privește performanța, aplicația distribuită realizată preia datele necesare funcționării de la alte noduri ale rețelei sau chiar direct din spațiul de stocare local, fapt care duce la scăderea timpului de răspuns pentru multe din cereri. De asemenea, spre deosebire de o arhitectură clasică de tip client-server, introducerea mai multor clienți în rețea va duce la o disponibilitate mai mare a datelor și la posibilitatea realizării mai multor conexiuni directe. Astfel, creșterea dimensiunilor rețelei peer-to-peer va produce în medie o scădere a timpului de accesare al resurselor.

Din punct de vedere al serviciilor oferite, arhitectura folosită pune la dispoziția utilizatorilor cele mai comune funcționalități ale unei rețele de socializare, lăsând loc pentru dezvoltarea acestora și introducerea unor noi facilități. De asemenea, eliminarea unui organism central cu puteri depline asupra rețelei duce la imposibilitatea blocării abuzive a unor idei publicate de anumite persoane, înlăturând posibilitatea practicării cenzurii. În schimb, aplicația oferă fiecărui utilizator mecanismele necesare pentru a bloca anumiți

utilizatori din proprie inițiativă, în cazul în care nu doresc să vadă conținutul distribuit de aceștia.

Încă un beneficiu al arhitecturii folosite este dat de eliminarea nevoii de întreținere și administrare a unui server central foarte performant. În schimb, aplicația se bazează pe spațiul de stocare pus la dispoziție de utilizatori în cadrul dispozitivelor proprii și pe persoane care să găzduiască voluntar modulele server necesare.

În ceea ce privește limitările identificate, acestea sunt cauzate în cea mai mare măsură de nevoia de a introduce în rețea modulele server. Din cauza faptului că unele cereri pot fi tratate doar de aceste module, în cazul în care numărul de clienți raportat la numărul de servere este prea mare s-ar putea produce îngreunări ale serviciilor sau chiar blocaje. De asemenea, descentralizarea modulelor server duce la o utilizare inefficientă a spațiului de stocare, întrucât aceleași profiluri și postări vor exista simultan pe servere diferite. Din moment ce rețeaua nu are nicio asigurare în ceea ce privește disponibilitatea acestor module, stocarea datelor în locații multiple este inevitabilă pentru a asigura accesibilitatea permanentă a informațiilor. Din această cauză, orice conținut multimedia de dimensiune mare distribuit ar trebui să fie găzduit într-o sursă externă rețelei.

Tot din cauza descentralizării serverelor, utilizatorii nu pot avea nicio garanție în ceea ce privește siguranța datelor distribuite în rețea. Orice atacator ar putea găzdui un modul server, în baza lui de date fiind salvate toate profilurile de utilizator și postările publicate. În acest fel se creează o modalitate de interceptare a postărilor chiar și de către utilizatorii blocați. Așadar, arhitectura prezentată permite exclusiv distribuirea conținutului cu destinație publică, accesibil nerestricționat, iar mecanismul de blocare al unor utilizatori se rezumă la stoparea primirii conținutului din partea acestora.

O altă limitare este dată de overhead-ul produs de sistemul de semnături digitale, necesar pentru autentificarea datelor din rețea. Fiecare resursă trimisă în rețea trebuie să fie însoțită de una sau mai multe semnături digitale, iar nodurile care primesc aceste resurse sunt nevoite să verifice de fiecare dată validitatea acestora.

4. EccentricPeer – rețea de socializare descentralizată

4.1. Funcționalități și diagrama cazurilor de utilizare

Prin intermediul aplicației dezvoltate sunt oferite utilizatorului funcționalități comune în cadrul rețelelor de socializare online actuale, respectând cerințele și conceptele de bază ale unei astfel de platforme după cum au fost discutate în capitolul 2.3. Aceste funcționalități au fost împărțite în trei categorii:

- A. Funcționalități necesare administrării contului de utilizator
 - înregistrarea unui cont nou
 - autentificarea folosind un cont existent
 - schimbarea parolei de autentificare
 - ștergerea contului
- B. Funcționalități specifice gestionării conținutului propriu
 - crearea unei postări noi
 - ștergerea unei postări existente
 - modificarea atributelor profilului personal
- C. Funcționalități care permit accesarea conținutului distribuit de alte persoane
 - vizualizarea profilurilor altor utilizatori
 - căutarea altor utilizatori
 - vizualizarea listei de persoane urmărite de către alți utilizatori
 - urmărirea unor utilizatori
 - blocarea unor utilizatori

