



UNIVERSITATEA DE VEST DIN TIMIȘOARA
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ
PROGRAMUL DE STUDII DE LICENȚĂ: Informatică
Aplicată

LUCRARE DE LICENȚĂ

COORDONATOR:
Lect. Dr. Cira Cristian

ABSOLVENT:
Livădariu Alexandru

TIMIȘOARA
2023

UNIVERSITATEA DE VEST DIN TIMIȘOARA
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ
PROGRAMUL DE STUDII DE LICENȚĂ: Informatică
Aplicată

Aplicație web pentru gestionarea descentralizată a dosarelor de sănătate

COORDONATOR:
Lect. Dr. Cira Cristian

ABSOLVENT:
Livădariu Alexandru

TIMIȘOARA
2023

Cuprins

Rezumat	4
1 Introducere	6
1.1 Descrierea problemei	7
1.2 Abordările existente	8
1.2.1 Gestionarea dosarelor de sănătate în România	8
1.2.2 Blockchain în domeniul sănătății	9
2 Bazele blockchain-ului	10
2.1 Componentele de bază ale arhitecturii blockchain	10
2.2 Evoluția blockchain-ului	11
2.2.1 1991 - Inventarea Blockchain-ului	12
2.2.2 2008-2013 - Apariția Bitcoin	12
2.2.3 2013-2015 - Dezvoltarea Ethereum	12
2.2.4 Hyperledger	13
2.2.5 Viitor	13
2.3 Tipuri de blockchain	13
2.3.1 Blockchain Public	13
2.3.2 Blockchain Privat	14
2.3.3 Blockchain Hibrid	15
2.4 Algoritmii de consens	15
2.4.1 Problema generalilor bizantini	15
2.4.2 Mecanisme de consens	16
3 Hyperledger Fabric	18
3.1 Introducere în Hyperledger Fabric	18
3.2 Arhitectura blockchain-ului Fabric	19
4 Metodologie	22
4.1 Arhitectura aplicație	22
4.1.1 Flux-ul de lucru	23

4.2	Tehnologii	25
4.3	Aplicația	29
4.3.1	Scriptul pentru rețea	29
4.3.2	Chaincode	30
5	Rezultate	37
5.1	Analiza performanței aplicației	37
5.1.1	Performanța aplicației pentru un utilizator	38
5.1.2	Performanța aplicației pentru 50 de utilizatori	38
5.2	Costul Aplicației	39
5.3	Interfața web a aplicației	41
5.3.1	Panoul de administrare	41
5.3.2	Pagina de Istoric pentru pacienți	42
5.3.3	Pagina de înregistrare	42
5.4	Impactul și beneficiile aplicației	43
5.4.1	Beneficiile aplicației bazate pe Hyperledger Fabric	43
5.4.2	Comparație cu o aplicație Ethereum	43
5.5	Dezavantaje ale aplicației bazate pe Hyperledger Fabric	44
5.6	Îmbunătățiri și direcții viitoare	46
	Concluzii	48
	Bibliography	50

Rezumat

Această lucrare explorează potențialul blockchain-ului în gestionarea descentralizată a dosarelor de sănătate . Soluția actuală în domeniul sănătății pentru stocarea și partajarea datelor medicale este dosarul electronic de sănătate (DES). Majoritatea partajării dosarelor se face în continuare prin email sau poșta , din cauza lipsei de mecanisme de încredere de partajare a dosarelor de sănătate. Asta duce la întârzieri semnificative în tratamentul pacientului și la costuri excesive în timp și bani pentru transportul informației din aceste dosare.

Blockchain-ul prezintă un potențial semnificativ de îmbunătățire a domeniului sănătății prin amplasarea pacientului în centrul sistemului și prin creșterea protecției datelor de sănătate și a interoperabilității. Acesta este un framework descentralizat care asigură integritatea datelor prin intermediul garanției criptografice și oferă securitate, confidențialitate și contracte inteligente pentru accesul la date.

Hyperledger Fabric, un framework open-source pentru dezvoltarea de aplicații blockchain, îndeplinește toate aceste cerințe, oferind un mediu sigur și distribuit pentru sistemele de sănătate. În industria medicală există numeroase domenii în care Hyperledger Fabric poate fi adoptat, dar această lucrare se concentrează pe managementul dosarelor medicale.

Capitolul 1

Introducere

Satoshi Nakamoto¹ este considerat inventatorul tehnologiei blockchain datorită publicării lucrării sale despre Bitcoin în 2008 sub numele de „Bitcoin: A Peer-to-Peer Electronic Cash System”. [Nak08] Această lucrare a descris un sistem de plată electronică bazat pe conceptul de criptografie și a oferit o soluție la problema cheltuielilor duble în care o monedă digitală nu poate fi duplicată și nimeni nu o poate cheltui de mai multe ori. Mai exact, Nakamoto a introdus conceptul de registru distribuit, în care istoricul tranzacțiilor cu monede electronice poate fi urmărit și confirmat pentru a preveni problema dublei cheltuieli. Programul open source pentru implementarea sistemului Bitcoin a fost lansat doar câteva luni mai târziu, iar prima rețea Bitcoin a fost începută la începutul anului 2009, când Satoshi Nakamoto a creat primii Bitcoin. În ciuda faptului că identitatea inventatorului Bitcoin rămâne anonimă, criptomonedele continuă să fie create și comercializate, cu o comunitate puternică pentru a sprijini și aborda diverse probleme legate de cod. Gestionarea dosarelor medicale ale pacientului este întotdeauna o sarcină dificilă, deoarece sunt date extrem de sensibile care necesită o manipulare atentă. Dosarul electronic de sănătate (DES) facilitează urmărirea istoricului medical al pacientului, oferind mai mult informații medicilor, deoarece îi ajută să exploreze bazele de date privind sănătatea pacienților pentru a lua o decizie adecvată în oferirea celui mai bun tratament, dar vine cu propriile sale probleme. Atunci când vine vorba de DES, trebuie acordată o atenție suplimentară securității, accesibilității ușoare și Veridicității. De exemplu, ori de câte ori ne gândim la informații, în special la datele noastre personale stocate online sau transferate online, la început este o mare îngrijorare cu privire la manipularea datelor, pierderea sau furtul acestora, ceea ce duce la indisponibilitatea datelor în perioada necesară. Adoptarea tehnologiei blockchain în domeniul sănătății poate transforma sistemul existent oferind mai multă fiabilitate și accesibilitate ușoară prin rețeaua sa distribuită, securitate prin utilizarea metodelor criptografice și Veridicității prin înregistrări imuabile. Pentru a îndeplini toate aceste

¹nume folosit de dezvoltatorul anonim al Bitcoin-ului

cerințe, un blockchain trebuie să aibă funcțiile menționate anterior, împreună cu o rețea de nivel de întreprindere pentru a restricționa accesul public fără autorizație. Hyperledger Fabric îndeplinește acest lucru oferind exact același tip de blockchain. Hyperledger Fabric oferă o rețea de blockchain de nivel de întreprindere care utilizează conceptul de contracte inteligente pentru a efectua tranzacțiile în rețea. Acest lucru ar putea fi o soluție excelentă în abordarea problemelor ridicate de sistemele tradiționale de baze de date în domeniul sănătății, împreună cu problemele implicate în domeniul sănătății în sine. În [FA16], este dată o definiție formală a orașelor inteligente: „Un oraș inteligent este un sistem care îmbunătățește capitalul uman și social în mod înțelept utilizarea și interacțiunea cu resursele naturale și economice prin soluții și inovații bazate pe tehnologie pentru a se adresa publicului probleme și realizarea eficientă a dezvoltării durabile și calitate înaltă a vieții”. Factorul cheie al orașelor inteligente este informația și comunicarea. [XTH⁺19]

1.1 Descrierea problemei

Deși blockchain-ul este folosit în principal pentru criptomonede, precum Bitcoin, capacitățile și aplicațiile sale nu au fost încă extinse cu mult dincolo de criptomonede. În ultimii ani, urbanizarea rapidă a lumii cauzează multe probleme din punct de vedere economic, social și de mediu, probleme care afectează condițiile de viață și calitatea oamenilor viața în mod semnificativ. Conceptul de „Smart City”² oferă oportunități pentru a rezolva aceste probleme urbane. Obiectivele orașelor inteligente sunt să folosească cât mai bine resursele publice, să ofere servicii de calitate pentru cetățeni și îmbunătățirea calității oamenilor de viață. Tehnologia Informației și Comunicațiilor (TIC) joacă un rol important în implementarea orașelor inteligente. Industria sănătății este caracterizată ca o industrie tradițională care este considerabil de rigidă datorită faptelor de schimbare și rezistentă la practicile inovatoare. Problemele din domeniul sănătății (de exemplu, confidențialitatea, calitatea îngrijirii, securitatea informațiilor) au atras atenția în ultimii ani la nivel mondial. Tehnologiile blockchain au fost din ce în ce mai recunoscute ca un instrument de abordare a problemelor existente de distribuție a informațiilor. Acesta poate îmbunătăți practicile imediate de asistență medicală, de exemplu în îmbunătățirea furnizării serviciilor de sănătate și a calității suportului de îngrijire. Un raport recent al Market Research Future ³ (MRF) arată că tehnologia blockchain în domeniul sănătății este de așteptat să genereze peste 42 de milioane de dolari în valoare și să atingă o rată de creștere anuală compusă de 71,8

²O zonă urbană dezvoltată care creează o dezvoltare economică durabilă și dispune de o calitate înaltă a vieții excelând în mai multe domenii cheie; economie, mobilitate, mediu, oameni, viață și guvern.