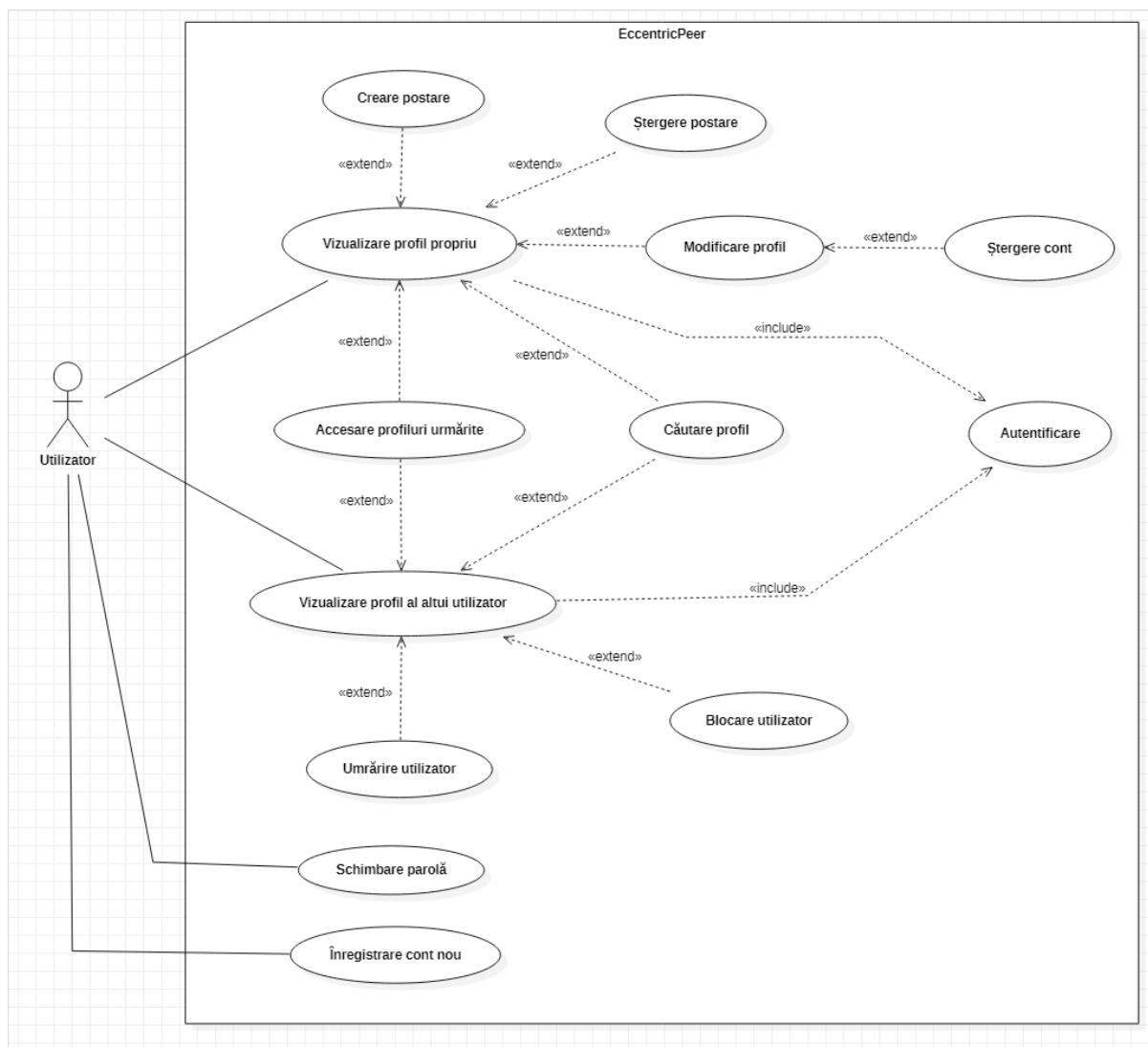


Fig. 3: Diagrama cazurilor de utilizare

4.2. Model conceptual

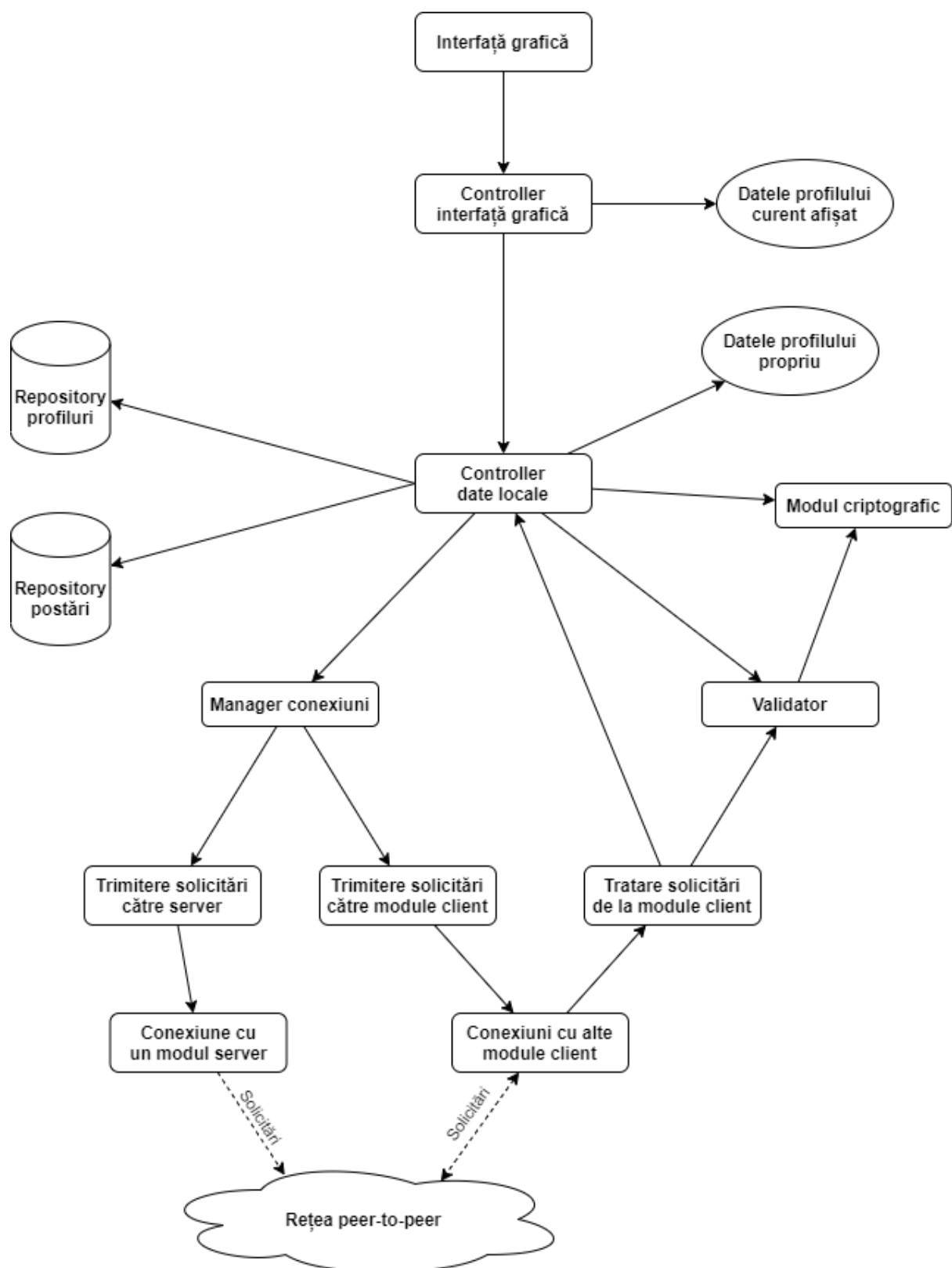


Fig. 4: Principalele interacțiuni între componentele modulului client

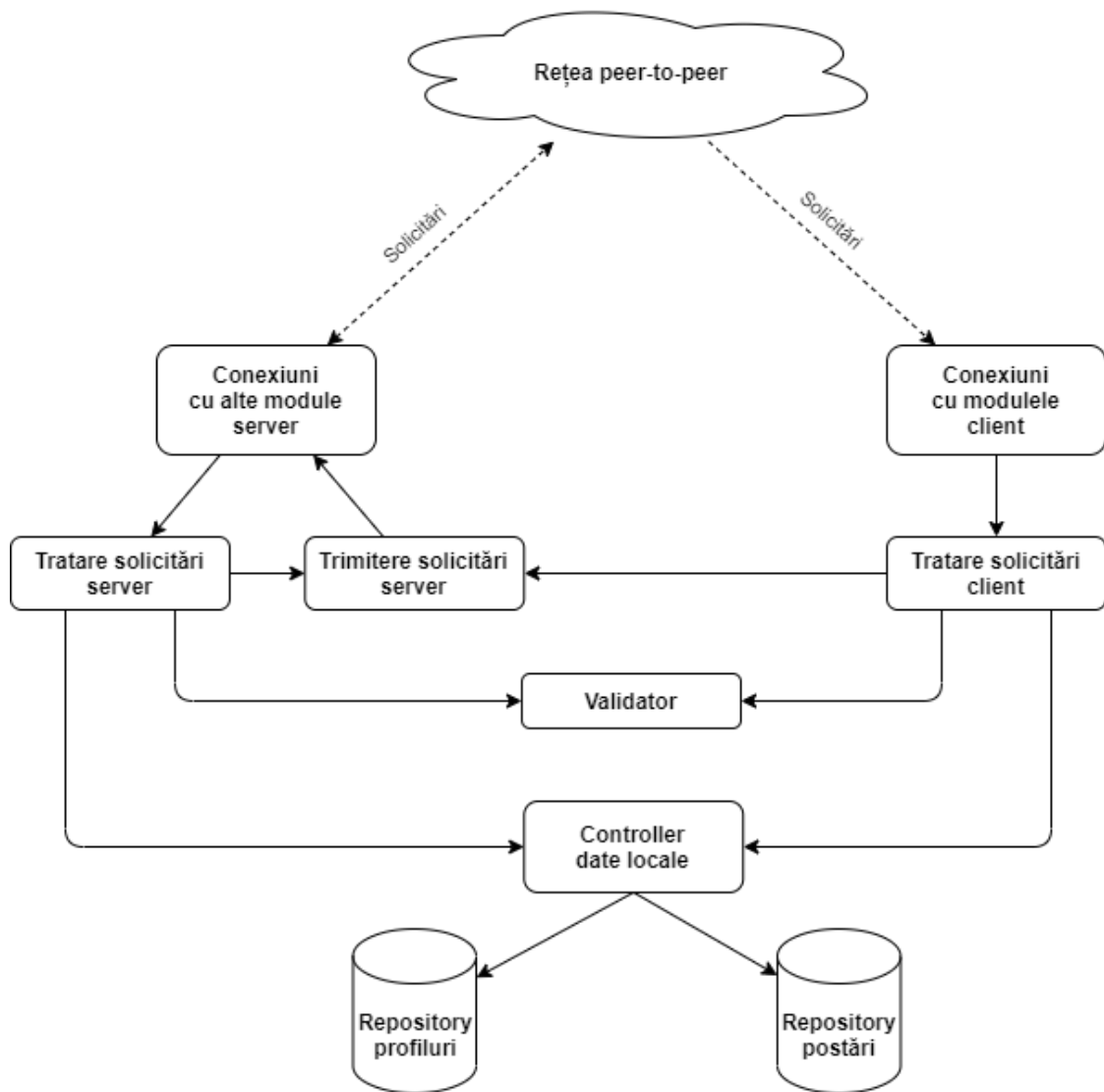


Fig. 5: Principalele interacțiuni între componentele modulului server

4.3. Tehnologii folosite

Modulul client a fost dezvoltat sub forma unei aplicații web cu o singură pagină (Single Page Application) folosind framework-ul Angular [7] și având o arhitectură de tip Application Shell [5]. În ceea ce privește limbajul de programare, pentru componentele logice s-a folosit TypeScript [19], iar structura interfeței grafice a fost realizată în HTML [44]. Elementele de stilizare au fost realizate în CSS, folosind și framework-ul Bootstrap [10]. Pentru salvarea datelor în acest modul s-a folosit API-ul IndexedDB. Conexiunile directe

între modulele client s-au realizat folosind WebRTC [48], iar conexiunea cu modulul server se stabilește prin WebSocket [46] cu ajutorul librăriei Socket.IO-client [40].