³o companie de cercetare de piață și analiză a tendințelor în industrie.

% până în 2023. O astfel de creștere puternică este determinată de implementarea caracteristicilor blockchain descentralizate cu o transparență mai mare, securitate și confidențialitate îmbunătățite, transparență sporită, eficiență sporită și costuri reduse. [Sar18]

1.2 Abordările existente

Un Dosar Electronic de Sănătate(DES) este o versiune electronică a detaliilor medicale ale unui pacient. În vremurile actuale, DES este folosit pentru a partaja dosarele medicale ale pacienților în diferite spitale. DES constă în informații de sănătate ale pacientului sub forma Fișei Medicale Electronice . Fișa Medicală Electronica conține diagnosticele medicale ale pacientului, alergiile, istoricul, tratamentul și rapoarte de laborator. Cantitatea mare de date din DES poate fi folosit pentru învățarea automată și analiza datelor.

1.2.1 Gestionarea dosarelor de sănătate în România

În România, există mai multe abordări pentru gestionarea datelor de sănătate, printre care se numără: Dosarul Electronic de Sănătate (DES) - este un sistem informatic integrat la nivel național care permite accesul în timp real la informațiile medicale ale pacienților. Acesta a fost lansat în 2018 și permite centralizarea datelor medicale ale pacienților, incluzând date despre diagnostic, tratament, examene de laborator și imagistice, alergii sau alte informații medicale relevante. Sisteme de management al documentelor medicale - multe spitale și clinici utilizează astfel de sisteme pentru a stoca și gestiona documentele medicale ale pacienților. Aceste sisteme pot fi utilizate pentru a gestiona dosarele medicale ale pacienților, precum și pentru a asigura accesul la informații medicale relevante pentru medici și personalul medical. Aceste abordări sunt utile în gestionarea datelor medicale, însă prezintă unele limitări, cum ar fi riscul de securitate al datelor și dificultatea de a asigura accesul la informații medicale relevante pentru toți medicii și personalul medical implicați în tratamentul pacientului. Utilizarea tehnologiei blockchain ar putea aduce o soluție pentru aceste probleme, prin asigurarea securității datelor medicale și facilitarea accesului la informații medicale relevante pentru toți cei implicați în tratamentul pacientului. Oferirea accesului pacienților la gestiunea propriei identități permite integrarea procedurii de consimțământ informat, asigurând în același timp confidențialitatea datelor individuale de sănătate. [PM19]

1.2.2 Blockchain în domeniul sănătății

Tehnologia blockchain are puterea de a schimba domeniul sănătății și de a pune datele în mâinile pacienților. Un pas înainte în această direcție este MedRec[AEVL16], care oferă pacienților și medicilor un jurnal imutabil al înregistrărilor medicale. Acesta adoptă o abordare diferită în ceea ce privește stimularea minerilor, oferindu-le acces la date anonime despre sănătate în schimbul menținerii rețelei. MedRec utilizează contracte inteligente pentru a mape relațiile dintre pacienți și furnizorii de servicii medicale, în care contractul prezintă o listă de referințe care detaliază relațiile dintre nodurile din blockchain. De asemenea, MedRec pune relațiile și serviciile medicale în mâinile pacientului, oferindu-i posibilitatea de a accepta, respinge sau modifica relațiile cu furnizorii de servicii medicale, precum spitalele, asigurătorii și clinicile.

Blockchain oferă oportunitatea de interoperabilitate în sistemele de sănătate, având un registru descentralizat al faptelor acceptate în înregistrările medicale, la care toți furnizorii de servicii medicale au acces. Aceasta înseamnă că, deși interfețele utilizator pot fi diferite, registrele centrale vor fi identice pentru toți furnizorii. Există o provocare legată de starea actuală a înregistrărilor de sănătate între furnizori, care conțin cantități semnificative de aceleași informații sub identificatori diferiți care nu pot fi legați. Aceasta creează duplicare, iar odată cu creșterea blockchain-ului, performanța scade, iar acest nivel de replicare a datelor înregistrate ar necesita duplicarea pentru a menține un sistem rezonabil de performant cu identificatori unici și anonimi pentru identificarea pacienților în toate serviciile. Acesta reprezintă în sine o provocare de natură comercială în adoptarea unui sistem de înregistrare medicală blockchain. Este important de menționat că înregistrările de sănătate nu ar începe de la zero, deoarece ar trebui să înlocuiască sistemul existent, ceea ce creează provocări.

În plus, volumul enorm de date generate în mediile medicale, care se preconizează că va crește și mai mult, Kaiser Permanente⁴ estimează că va avea între 26 și 44 de petabiți de date despre cei 9 milioane de membri ai săi, provenind din dosarele electronice de sănătate și alte date medicale [RR14]. Volumul datelor înregistrate se va adăuga la această problemă de scalabilitate.

⁴este un consorțiu american integrat de îngrijire gestionată, cu sediul în California, Statele Unite

Capitolul 2

Bazele blockchain-ului

Blockchain este un registru distribuit descentralizat, de încredere într-o rețea de la utilizator la utilizator, care constituie o listă de blocuri ordonate cronologic. Fiecare bloc constă într-un hash¹ al unui bloc anterior și, prin urmare, formează un lanț. 2.1

Primul bloc din blockchain este blocul geneză, iar blocul dinaintea unui anumit bloc este numit blocul părinte. Un bloc din blockchain este format din două componente principale: antetul și corpul. Antetul, fiind o parte esențială a blocului, conține informații relevante precum versiunea, hash-ul blocului anterior, timpul de creare și Merkle root hash-ul. Versiunea specifică regulile de validare a blocurilor și asigură compatibilitatea cu restul sistemului. Hash-ul blocului anterior are rolul crucial de a verifica integritatea lanțului blockchain, astfel încât orice modificare a blocului anterior va afecta și antetul blocului curent. Timpul înregistrează momentul exact al creării blocului, oferind o perspectivă cronologică în cadrul lanțului de blocuri. Merkle root hash-ul este rezultatul aplicării unei funcții de hash asupra tuturor tranzacțiilor incluse în bloc și asigură că orice modificare a tranzacțiilor va afecta și valoarea acestui hash. Nonce-ul este un număr unic de 4 octeți, utilizat o singură dată în comunicarea între blocuri. Toate aceste informații combinate în antetul blocului asigură securitatea, autenticitatea și integritatea datelor stocate în cadrul blockchain-ului.[Lia20]

2.1 Componentele de bază ale arhitecturii blockchain

În cadrul rețelei blockchain, există diverse elemente cheie care contribuie la funcționarea și securitatea sa. Unul dintre aceste elemente este nodul.

Un nod reprezintă un dispozitiv, precum un computer, care se află în rețeaua blockchain și deține o copie a tuturor tranzacțiilor efectuate. Aceste noduri formează

¹este un termen matematic utilizat în special în criptografie și mai ales în criptografia cu chei publice.

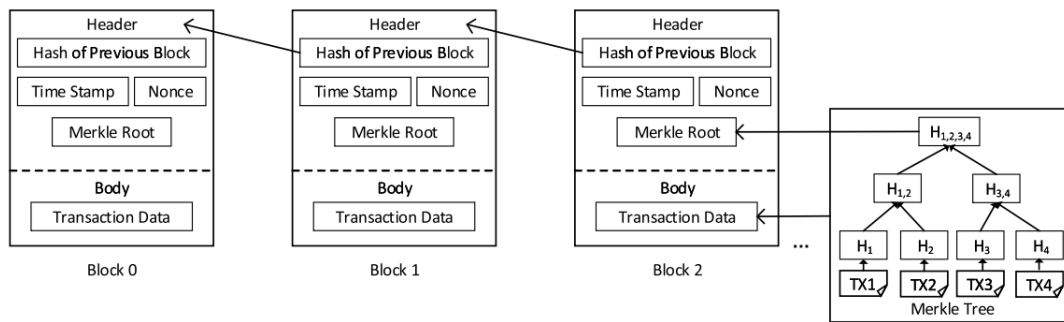


Figura 2.1: Structura unui bloc

o infrastructură distribuită care asigură redundanță și securitate.

Tranzacțiile sunt schimburile de informații între adresele din blockchain. Ele reprezintă acțiunile prin care activele digitale sunt transferate între participanții rețelei. Fiecare tranzacție este înregistrată în blocurile din blockchain.

Un bloc este o unitate de bază a blockchain-ului, utilizată pentru a grupa un set de tranzacții. Fiecare bloc conține informații despre tranzacțiile incluse în el, precum și referințe către blocurile anterioare, formând astfel o structură secvențială și cronologică numită lanț.

Minerii sunt participanții rețelei blockchain responsabili de verificarea și validarea tranzacțiilor. Ei efectuează calcule complexe pentru a rezolva algoritmi criptografici și a adăuga noi blocuri în lanțul blockchain. Aceștia asigură securitatea și imutabilitatea datelor.

Consensul reprezintă mecanismul prin care toți participanții rețelei blockchain ajung la un acord cu privire la validitatea tranzacțiilor. Acesta este realizat prin aplicarea unor reguli și algoritmi conveniți de toți colegii rețelei, garantând astfel coerența și integritatea blockchain-ului.

Prin combinarea și sincronizarea acestor elemente, rețelele blockchain asigură un mediu sigur, transparent și descentralizat pentru înregistrarea și gestionarea tranzacțiilor digitale.

2.2 Evoluția blockain-ului

Utilizarea în masă a internetului la începutul secolului a ajutat la introducerea monedei digitale ca extindere a sistemelor electronice de numerar. Următoarea parte este o cronologie a evoluției blockchain-ului.

2.2.1 1991 - Inventarea Blockchain-ului

Scott Stornetta și Stuart Haber² au inventat conceptul de Blockchain. Au creat un lanț de blocuri protejat criptografic, care ar împiedica pe oricine modificarea marcajelor de timp ale documentului [SG19]

2.2.2 2008-2013 - Apariția Bitcoin

Tehnologia blockchain a obținut o popularitate crescândă datorită apariției Bitcoin, care a fost prima aplicație bazată pe blockchain. Conceptualizat de Satoshi Nakamoto, bitcoin a reprezentat un punct de referință în evoluția acestei tehnologii, iar în anul 2009, Nakamoto a publicat o carte albă ce prezenta principiile de funcționare ale acestei monede virtuale. Acest document a reprezentat unul dintre primele eforturi de a explica modul în care blockchain-ul poate fi utilizat într-un sistem de plată descentralizat și securizat, iar inovația a atras atenția din ce în ce mai multor dezvoltatori și investitori din întreaga lume. Pe măsură ce popularitatea și utilizarea bitcoin a crescut, au fost dezvoltate și alte aplicații bazate pe tehnologia blockchain, care au început să acopere o gamă largă de domenii, de la finanțe și comerț electronic, la logistică, asigurări, sănătate și multe altele.

2.2.3 2013-2015 - Dezvoltarea Ethereum

În perioada 2013-2015, dezvoltarea tehnologiei blockchain a evoluat semnificativ odată cu conceptualizarea și dezvoltarea platformei ethereum. Această platformă a fost ideea lui Vitalik Buterin³ și a fost concepută cu suplimentarea unor caracteristici avansate precum contractele inteligente. Această inovație a fost un moment important în istoria blockchain-ului, deoarece a permis dezvoltarea aplicațiilor descentralizate și a deschis ușa pentru noi cazuri de utilizare. Ethereum este utilizat în prezent atât pentru criptomonede, cât și pentru multe alte aplicații descentralizate, în special datorită puterii sale de a utiliza contracte inteligente pentru a implementa automatizarea, securitatea și transparența.

²Scott Stornetta și Stuart Haber sunt doi cercetători și criptografi renumiți în domeniul blockchain și securitate cibernetică.

³Vitalik Buterin, este un programator ruso-canadian și fondator al Ethereum

2.2.4 Hyperledger

În 2015, Linux Foundation ⁴ a dezvoltat tehnologia Hyperledger, care permite dezvoltarea blockchain-ului open-source⁵. Hyperledger oferă un cadru de lucru modular, scalabil și sigur pentru crearea de aplicații blockchain. Există mai multe framework-uri⁶ dezvoltate în cadrul Hyperledger, fiecare având caracteristici unice și abordări diferite. Acestea includ Hyperledger Fabric, Hyperledger Iroha, Hyperledger Sawtooth și Hyperledger Burrow. Fiecare framework este conceput pentru a satisface nevoile diferite ale diferitelor aplicații blockchain și oferă o gamă largă de funcționalități, cum ar fi suportul pentru contracte inteligente, criptare puternică și mecanisme de consens sigure. Prin intermediul Hyperledger, comunitatea open-source are acces la o tehnologie blockchain robustă și inovatoare pentru a dezvolta soluții descentralizate și scalabile pentru o varietate de industrii, inclusiv domeniul sănătății. [ABB⁺18]

2.2.5 Viitor

Odată cu trecerea timpului, tehnologia blockchain a devenit tot mai populară și mai folosită de diverse companii și organizații, având un impact semnificativ asupra economiei și societății în general. În viitor, se preconizează că tehnologia blockchain va continua să se dezvolte și să evolueze, oferind noi oportunități și soluții pentru diverse industrii, inclusiv în domeniul sănătății, finanțelor, logisticii, guvernării și multe altele. Pe măsură ce tehnologia blockchain devine tot mai matură și mai avansată, se preconizează că va deveni mai scalabilă, mai securizată și mai eficientă, ceea ce va permite utilizatorilor să beneficieze de o experiență mai bună și mai sigură în ceea ce privește transferul și stocarea datelor și tranzacțiilor financiare. De asemenea, se așteaptă ca tehnologia blockchain să joace un rol important în dezvoltarea internetului lucrurilor (IoT) și a inteligenței artificiale, oferind o platformă securizată și descentralizată pentru schimbul de date și informații între dispozitive și sisteme.

2.3 Tipuri de blockchain

2.3.1 Blockchain Public

Este o tehnologie distribuită fără permisiune. După cum sugerează și numele, oricine cu acces la internet îi este permis să acceseze blockchain-ul public și să participe la

⁴Linux Foundation este o organizație non-profit fondată în anul 2000, care promovează dezvoltarea și adoptarea tehnologiei open source și sprijină comunitatea globală a utilizatorilor și dezvoltatorilor de Linux și alte proiecte open source

⁵software sau o soluție tehnologică care are codul sursă disponibil public și liber pentru a fi accesat, studiat, modificat și distribuit de către oricine

⁶o structură conceptuală sau o platformă software care oferă un set de instrumente, biblioteci și componente pentru a dezvolta și a construi aplicații, site-uri web sau alte soluții software

tranzacții. Fiecare utilizator din această rețea are propria copie a registrului. Mecanisme de consens precum Proof of Work (PoW), Proof of Stake (PoS) etc. , trebuie să ajungă la un consens în timpul adăugării de noi blocuri și în timpul verificării tranzacțiilor. Blockchain-urile publice sunt complet descentralizate, deoarece toți utilizatorii au o autoritate egală, ceea ce înseamnă că o face securizată , nimeni nu poate să aibă control deplin asupra rețelei blockchain. La rândul său, asigurarea securității datelor ajută la menținerea registrului-ului imutabil. Câteva exemple sunt Bitcoin, Litecoin, și Ethereum. Deschis și închis sunt folosite pentru a spune cine poate citi datele într-un blockchain. Aceste două proprietăți sunt folosite pentru a descrie blockchain-ul public:

Public și deschis acest blockchain este accesibil pentru toată lumea și datele pot fi citite de oricine. De exemplu, criptomonede precum Bitcoin și Ethereum.

Public și închis consumatorii pot stoca informații confidențiale și poate restricționa accesul pentru participanții selectați. În acest blockchain, toată lumea poate scrie în blockchain, dar drepturile de citire sunt deținute doar de entități selectate. De exemplu, votarea

2.3.2 Blockchain Privat

Blockchain-ul privat, cunoscut și sub denumirea de blockchain permis, se referă la o formă de blockchain în care există restricții privind accesul și contribuția la rețea. Acest tip de blockchain este utilizat în mod obișnuit de către organizații și companii în scopuri interne.

Există două variante principale de blockchain privat:

1. **Privat și deschis:** În acest tip de blockchain, fiecare colaborator din rețea are dreptul să citească datele stocate în blockchain. Aceasta poate fi utilă în cazul în care organizația dorește să ofere transparență internă și acces la anumite informații pentru toți participanții. De exemplu, o organizație poate utiliza un blockchain privat și deschis pentru a stoca declarațiile de venit corporative, permițând tuturor membrilor să aibă acces la aceste informații.
2. **Privat și închis:** În acest caz, accesul la rețeaua blockchain este restricționat doar la participanții cunoscuți și verificați. Doar acești participanți au dreptul să citească și să scrie în blockchain-ul privat. Acest tip de blockchain poate fi utilizat atunci când există nevoia de confidențialitate și securitate sporite. De exemplu, o organizație poate utiliza un blockchain privat și închis pentru a stoca declarațiile fiscale, astfel încât numai persoanele autorizate să aibă acces la aceste informații sensibile.

Un exemplu de platformă care permite implementarea blockchain-ului privat este Hyperledger Fabric. Acesta oferă un cadru flexibil și modular pentru construirea și implementarea blockchain-urilor private în cadrul organizațiilor. Prin utilizarea acestui tip de blockchain, organizațiile pot beneficia de avantajele tehnologiei blockchain, cum ar fi transparența, securitatea și imutabilitatea datelor, adaptându-le nevoilor și regulamentelor interne.

2.3.3 Blockchain Hibrid

Este un amestec de privat și public. În blockchain, participanții determină cine are acces la ce date. Unele procese sunt păstrate private, în timp ce altele sunt făcute publice. Într-un registru deschis, companiile pot proteja fondul tranzacției cu partenerii de afaceri, furnizând în continuare detaliile produsului către clienți.