Modulul server este implementat în limbajul JavaScript [18] și folosește Node.js [31] ca mediu de rulare. Datele sunt salvate într-o bază de date MySQL folosind un driver pentru Node.js [30]. La partea de conexiuni, modulul server stabilește doar conexiuni WebSocket cu ajutorul a două librării: Socket.IO și Socket.IO-client [40]. Ambele sunt necesare pentru a permite așteptarea solicitărilor de conectare de la alte module server sau client, respectiv pentru a permite solicitarea conectării la un alt modul server.

Ambele module generează aleatoriu identificatori unici pentru cererile transmise în rețea folosind librăria uuid [41]. De asemenea, funcțiile criptografice folosite de ambele module sunt oferite de librăria cryptico-js [12].

4.4. Mockups

Primul lucru vizibil odată cu lansarea aplicației este pagina de autentificare.

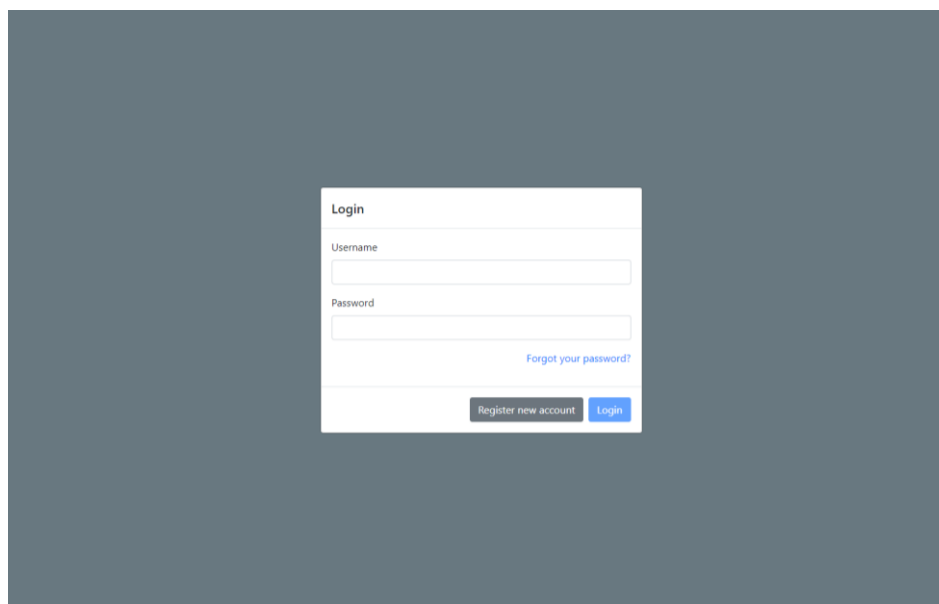


Fig. 6: Pagina de autentificare

Prin intermediul acestei pagini, utilizatorul se poate autentifica folosind un nume de utilizator și o parolă înregistrate anterior, poate opta pentru înregistrarea unui cont nou sau

poate naviga către pagina de schimbare a parolei unui cont existent, în cazul în care a pierdut accesul la acesta:

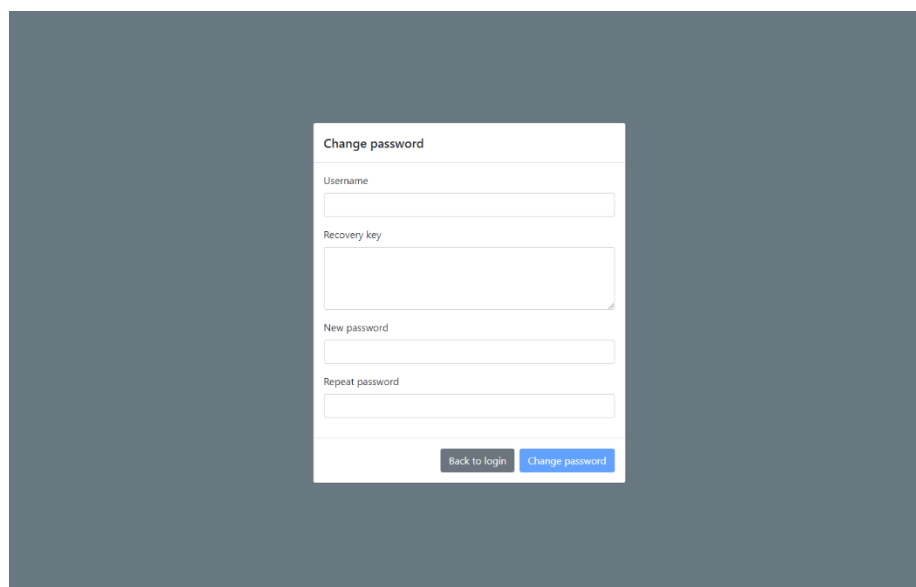
A screenshot of a web form titled "Change password". The form is centered on a dark gray background. It contains five input fields: "Username", "Recovery key", "New password", and "Repeat password". The "Recovery key" field is larger than the others. At the bottom of the form, there are two buttons: "Back to login" and "Change password".