2.4 Algoritmii de consens

Algoritmii de consens sunt esențiali în rețelele blockchain și sunt responsabili pentru asigurarea securității și fiabilității informațiilor înregistrate în rețea. Acești algoritmi permit participanților din rețea să ajungă la un acord comun în ceea ce privește starea rețelei și tranzacțiile efectuate în aceasta. Înainte de Bitcoin, o mulțime de sisteme valutare descentralizate au eșuat pentru că atunci când s-a ajuns la un consens, ei nu au reușit să rezolve cea mai importantă problemă, adică problema generalilor bizantini. [LSP19]

2.4.1 Problema generalilor bizantini

Problema generalilor bizantini (PGB) este o problemă matematică care analizează comunicarea între mai multe sisteme distribuite pentru a ajunge la un consens în situații în care există erori și eșecuri ale sistemelor. Această problemă este numită după "generalii bizantini" care se referă la situația în care mai mulți generali au nevoie să coordoneze un atac asupra unui oraș. Cu toate acestea, unii dintre generali pot fi trădători și pot trimite informații false. PGB are ca scop dezvoltarea unui algoritm care să permită generalilor să ajungă la un consens într-o astfel de situație, astfel încât toți generalii să ia aceeași decizie. Această problemă este importantă în cadrul sistemelor distribuite, unde informațiile trebuie să fie verificate și validate pentru a ajunge la un consens. Algoritmul propus trebuie să ia în considerare posibilitatea de a avea erori ale sistemelor, informații false sau trădători și să asigure că deciziile luate sunt corecte și că toți membrii sistemului ajung la aceeași decizie. PGB a fost introdusă pentru

prima dată în 1982 de către Leslie Lamport, Robert Shostak și Marshall Pease⁷, iar soluțiile pentru această problemă au fost studiate în detaliu în ultimii ani. Soluțiile propuse includ algoritmi bazați pe criptografie și blockchain, care utilizează un registru distribuit pentru a asigura că toți membrii ajung la aceeași decizie.

2.4.2 Mecanisme de consens

Următoarele sunt câteva mecanisme de consens care pot fi utilizate pentru a aborda Problema generalilor bizantini:

Proof of Work

Bitcoin folosește PoW ca mecanism de consens. Minerii „minează” un bloc la conectarea la blockchain rezolvând puzzle-uri criptografice. Aceasta metoda necesită o cantitate semnificativă de energie și calcul. Puzzle-urile au fost create în așa fel încât să fie provocatoare și solicitante sistemul. Când un miner completează puzzle-ul, își trimite blocul pentru verificare la rețea. Metoda de a determina dacă un bloc aparține sau nu în lanț este extrem de ușoară. Un atac de 51 la suta este un atac posibil în blockchain, în care un miner sau un grup de mineri cu mai mult de 51 la suta putere de calcul va împiedica producerea de noi blocuri și pot stabili înregistrări false ale tranzacțiilor care beneficiază atacatorii

Proof of Stake

Este o versiune mai eficientă din punct de vedere energetic a PoW. Nodurile cu cele mai multe mize (de exemplu, moneda) sunt considerate a fi mai puțin probabil să lovească rețeaua. Dezavantajul aici este monopolul, când luăm în considerare aspectul tehnologic cât și aspectele economice ale schemei ,principala parte majoritară are influență deplină și autoritate

Delegated Proof of Stake

Delegated Proof of Stake (DPoS) este un algoritm de consens folosit în rețelele blockchain pentru a verifica tranzacțiile și a decide cine validează blocurile. Acesta este utilizat în special în rețelele blockchain cu mai multe noduri, precum EOS, TRON, Solana⁸. DPoS funcționează prin alegerea unui număr restrâns de verificatori, numiți delegați,

⁷Leslie Lamport, Robert Shostak și Marshall Pease sunt trei cercetători și informaticieni renumiți care au făcut contribuții semnificative în domeniul informaticii teoretice și al sistemelor distribuite. Aceștia sunt cunoscuți în special pentru lucrarea lor clasică din 1982, intitulată ”The Byzantine Generals Problem”

⁸EOS, TRON și Solana sunt trei platforme blockchain și criptomonede importante

care sunt aleși de către comunitatea deținătorilor de monede. Acești delegați sunt responsabili pentru validarea tranzacțiilor și crearea blocurilor noi. Într-un sistem DPoS, utilizatorii deținători de monede pot vota pentru delegați și își pot revoca voturile oricând doresc. În DPoS, delegații sunt selectați în funcție de numărul de voturi pe care le primesc de la comunitatea deținătorilor de monede. Acest lucru duce la o descentralizare mai mare decât în PoW și PoS, deoarece delegații pot fi aleși din întreaga comunitate.

RAFT

RAFT este un algoritm de consens distribuit care se concentrează pe ușurința de înțeles și implementat, fiind utilizat în principal pentru sisteme cu un număr mic de noduri. Acest algoritm a fost dezvoltat de Diego Ongaro și John Ousterhout⁹ în 2014 și se concentrează pe menținerea în siguranță a datelor și a informațiilor în cazul pierderii sau deteriorării nodurilor. RAFT funcționează prin împărțirea cluster-ului într-un lider și o serie de urmăritori. Cel mai important rol al liderului este acela de a coordona deciziile din cadrul cluster-ului. Următorii așteaptă să primească instrucțiuni de la lider, urmând ca apoi să confirme sau să respingă aceste instrucțiuni. În cazul în care liderul devine indisponibil, unul dintre urmăritori va fi ales ca noul lider prin vot. Un alt aspect important al RAFT este reprezentat de alegerile de lider. RAFT utilizează un sistem de votare în care fiecare nod primește un vot, iar nodul care primește cele mai multe voturi devine liderul. Acest sistem de votare are ca scop asigurarea alegerii unui lider legitim și evitarea situațiilor în care două noduri își declară liderul în același timp. În general, algoritmul RAFT este utilizat în sistemele distribuite care necesită un consens rapid și sigur între un număr mic de noduri. Datorită ușurinței de implementare și înțelegere, RAFT a devenit unul dintre cei mai populari algoritmi de consens din industria IT.

⁹Diego Ongaro și John Ousterhout, au adus contribuții semnificative în domeniul sistemelor distribuite, punând bazele pentru dezvoltarea și implementarea algoritmilor de consens și a sistemelor de stocare distribuită.

Capitolul 3

Hyperledger Fabric

3.1 Introducere în Hyperledger Fabric

O rețea Fabric blockchain constă într-un set de noduri care formează o rețea . Deoarece Fabric este permis, toate nodurile care participă la rețea au o identitate, așa cum este furnizată de un furnizor de servicii de membru . Nodurile dintr-o rețea Fabric adoptă unul dintre cele trei roluri:

Clienții trimit propuneri de tranzacții pentru a fi executate, ajută la orchestrarea fazei de execuție și, în cele din urmă, transmit tranzacțiile pentru a fi ordonate. Nodurile pereche execută propunerile de tranzacții și validează tranzacțiile. Toate nodurile pereche mențin registrul blockchain, o structură de date de tip doar adăugare , care înregistrează toate tranzacțiile sub forma unui lanț de hash-uri, precum și starea, o reprezentare succintă a celei mai recente stări a registrelor. Nu toate nodurile pereche execută toate propunerile de tranzacții, ci numai o submulțime dintre acestea, numite moduri pereche care furnizează aviz conform politicii de cod în lanț (chaincode) menționată în tranzacție.

Nodurile serviciului de ordonare sunt nodurile care formează colectiv serviciul de ordonare. În principiu, serviciul de ordonare stabilește ordinea totală a tuturor tranzacțiilor în Fabric, unde fiecare tranzacție conține actualizări ale stării și dependențe calculate în timpul fazei de execuție, împreună cu semnături criptografice ale evaluatorilor. Ordonatorii nu sunt conștienți de starea aplicației și nu participă la executarea sau validarea tranzacțiilor. Această alegere de proiectare face consensul în Fabric cât mai modular posibil și simplifică înlocuirea protocolului de consens în Fabric. De fapt, o rețea Fabric suportă mai multe blockchain-uri conectate la același serviciu de ordonare. Fiecare astfel de blockchain este numit canal și poate avea membri diferiți. 3.1

Canalele pot fi utilizate pentru a partiționa starea rețelei blockchain, dar consensul în cadrul canalelor nu este coordonat și ordinea totală a tranzacțiilor în fiecare canal

este separată de celelalte. Anumite implementări care consideră toți ordonatorii ca fiind de încredere pot, de asemenea, implementa controlul accesului pe canal pentru nodurile pereche.