Fig. 7: Pagina de schimbare a parolei contului

Un aspect neobișnuit al acestei pagini, introdus prin modul de autentificare folosit în arhitectura peer-to-peer dezvoltată, este nevoia introducerii cheii de recuperare pentru a permite schimbarea parolei. Această cheie este primită imediat după înregistrarea contului și este afișată pe ecranul principal pe toată durata sesiunii de utilizare a aplicației, timp în care utilizatorul trebuie să o salveze. Odată cu încheierea acestei sesiuni, cheia de recuperare nu mai poate fi obținută.

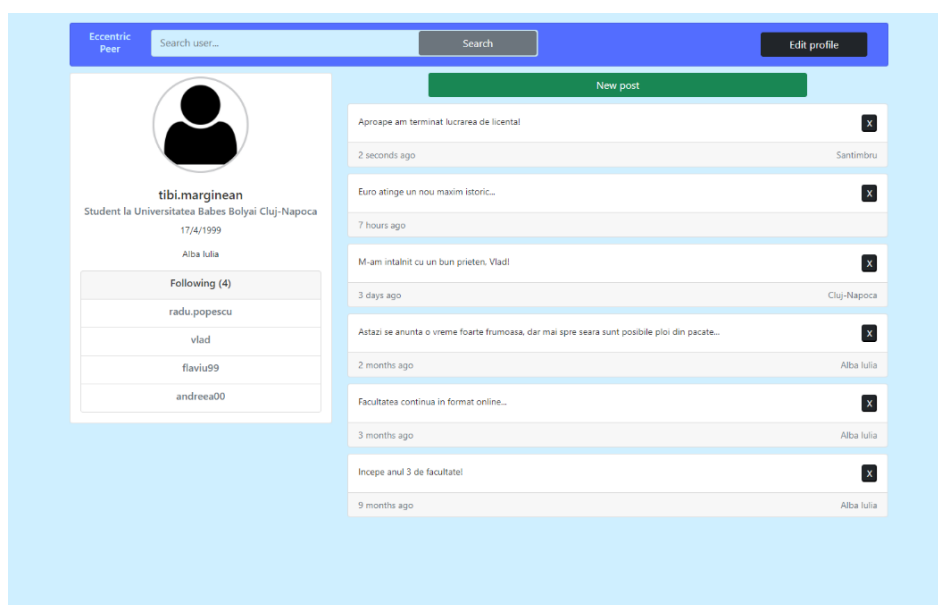


Fig. 8: Pagina profilului de utilizator, văzută de deținătorul acestuia

După finalizarea autentificării este prezentată pagina propriului profil de utilizator. În partea stângă a acesteia se găsesc informațiile personale publicate, în ordine: o fotografie de profil, numele de utilizator, o descriere scrisă personal, data nașterii, locația curentă și lista de persoane urmărite.

În partea dreaptă este vizibilă lista cu postările publicate. Acestea conțin textul mesajului, timpul trecut de la publicare, precum și locația în cazul în care utilizatorul a completat-o. În partea de sus a acestei liste se găsește un buton care permite publicarea unei noi postări. Acționarea acestuia produce următorul efect:

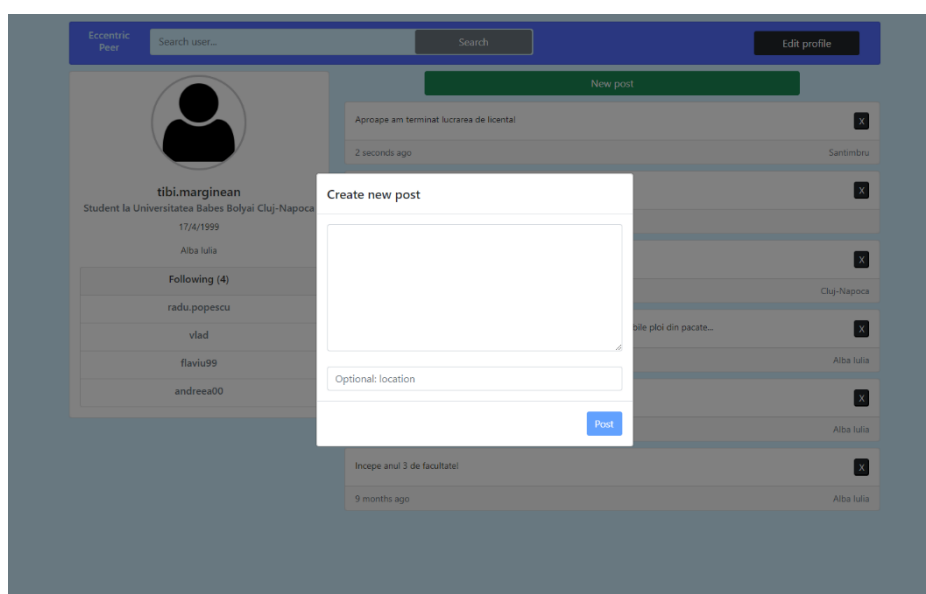


Fig. 9: Fereastra care permite publicarea unei noi postări

În bara din partea de sus a ecranului se află un câmp care permite căutarea utilizatorilor în rețea și navigarea către profilurile acestora. Această navigare poate fi realizată cu ușurință și printr-un click pe oricare element al listei de utilizatori urmăriți.