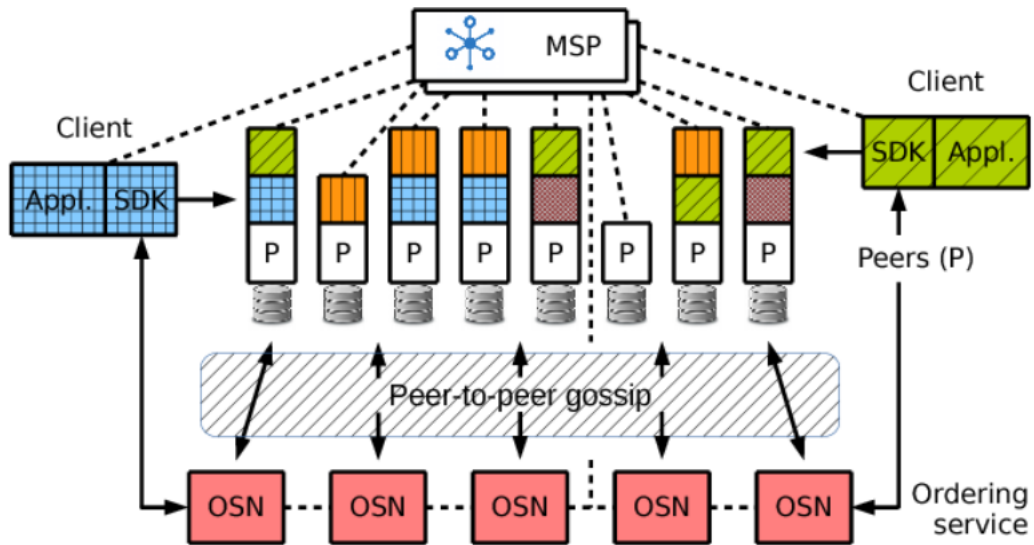


Figura 3.1: O rețea Fabric care rulează multiple chaincode-uri

3.2 Arhitectura blockchain-ului Fabric

Fabric introduce o nouă arhitectură blockchain cu scopul de a avea reziliență, flexibilitate, scalabilitate și confidențialitate. Proiectat ca un blockchain permis modular și extensibil, fabric este primul sistem blockchain care susține execuția aplicațiilor distribuite scrise în limbaje de programare standard, într-un mod care le permite să fie executate consecvent pe multe noduri, oferind impresia de execuție pe un singur computer blockchain global distribuit. Aceasta face ca Fabric să fie primul sistem de operare distribuit [Tan93] pentru blockchain-uri permise.

Arhitectura Fabric urmează o nouă paradigmă de execuție-ordonare-validare pentru execuția distribuită a codului neîncredere într-un mediu neîncredere. Aceasta separă fluxul de tranzacții în trei pași, care pot fi rulați pe entități diferite în sistem: Executarea unei tranzacții și verificarea corectitudinii sale, validarea acesteia (corespunzând "validării tranzacției" în alte blockchain-uri); Ordonarea prin intermediul unui protocol de consens, indiferent de semantica tranzacției; Validarea tranzacției pe baza asumpțiilor specifice aplicației de încredere, care previne, de asemenea, condițiile de concurență.

Această proiectare se diferențiază radical de paradigmă ordonare-execuție prin faptul că Fabric execută în mod obișnuit tranzacțiile înainte de a ajunge la un acord final asupra ordinii lor. Aceasta combină cele două abordări bine-cunoscute de replicare,

pasivă și activă, astfel:

Fabric folosește replicarea pasivă sau primar-secundară [BMST93, CBPS10], așa cum se găsește adesea în bazele de date distribuite, dar cu procesarea asimetrică a actualizărilor bazată pe middleware ¹ și adaptată la medii nesigure cu defecțiuni bizantine. [KJPPM10] În Fabric, fiecare tranzacție este executată (avizată) doar de un subset de noduri, ceea ce permite executarea în paralel și abordează posibilitatea de nedeterminism, bazându-se pe replicarea "execută-verifică" [KWQ⁺12].

O politică de avizare flexibilă specifică care noduri, sau câți dintre acestea, trebuie să ateste execuția corectă a unui anumit contract inteligent. Fabric încorporează replicarea activă în sensul că efectele tranzacției asupra stării registrelor sunt scrise doar după ce s-a ajuns la un consens cu privire la un ordin total între ele, în etapa de validare deterministă executată de fiecare nod individual. Acest lucru permite Fabric să respecte presupunerile de încredere specifice aplicației în funcție de avizarea tranzacției.

În plus, ordonarea actualizărilor de stare este delegată către un component modular pentru consens (adică difuziune atomică), care este fără stare și logic decuplat de nodurile care execută tranzacțiile și mențin registrele.

Deoarece consensul este modular, implementarea sa poate fi adaptată la presupunerile de încredere ale unei implementări particulare.

Deși este posibil să se utilizeze nodurile blockchain și pentru implementarea consensului, separarea celor două roluri adaugă flexibilitate și permite utilizarea de kituri de instrumente bine stabilite pentru ordonarea CFT (toleranță la defecte de tip crash) sau BFT.

În ansamblu, acest design hibrid de replicare, care combină replicarea pasivă și activă în modelul bizantin, și paradigmă de executare-ordonare-validare, reprezintă principala inovație în arhitectura Fabric. Acestea rezolvă problemele menționate anterior și fac din Fabric un sistem scalabil pentru blockchain-urile permise care susțin presupunerile flexibile de încredere.

Pentru a implementa această arhitectură, fabric conține blocuri de construcție modulare pentru fiecare dintre următoarele componente: Un serviciu de comandă care transmite atomic actualizări de stare către nodurile și stabilește un consens asupra ordinii tranzacțiilor. Un furnizor de servicii de membru este responsabil de asocierea nodurilor cu identități criptografice. Acesta menține natura permisivă a Fabric. Contractele inteligente din Fabric rulează într-un mediu de containere pentru izolare. Acestea pot fi scrise în limbaje de programare standard, dar nu au acces direct la starea registrelor. Fiecare nod menține local registrul sub formă de blockchain doar de adăugare și sub formă de snapshot a celei mai recente stări într-un magazin cheie-

¹un set de software situat între aplicațiile software și sistemul de operare sau între diferite componente ale unei aplicații

valoare.

Fabric introduce arhitectura blockchain "execute-order-validate" 3.2 și nu urmează designul standard order-execute. O aplicație distribuită pentru Fabric constă din două părți:

Un contract inteligent, numit chaincode, care este codul program care implementează logica aplicației și rulează în timpul fazei de execuție. Chaincode-ul este partea centrală a unei aplicații distribuite în Fabric și poate fi scris de un dezvoltator neîncredere. Există chaincode-uri speciale pentru gestionarea sistemului blockchain și menținerea parametrilor, denumite colectiv chaincode-uri de sistem

Politica de avizare care este evaluată în faza de validare. Politicile de avizare nu pot fi alese sau modificate de către dezvoltatorii de aplicații necinstiți. O politică de avizare acționează ca o bibliotecă statică pentru validarea tranzacțiilor în Fabric, care poate fi doar adaptată de chaincode. Doar administratorii desemnați pot avea permisiunea de a modifica politicile de avizare prin funcțiile de management de sistem.

O politică de avizare tipică permite chaincode-ului să specifice aprobatorii pentru o tranzacție sub forma unui set de colegi necesari pentru aprobare; utilizează o expresie logică monotonă pe seturi, cum ar fi "trei din cinci". Politicile personalizate de avizare pot implementa logici arbitrării. [ABB⁺18]

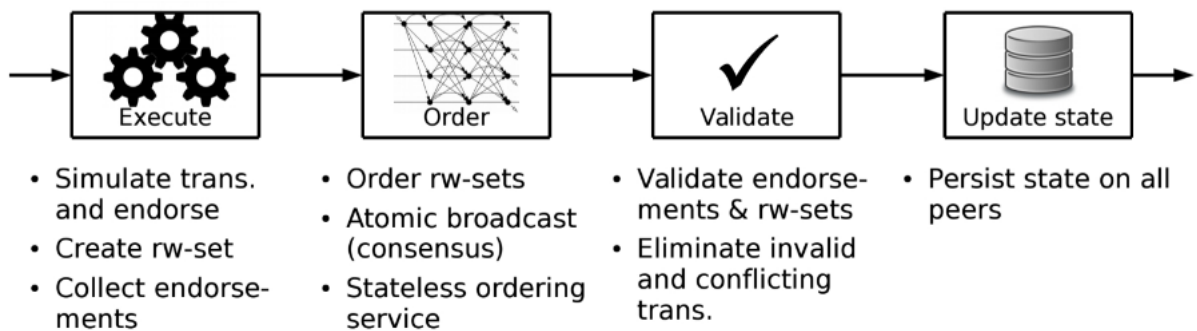


Figura 3.2: Arhitectura ordinii de execuție

Capitolul 4

Metodologie

4.1 Arhitectura aplicație

Arhitectura sistemului de gestionare a datelor pacientului constă în următoarele componente: un canal de comunicare, un consorțiu format din două organizații (spitale), un Furnizor de Servicii de Membru și o Autoritate de Certificare pentru fiecare spital, un utilizator individual pentru fiecare spital, un registru cu o copie a stării lumii pe fiecare utilizator de spital, chaincode-ul (codul lanțului) implementat pe fiecare utilizator de spital, un serviciu de ordonare cu un nod ordonator, o politică de aprobare, actori și o aplicație web. Figura 4.1 poate fi consultată pentru o mai bună înțelegere a arhitecturii propuse.

Pentru a menține arhitectura simplă și scalabilă, există doar un canal (canalul C) în sistem. Toate organizațiile spitalicești accesează acest canal pentru a efectua tranzacții. Inițial, două spitale (organizații) fac parte din această arhitectură, dar există flexibilitate și pot fi adăugate atâtea spitale câte sunt necesare. Fiecare spital are propriul Furnizor de Servicii de Membru și Autoritate de Certificare, care sunt responsabile de generarea perechilor de chei publice-private pentru organizații și semnarea certificatelor.