Tot în bara din partea de sus a ecranului se află un buton care permite modificarea atributelor profilului personal, vizibile în stânga ecranului. Acționarea acestui buton produce efectul din figura 10.

În cazul în care se navighează către un profil al altui utilizator, modul de afișare al acestuia este similar (figura 11), dar apar niște diferențe: dispăre butonul care permite publicarea unei postări noi, iar butonul care permite modificarea profilului este înlocuit de două butoane, unul care permite urmărirea unui utilizator și altul care permite blocarea acestuia.

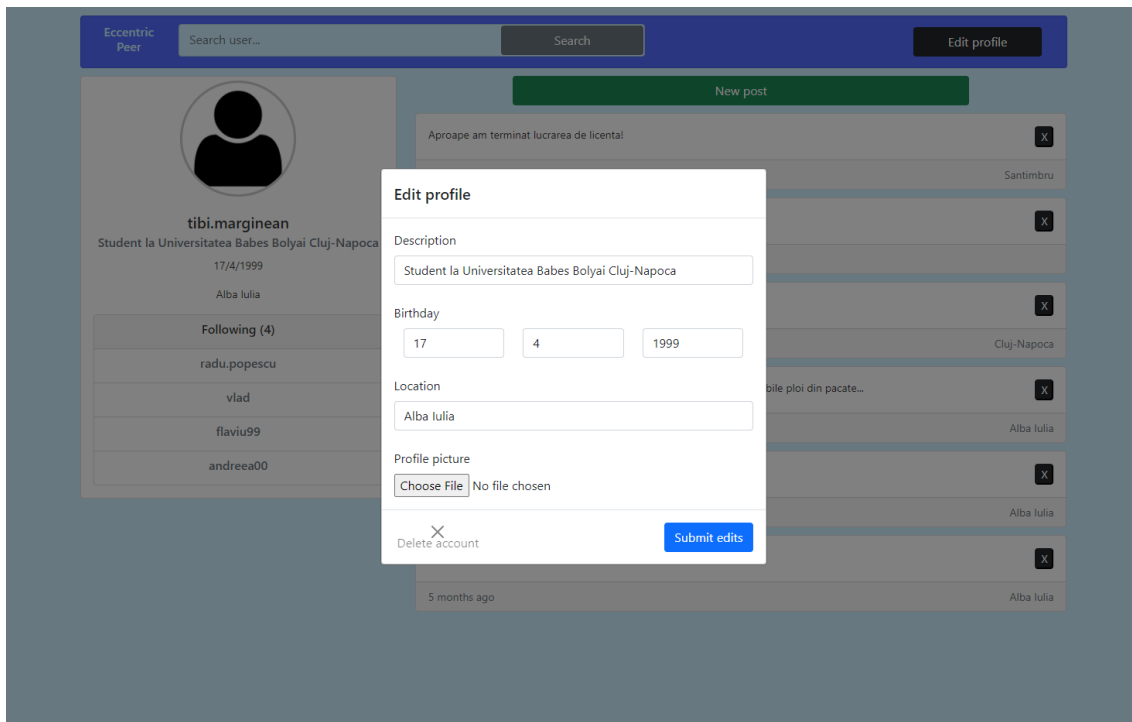


Fig. 10: Fereastra care permite modificarea atributelor profilului personal

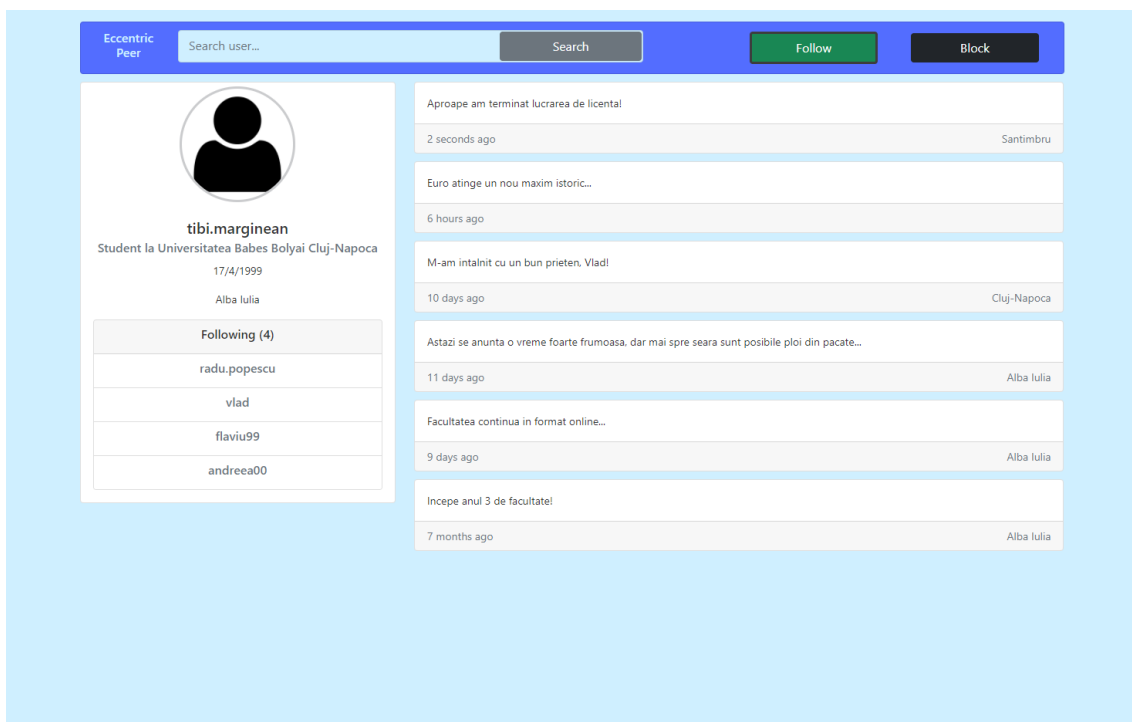


Fig. 11: Pagina profilului de utilizator, văzută de alți utilizatori

5. Concluzii

În prezenta lucrare s-au aplicat principiile descentralizării în cazul rețelelor de socializare online, fără a modifica metodele clasice de interacțiune a utilizatorilor cu acestea prin browserul web. Tehnologiile existente fac posibilă crearea unei rețele peer-to-peer prin intermediul căreia datele pot fi distribuite facil către persoanele care folosesc aplicația. De asemenea, utilizarea unor algoritmi criptografici oferă mijloacele necesare creării unor conturi de utilizator similare cu cele folosite într-o arhitectură clasică de tip client-server, și asigură autenticitatea informațiilor transmise în rețea prevenind în același timp introducerea datelor cu caracter malițios. Toate aceste aspecte pot fi încapsulate într-o simplă aplicație web care funcționează similar cu rețelele de socializare clasice din punctul de vedere al utilizatorului, punând la dispoziția acestuia facilități necesare precum un profil personal, posibilitatea distribuirii unor postări, sau stabilirea unor conexiuni cu alți utilizatori.

În schimbul punerii la dispoziție a unui spațiu de stocare pe dispozitivul personal, implementarea prezentată reușește să ofere utilizatorului o performanță bună a serviciilor oferite și să elimine prezența unui organism central care să aibă puteri depline în cadrul rețelei de socializare. Astfel a fost înlăturat orice fel de mecanism care ar putea restricționa în totalitate ideile publicate de anumite persoane, lăsând la latitudinea fiecărui utilizator să aleagă acel conținut pe care dorește să îl blocheze.

În vederea îmbunătățirii viitoare a arhitecturii propuse, principalele puncte care pot fi studiate sunt datele administrate de modulele server. Mai exact, ar putea fi creat un sistem care să determine ordinea de ștergere a profilurilor din baza de date în funcție de gradul de disponibilitate în rețea, scăzând astfel cantitatea de date salvată în noduri multiple. De asemenea, se poate studia un mecanism de securizare a datelor salvate în modulele server care să permită citirea lor doar în momentul în care ajung la modulul client.

6. Bibliografie

- [1] A. Begen, P. Kyzivat, C. Perkins și M. Handley, „SDP: Session Description Protocol,” Ianuarie 2021. [Interactiv]. Available: <https://datatracker.ietf.org/doc/html/rfc8866>.
- [2] A. Bielenberg, L. Helm, A. Gentilucci, D. Stefanescu și H. Zhang, „The growth of Diaspora - A decentralized online social network in the wild,” 2012 Proceedings IEEE INFOCOM Workshops, 2012.
- [3] A. Marin și B. Wellman, „Social Network Analysis: An Introduction,” în *The SAGE Handbook of Social Network Analysis*, SAGE, 2011.
- [4] A. N. Joinson, „Looking at, looking up or keeping up with people?: Motives and Use of Facebook,” *Proceedings of the 2008 Conference on Human Factors in Computing*, 2008.
- [5] A. Osmani și M. Gaunt, „Instant Loading Web Apps with an Application Shell Architecture,” Google Developers, 2015.
- [6] A. Russel, „Progressive Web Apps: Escaping Tabs Without Losing Our Soul,” Infrequently Noted, 2015.
- [7] „Angular,” [Interactiv]. Available: <https://angular.io/>.
- [8] B. Sredojev, D. Samardzija și D. Posarac, „WebRTC technology overview and signaling solution design and implementation,” *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2015.
- [9] B. Yang și H. Garcia-Molina, „Improving Search in Peer-to-Peer Networks,” *Proceedings 22nd International Conference on Distributed Computing Systems*, 2002.
- [10] „Bootstrap,” [Interactiv]. Available: <https://getbootstrap.com/>.
- [11] C. Lampe, N. Ellison și C. Steinfield, „A Face(book) in the Crowd: Social Searching vs. Social Browsing,” *CSCW '06*, 2006.
- [12] „cryptico-js,” [Interactiv]. Available: <https://github.com/tracker1/cryptico-js>.
- [13] D. M. Boyd și N. B. Ellison, „Social Network Sites: Definition, History, and Scholarship,” *Journal of Computer-Mediated Communication*, 2007.