Organizațiile nu pot interacționa direct cu canalul într-o rețea Hyperledger Fabric, așa că au nevoie de utilizatorii pentru a accesa acest canal. Aici avem câte un nod pentru fiecare spital. Fiecare utilizator are o copie a bazei de date de stare care conține starea lumii curente a rețelei. Atunci când este efectuată o tranzacție de creare, citire, actualizare sau ștergere, starea lumii este actualizată pentru toate registrele utilizatorilor. Sunt scrise contracte inteligente pentru fiecare tip de interacțiune necesară cu rețeaua Hyperledger Fabric. Aceste contracte inteligente sunt împachetate într-un chaincode și acest chaincode este implementat pe fiecare nod pentru spitale.

Orice tranzacție validată de un nod este trimisă serviciului de ordonare, unde se iau decizii ulterioare în conformitate cu politica de aprobare activă. În arhitectura

noastră, există un nod ordonator, dar într-o implementare realistă, ar putea fi utilizate mai multe noduri ordonatoare, ceea ce ar face sistemul mai fiabil și tolerabil la erori. În ceea ce privește politica de aprobare, care specifică organizațiile care trebuie să execute chaincode pentru a valida tranzacțiile, am păstrat politica de aprobare implicită, conform căreia ambele organizații spitalicești trebuie să aprobe o tranzacție. În sistemul nostru există trei actori: administrator, medic și pacient. Fiecare organizație are un administrator care este responsabil de înscrierea medicilor și pacienților. Medicii sunt practicieni medicali, iar pacienții sunt utilizatorii pentru care vor fi create înregistrările medicale în registrele distribuite.

O aplicație web formată dintr-un frontend și un backend acționează ca interfață între actori și rețeaua Hyperledger Fabric. Doar utilizatorii autorizați pot accesa sistemul și își pot îndeplini sarcinile atribuite.

Arhitectura descrisă până în acest moment este cea implementată pentru prototip. Aceasta are numeroase avantaje, cum ar fi simplitatea, securitatea și scalabilitatea ușoară, dar este o versiune rafinată. În faza incipientă a acestui proiect, au fost generate multe idei și merită menționată în mod special o anumită arhitectură. Ideea pentru acea arhitectură presupunea ca fiecare organizație care se alătură rețelei să aibă propriul său canal. Pentru fiecare organizație ar exista două noduri (un nod pentru medici și un nod pentru pacienți). Fiecare nod ar avea o copie a stării lumii și a chaincode-ului specific acelei organizații. Pacienții și medicii ar interacționa cu rețeaua blockchain prin intermediul interfețelor web diferite. Beneficiile acestui design erau securitatea și confidențialitatea. Deoarece datele pacienților erau prezente doar în canalul specific spitalului la care aceștia se prezentau, alți medici și pacienți nu ar putea accesa aceste date. Dezavantajele semnificative erau complexitatea și scalabilitatea deficitară. Numărul de canale într-un astfel de design ar crește în mod liniar în funcție de numărul de spitale adăugate în rețeaua blockchain.

4.1.1 Flux-ul de lucru

Fiecare spital are un administrator care este responsabil de înregistrarea medicilor în acel spital și, de asemenea, a oricărui pacient nou care vizitează acel spital. Sistemul de gestionare a datelor pacientului înregistrează un pacient și îi acordă pacientului întreaga autoritate asupra înregistrării sale de sănătate; Fiecare spital ar avea înregistrarea de sănătate a pacientului în propriul lor registru. Pacientul are autoritatea de a permite oricărui medic să actualizeze înregistrarea sa de sănătate sau să restricționeze orice medic. Acest lucru permite pacienților să se deplaseze ușor de la un medic la alt medic sau de la un spital la alt spital în cadrul rețelei Hyperledger Fabric.

La momentul înregistrării pacientului, este atribuit și un medic pacientului. Înregistrarea

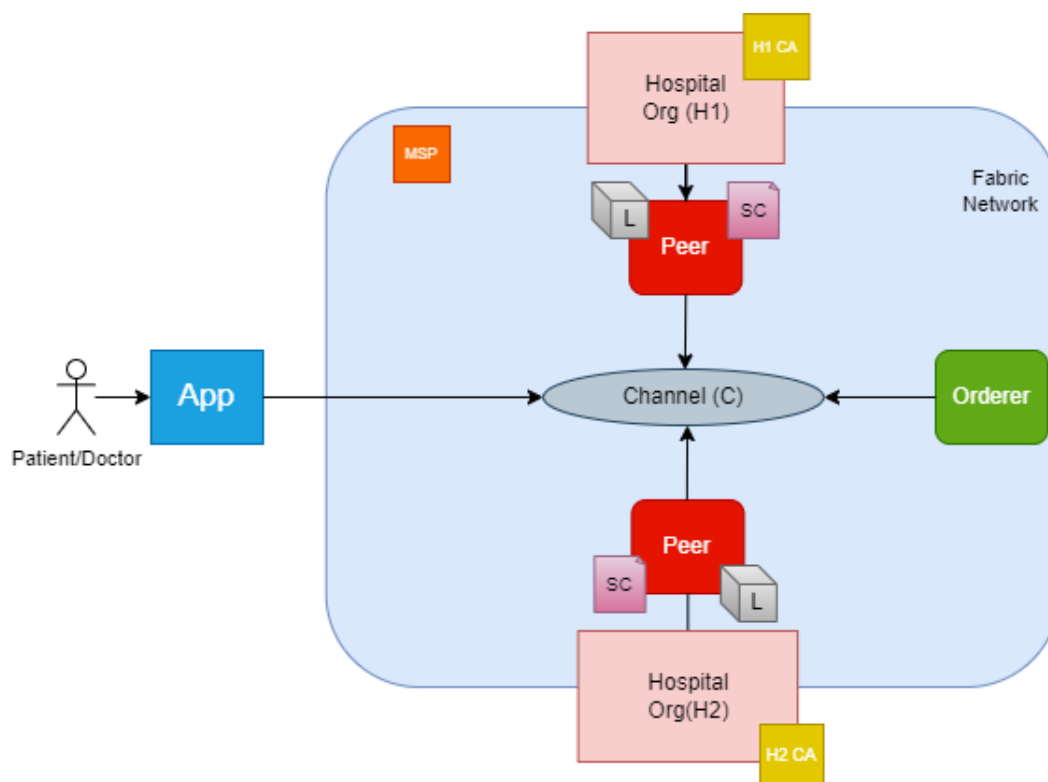


Figura 4.1: Diagrama generală

de sănătate a pacientului conține două părți: date personale și date medicale. Datele personale conțin câmpuri precum nume, ID, adresă. Pacientul are dreptul de a citi și actualiza datele personale, în timp ce poate doar să citească datele medicale și nu poate face modificări în acestea. În plus, pacientul are dreptul de a acorda acces la înregistrarea sa de sănătate oricărui alt medic și de a revoca accesul de la orice medic ori de câte ori dorește.

Odată ce un medic este asignat unui pacient, acesta primește acces la înregistrarea de sănătate a pacientului. Medicii pot citi înregistrările de sănătate ale pacienților și pot actualiza doar datele medicale ale înregistrării de sănătate a pacientului atunci când este necesar. Odată ce pacientul revocă accesul pentru un medic, acesta nu mai poate accesa înregistrarea de sănătate a pacientului.

Există multe scenarii posibile pentru acest sistem atunci când se ia în considerare interacțiunea dintre actorii implicați. În secțiunile următoare, voi discuta câteva dintre scenariile comune.

Scenariu 1 - Pacientul vizitează doctorul 1 în Spitalul 1 pentru prima dată

În acest scenariu, pacientul vizitează un spital din rețeaua blockchain pentru prima dată. Până în acest moment, nu va exista un dosar electronic de sănătate (DES) creat pentru acest pacient în parte. Administratorul înregistrează pacientul în sistem, ceea ce creează un nou DES pentru pacient cu detalii personale și date medicale. Doctorul pe

care pacientul îl vizitează (doctorul 1) este stabilit ca medic pentru această înregistrare și acest doctor, în mod implicit, are accesul pentru a citi sau actualiza acest DES. Desigur, condiția este ca doctorul să fie deja înregistrat de către administrator înainte de a putea fi asignat unui pacient.

Scenariu 2- Pacientul vizitează doctorul 2 în Spitalul 2

Deoarece DES-ul pentru pacientul 1 este deja prezent, nu există nicio acțiune pentru administrator legată de pacient. Cu toate acestea, doctorul trebuie să fie înregistrat în sistem dacă nu este deja înregistrat. DES-ul existent poate fi apoi actualizat într-un mod în care acest nou doctor (doctorul 2) să fie inclus în lista doctorilor autorizați să acceseze DES-ul pentru acest pacient (pacientul 1). Pacientul poate face acest lucru acordând acces noului doctor. Doctorul 2 poate citi sau actualiza acum DES-ul și drepturile doctorului 1 nu sunt afectate de aceasta.