- [14] D. Skvorc, M. Horvat și S. Srbljic, „Performance Evaluation of WebSocket Protocol for Implementation of Full-Duplex Web Streams,” *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2014.
- [15] E. Franchi, A. Poggi și M. Tomaiuolo, „Blogracy: a Peer-to-Peer Social Network,” *International Journal of Distributed Systems and Technologies*, 2016.
- [16] E. Milanov, „The RSA Algorithm,” 3 Iunie 2009. [Interactiv]. Available: <https://pdfdirectory.com/pdf/0702-the-rsa-algorithm.pdf>.
- [17] E. Rescorla, „WebRTC Security Architecture,” *Internet Engineering Task Force (IETF)*, 2019.
- [18] Ecma, „ECMAScript 2020 Language Specification,” Ecma International, 2020. [Interactiv]. Available: <https://262.ecma-international.org/11.0/>.
- [19] G. Bierman, M. Abadi și M. Torgersen, „Understanding TypeScript,” *European Conference on Object-Oriented Programming*, 2014.
- [20] G. Blank și B. Reisdorf, „The Participatory Web,” *Information Communication and Society*, 2012.
- [21] G. Sivek, S. Sivek, J. Wolfe și M. Zhivich, „WebTorrent: a BitTorrent Extension for High Availability Servers,” 2004.
- [22] H. Alvestrand, „Google release of WebRTC source code,” 1 Iunie 2011. [Interactiv]. Available: <https://lists.w3.org/Archives/Public/public-webrtc/2011May/0022.html>.
- [23] I. Fette și A. Melnikov, „The WebSocket Protocol,” Decembrie 2011. [Interactiv]. Available: <https://datatracker.ietf.org/doc/html/rfc6455>.
- [24] ISO/IEC 22275, *Information technology — Programming languages, their environments, and system software interfaces — ECMAScript® Specification Suite*, 2018.
- [25] J. A. Obar și S. Wildman, „Social Media Definition and the Governance Challenge: An Introduction to the Special Issue,” *SSRN Electronic Journal*, 2015.
- [26] J. Ratkiewicz, F. Menczer, S. Fortunato, A. Flammini și A. Vespignani, „Traffic in Social Media II: Modeling Bursty Popularity,” *2010 IEEE Second International Conference on Social Computing*, 2010.
- [27] M. Bellare și P. Rogaway, „Introduction to modern cryptography,” *Ucsd Cse 207*, 2005.

- [28] M. Heins, „The brave new world of social media censorship,” *Harvard Law Review*, 2013.
- [29] M. Jazayeri, „Some Trends in Web Application Development,” *Future of Software Engineering*, 2007.
- [30] „mysqljs/mysql,” [Interactiv]. Available: <https://github.com/mysqljs/mysql>.
- [31] „Node.js,” OpenJS Foundation, [Interactiv]. Available: <https://nodejs.org/en/>.
- [32] OECD, „Society at a Glance 2019: OECD Social Indicators,” OECD Publishing, 2019.
- [33] P. Lubbers și F. Greco, „HTML5 WebSocket: A Quantum Leap in Scalability for the Web,” [Interactiv]. Available: <http://www.websocket.org/quantum.html>.
- [34] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach și T. Berners-Lee, „Hypertext Transfer Protocol -- HTTP/1.1,” Iunie 1999. [Interactiv]. Available: <https://www.hjp.at/doc/rfc/rfc2616.html>.
- [35] R. Schollmeier, „A Definition of Peer-to-Peer Networking for the Classification of Peer-toPeer Architectures and Applications,” *Proceedings First International Conference on Peer-to-Peer Computing*, 2001.
- [36] S. Kemp, „Digital 2021: Global Overview Report,” 2021. [Interactiv]. Available: <https://datareportal.com/reports/digital-2021-global-overview-report>.
- [37] S. Loreto și S. P. Romano, *Real-Time Communication with WebRTC: Peer-to-Peer in the Browser*, O'Reilly Media, Inc., 2014.
- [38] S. Nakamoto, „Bitcoin: A Peer-to-Peer Electronic Cash System,” 2009. [Interactiv]. Available: <https://bitcoin.org/bitcoin.pdf>.
- [39] S. S. Tandel și A. Jamadar, „Impact of Progressive Web Apps on Web App Development,” *International Journal of Innovative Research in Science, Engineering and Technology*, 2018.
- [40] „Socket.IO,” [Interactiv]. Available: <https://socket.io/>.
- [41] „uuidjs/uuid,” [Interactiv]. Available: <https://github.com/uuidjs/uuid>.
- [42] W. Diffie și M. Hellman, „New Directions in Cryptography,” *IEEE Transactions on Information Theory*, 1976.
- [43] W3C Technical Architecture Group, „Architecture of the World Wide Web, Volume

- One,” W3C, 15 Decembrie 2004. [Interactiv]. Available: <https://www.w3.org/TR/webarch/>.
- [44] W3C, „HTML 5,” Iunie 2008. [Interactiv]. Available: <https://www.w3.org/TR/2008/WD-html5-20080610/>.
- [45] W3C, „Indexed Database API 3.0,” 11 Martie 2021. [Interactiv]. Available: <https://www.w3.org/TR/2021/WD-IndexedDB-3-20210311/>.
- [46] W3C, „The WebSocket API,” 2021. [Interactiv]. Available: <https://www.w3.org/TR/2021/NOTE-websockets-20210128/>.
- [47] W3C, „Web Storage (Second Edition),” 28 Ianuarie 2021. [Interactiv]. Available: <https://www.w3.org/TR/2021/SPSD-webstorage-20210128/>.
- [48] W3C, „WebRTC 1.0: Real-Time Communication Between Browsers,” 26 Ianuarie 2021. [Interactiv]. Available: <https://www.w3.org/TR/2021/REC-webrtc-20210126/>.
- [49] W3C, IETF, „Web Real-Time Communications (WebRTC) transforms the communications landscape; becomes a World Wide Web Consortium (W3C) Recommendation and multiple Internet Engineering Task Force (IETF) standards,” 26 Ianuarie 2021. [Interactiv]. Available: <https://www.w3.org/2021/01/pressrelease-webrtc-rec.html.en>.