Scenariul 3 - Pacientul a finalizat tratamentul cu Doctorul 1

În acest caz, pacientul poate decide că, deoarece tratamentul s-a încheiat, doctorul 1 nu ar trebui să mai aibă acces la datele pacientului. Pacientul poate revoca accesul doctorului 1 la DES-ul său. Acest lucru va elimina doctorul 1 din lista doctorilor autorizați și nu va mai putea citi sau actualiza înregistrarea pentru pacientul 1. Așa cum era de așteptat, nu va exista nicio schimbare în accesul doctorului 2. Pacientul și doctorul 2 vor putea, desigur, să citească întreaga dată din DES, inclusiv datele generate de doctorul 1.

4.2 Tehnologii

Implementarea aplicației poate fi împărțită în diverse componente, și anume rețeaua Hyperledger Fabric și registrul distribuit, chaincode-ul (contractul inteligent), SDK-ul aplicației și interfața aplicației. Rețeaua reprezintă partea reală a blockchain-ului în aplicație. Chaincode-ul e implementat în Go¹. SDK-ul este dezvoltat folosind Node.js², iar interfața aplicației este construită utilizând React.js³. În această secțiune, voi analiza aceste componente în mai mult detaliu.

¹un limbaj de programare open source dezvoltat de către Google

²un mediu de executare open-source, bazat pe motorul JavaScript V8 dezvoltat de Google, care permite rularea codului JavaScript în afara browser-ului

³o bibliotecă JavaScript open-source utilizată pentru construirea interfețelor utilizator dinamice și reutilizabile

Rețeaua Hyperledger Fabric

Rețeaua Hyperledger Fabric reprezintă baza sistemului. Aceasta constă în spitalele cu nodurile lor care participă în canalul cu registrul distribuit. După cum este explicat în arhitectură, toate spitalele sunt conectate la un singur canal. Pentru ușurința utilizării, s-a folosit test-network-ul disponibil de la Hyperledger pentru configurarea Sistemului de gestiune al dosarelor de sănătate.

Docker

Utilizând rețeaua existentă, organizațiile sunt modificate pentru a reprezenta spitalele prin ajustarea fișierelor Docker, fișierelor de configurare și certificatelor corespunzătoare. Modificările includ în principal ajustarea numelui organizațiilor pentru a include denumirea spitalului în fișierul `configtx.yaml` și Autoritățile de Certificare asociate în fișierul `docker-compose`. Apoi, fișierele care fac referire la aceste fișiere sunt actualizate. Odată ce rețeaua modificată este configurată cu două spitale și un canal, așa cum este prezentat în arhitectură, toate certificatele necesare pentru organizație și nodurile implicate vor fi create

Baza de Date - CouchDB

Conform documentelor oficiale [Hyp21], Hyperledger Fabric suportă două baze de date pentru nodurile participante: LevelDB și CouchDB.

LevelDB reprezintă baza de date implicită pentru Hyperledger Fabric, care stochează datele sub forma de perechi cheie-valoare, în timp ce CouchDB este o alternativă care înregistrează datele sub forma de documente JSON⁴. CouchDB permite interogări complexe asupra documentelor JSON în comparație cu LevelDB, care permite doar interogări cu chei compozite.

În acest proiect, am utilizat CouchDB ca bază de date pentru nodurile participante și am modelat fiecare înregistrare a stării de sănătate a pacienților ca un document JSON, așa cum este prezentat în exemplul de cod de mai jos.

Fiecare înregistrare electronică de sănătate conține mai multe câmpuri, iar un câmp `PatientId` este utilizat pentru a identifica proprietarul înregistrării de sănătate. Celelalte câmpuri sunt destinate detaliilor personale și datelor medicale ale pacientului. Un alt câmp reprezintă un șir de liste de medici autorizați să acceseze înregistrarea; doar medicii menționați în listă pot accesa această înregistrare, în caz contrar, Hyperledger Fabric va refuza accesul. Motivul pentru utilizarea CouchDB este posibilitatea executării de interogări grupate folosind indexarea CouchDB, care permite gruparea

⁴un format de schimb de date ușor de utilizat și de citit de către oameni și de interpretat de către mașini

documentelor JSON (în cazul acestei lucrări, înregistrări de sănătate) în funcție de orice câmp prezent în documentul JSON.

De asemenea, imaginea Docker a CouchDB rulează pe același server ca și nodul participativ, iar numărul de imagini depinde de numărul de noduri participative. Fiecare nod participativ are un registru, deci pentru o rețea Hyperledger Fabric ar fi necesară o imagine CouchDB pentru fiecare nod participativ.

```
1 Record {
2
3     PatientId: "Patient1",
4     Address: "Address1",
5     Telephone: "071234",
6     Diagnosis: "Diagnosic1",
7     Medication: "Medication1",
8     DoctorAuthorizationList: []string{"Doc1"}
9 }
```

Listing 4.1: Exemplu de dosar JSON

Contractele inteligente și chaincode

Toată logica pentru aplicație este implementată folosind contracte inteligente. Aceasta înseamnă că orice operații de creare, citire, actualizare sau ștergere pe registrul distribuit sunt realizate prin intermediul contractelor inteligente. Contractele inteligente pot fi diferite funcții sau chiar fișiere (sau clase) diferite, în funcție de arhitectura și limbajul de programare utilizat. Pentru această lucrare s-a utilizat limbajul GO pentru implementarea contractelor inteligente. Pentru a se menține o arhitectura simplă și modulară, s-a scris câte o funcție pentru fiecare funcționalitate cu care sistemul trebuie să interacționeze în rețeaua HLF.

Contractele inteligente sunt în general împachetate în Chaincode și sunt implementate pe rețeaua blockchain. Astfel, un chaincode poate avea mai multe contracte inteligente și odată ce acest chaincode este implementat pe rețea, toate contractele devin disponibile pentru aplicație. Implementarea chaincode-ului în rețeaua HLF poate fi împărțită în mod general în 4 etape :

1. Împachetarea chaincode-ului
2. Instalarea chaincode-ului pe noduri participative
3. Aprobarea definiției chaincode-ului pentru o organizație
4. Confirmarea definiției chaincode-ului pe canal.

Toate aceste etape pot fi realizate într-un singur pas la executarea comenzii `deployCC` după ce rețeaua HLF este activă. Trebuie să fie furnizate flag-urile corecte pentru calea chaincode-ului și limbajul în care este scris.

Hyperledger Fabric SDK

Conexiunea la rețeaua Hyperledger Fabric pentru a invoca contractele inteligente și a efectua tranzacții se realizează cu ajutorul API-urilor furnizate de Kitul de Dezvoltare a Software-ului al clientului Hyperledger Fabric. "Hyperledger Fabric SDK oferă diverse componente configurabile, cum ar fi algoritmi criptografici pentru semnături, jurnalizare prin intermediul unei interfețe standard extensibile" [Fab22c] . Există diverse SDK-uri disponibile, dar pentru dezvoltarea acestui prototip, am optat pentru Node.js.

Node.js

Node.js este un mediu de rulare JavaScript open-source popular și larg utilizat, care permite dezvoltatorilor să construiască aplicații scalabile și eficiente la nivel de server. Folosește un model de I/O bazat pe evenimente și ceea ce îl face ușor și eficient în gestionarea cererilor concurente. Node.js este în special potrivit pentru construirea aplicațiilor web și a interfețelor de programare la nivel de server.

În ceea ce privește aplicațiile Hyperledger Fabric, Node.js poate fi utilizat ca limbaj de programare pentru dezvoltarea de contracte inteligente, interacțiunea cu rețeaua Fabric și implementarea logicii de afaceri. Node.js oferă un ecosistem solid de biblioteci și cadre de lucru, ceea ce facilitează dezvoltarea, testarea și implementarea aplicațiilor Hyperledger Fabric.

React.js

React.js este o bibliotecă JavaScript open-source pe care o folosesc pentru a construi interfețe utilizator captivante și interactive. Cu ajutorul React.js, pot crea componente reutilizabile și pot dezvolta rapid o interfață dinamică și modernă pentru aplicația web. Prin utilizarea conceptelor de stări și proprietăți ale componentelor, pot actualiza în timp real informațiile afișate .

Beneficiile utilizării React.js în aplicație sunt multiple. Poate crea componente modulare și extensibile, ceea ce ajută să se organizeze și să structureze codul sursă într-un mod eficient. Aceasta înseamnă că se pot adăuga și gestiona diferite vizualizări și funcționalități pentru utilizatori într-un mod rapid și ușor de întreținut.

În ceea ce privește integrarea cu Hyperledger Fabric, se poate utiliza SDK-ul Fabric pentru a interacționa cu rețeaua Fabric și a efectua tranzacții pe registrul distribuit. Prin intermediul React.js, se poate gestiona interacțiunea cu SDK-ul în componente,

oferind utilizatorilor acces securizat la datele lor medicale și posibilitatea de a le actualiza în mod eficient

4.3 Aplicația

Aplicația ar trebui să aibă instalate unele instrumente precum docker, docker-compose, node, npm, golang și curl. Pot fi urmăriți pașii detaliați de pe site-ul oficial al Hyperledger Fabric pentru versiunea specifică și comenzile necesare. [Fab22a] Pentru a descărca fabric basic framework care generează binare specifice platformei și imagini Docker se poate urmări acest link [Fab22b] Recomand folosirea acestor versiuni pentru imaginile docker : versiunea 2.4.9 pentru fabric și versiunea 1.5.5 pentru fabric-ca.

4.3.1 Scriptul pentru rețea

Codul 4.2 bash este utilizat pentru a inițializa și porni rețeaua Hyperledger Fabric. Această comandă va descărca și instala toate dependențele necesare, configurând mediul pentru a rula rețeaua . După rularea cu succes a acestui cod, rețeaua Hyperledger Fabric va fi în funcțiune.

```
1
2 function createConsortium() {
3
4     which configtxgen
5     if [ "$?" -ne 0 ]; then
6         fatalln "configtxgen tool not found."
7     fi
8
9     infoln "Generating Orderer Genesis block"
10
11     set -x
12     configtxgen -profile TwoOrgsOrdererGenesis -channelID
13     system-channel -outputBlock
14     ./system-genesis-block/genesis.block
15     res=$?
16     { set +x; } 2>/dev/null
17     if [ $res -ne 0 ]; then
18         fatalln "Failed to generate orderer genesis block..."
19     fi
20 }
```

Listing 4.2: Funcție pentru a crea primul bloc

```

1
2 function networkUp() {
3
4     checkPrereqs
5     if [ ! -d "organizations/peerOrganizations" ]; then
6         createOrgs
7         createConsortium
8     fi
9
10    COMPOSE_FILES="-f ${COMPOSE_FILE_BASE}"
11
12    if [ "${DATABASE}" == "couchdb" ]; then
13        COMPOSE_FILES="${COMPOSE_FILES} -f ${COMPOSE_FILE_COUCH}"
14    fi
15
16    IMAGE_TAG=$IMAGETAG docker-compose ${COMPOSE_FILES} up -d
17    2>&1
18
19    docker ps -a
20    if [ $? -ne 0 ]; then
21        fatalln "Unable to start network"
22    fi
23 }

```

Listing 4.3: Funcție pentru a porni rețeaua

4.3.2 Chaincode

Contractele inteligente sunt în general dezvoltate în jurul entității asupra căreia se presupune că vor avea loc tranzacțiile în rețea. În cazul lucrări de față, dosarul de sănătate și contracte inteligente sunt dezvoltate în jurul acestui concept.

Funcția `CreateRecord()` listată la 4.7 ajută la crearea unui nou dosar în registrul distribuit atunci când un administrator înregistrează un nou pacient.

```

1 func (rc *RecordContract) CreateRecord(ctx
2 contractapi.TransactionContextInterface, userObjJSON string)
3
4 (*Record, error) {
5     userObj := struct {
6         PatientId string `json:"patientId"`

```

```

7      Address string `json:"address"`
8      Telephone string `json:"telephone"`
9      Diagnosis string `json:"diagnosis"`
10     Medication string `json:"medication"`
11     DoctorId string `json:"doctorId"`
12     OrgId      string `json:"orgId"`
13 }{}
14
15 err := json.Unmarshal([]byte(userObjJSON), &userObj)
16 if err != nil {
17     return nil, err
18 }
19
20 exists, err := rc.RecordExists(ctx, userObj.PatientId)
21 if err != nil {
22     return nil, err
23 }
24
25 if exists {
26     return nil, fmt.Errorf("The record %s already
27
28     exists", userObj.PatientId)
29 }
30
31 record := Record{
32     PatientId: userObj.PatientId,
33     Address:   userObj.Address,
34     Telephone: userObj.Telephone,
35     Diagnosis: userObj.Diagnosis,
36     Medication: userObj.Medication,
37     DoctorAuthorizationList: []string{userObj.DoctorId},
38     OrganisationAuthorizationList: []string{userObj.OrgId},
39 }
40
41 recordJSON, err := json.Marshal(record)
42 if err != nil {
43     return nil, err
44 }
45

```

```

46
47     err = ctx.GetStub().PutState(record.PatientId, recordJSON)
48     if err != nil {
49         return nil, err
50     }
51
52     return &record, nil
53 }

```

Listing 4.4: Functia pentru a crea un dosar

Contractul `GetRecord()` este declanșat atunci când un medic sau un pacient încearcă să citească un dosar.

```

1
2 func (rc *RecordContract) GetRecord(ctx contractapi.
3 TransactionContextInterface, patientId string) (*Record, error) {
4     recordJSON, err := ctx.GetStub().GetState(patientId)
5     if err != nil {
6         return nil, err
7     }
8     if recordJSON == nil {
9         return nil, fmt.Errorf("The record %s does not exist",
10 patientId) }
11
12     var record Record
13     err = json.Unmarshal(recordJSON, &record)
14     if err != nil {
15         return nil, err
16     }
17
18     return &record, nil
19 }

```

Listing 4.5: Functia pentru a citi un dosar

Contractul `GetRecordHistory()` este utilizat pentru a obține istoricul unei anumite DES. Funcția `GetHistory` reprezintă o caracteristică a rețelei blockchain HLF, care ajută la obținerea istoricului tranzacțiilor care au avut loc asupra unei entități specifice. Acest lucru poate fi util deoarece starea lumii stochează doar starea actualizată sau cea mai recentă a unei înregistrări, iar cu ajutorul istoricului pot fi urmărite tranzacțiile anterioare.


```

1 func (rc *RecordContract) GetRecordHistory(ctx contractapi.
2 TransactionContextInterface, userObjStr string)
3 (string, error) {
4     var userObj struct {
5         PatientID string `json:"patientId"`
6     }
7
8     if err := json.Unmarshal([]byte(userObjStr), &userObj);
9
10    err != nil {
11        return "", err
12    }
13
14    resultsIterator, err := ctx.GetStub().GetHistoryForKey
15
16    (userObj.PatientID)
17    if err != nil {
18        return "", err
19    }
20    defer resultsIterator.Close()
21
22    var allResults []*Record
23    for resultsIterator.HasNext() {
24        queryResponse, err := resultsIterator.Next()
25        if err != nil {
26            return "", err
27        }
28
29        var record Record
30        err = json.Unmarshal(queryResponse.Value, &record)
31        if err != nil {
32            return "", err
33        }
34
35        allResults = append(allResults, &record)
36    }
37
38    jsonResults, err := json.Marshal(allResults)

```

```

39     if err != nil {
40         return "", err
41     }
42
43     return string(jsonResults), nil
44 }

```

Listing 4.6: Funcția pentru a citi istoricul dosarului

Acordarea și revocarea accesului reprezintă una dintre caracteristicile cheie ale sistemului de management al dosarelor pacienților și este implementată folosind funcțiile `GrantAccess()` și `RevokeAccess()`. Acestea permit pacientului să acorde sau să revoce accesul unui medic la dosarul său.

Este implementată în contractul inteligent, deoarece accesul la date poate fi controlat în momentul recuperării datelor, astfel încât datele să fie în cadrul rețelei Fabric înainte ca verificarea să fie finalizată. Acest lucru se realizează prin existența unei liste de control al accesului în dosar, numită

`DoctorAuthorizationList`, care conține lista de ID-uri ale medicilor autorizați să acceseze acel DES. Pacienții pot specifica un medic căruia trebuie să i se acorde acces din interfața utilizatorului, iar ID-ul celui medic va fi adăugat în DES-ul pacientului.

Similar, pentru a revoca accesul unui medic, pacientul poate selecta medicul și ID-ul celui medic va fi eliminat din `DoctorAuthorizationList`. Doar pacientul va avea acces la această listă și la metoda de acordare sau revocare a accesului, iar medicul nu va avea acces la niciuna dintre acestea. De asemenea, această listă nu va fi vizibilă pentru medic.

Atunci când un medic încearcă să acceseze sau să modifice dosarul unui pacient, sistemul verifică ID-ul utilizatorului medic în raport cu `DoctorAuthorizationList` din acel dosar și dacă acel ID nu este disponibil în listă, atunci medicul nu este autorizat să vizualizeze sau să modifice DES-ul și sistemul va returna mesajul "Acces Refuzat" medicului respectiv. Similar, dacă ID-ul este disponibil, medicul are autorizația de a vizualiza sau modifica dosarul.

```

1 func (rc *RecordContract) GrantAccess(ctx contractapi.
2
3 TransactionContextInterface, userObj string) (string, error) {
4     var err error
5     var updatedRecordJSON []byte
6
7     var user map[string]interface{}
8     err = json.Unmarshal([]byte(userObj), &user)
9     if err != nil {

```

```

10         return "", err
11     }
12
13     patientId := user["patientId"].(string)
14     doctorId := user["doctorId"].(string)
15
16     recordJSON, err := ctx.GetStub().GetState(patientId)
17     if err != nil {
18         return "", fmt.Errorf("Failed to read from
19         world state: %v", err)
20     }
21
22     if recordJSON == nil {
23         return "", fmt.Errorf("The Record %s does not exist",
24         patientId)
25     }
26
27     var existingRecord map[string]interface{}
28     err = json.Unmarshal(recordJSON, &existingRecord)
29     if err != nil {
30         return "", err
31     }
32
33     authList := existingRecord["DoctorAuthorizationList"].
34
35     ([]interface{})
36     authList = append(authList, doctorId)
37
38     updatedAuthList := make([]string, len(authList))
39     for i, id := range authList {
40         updatedAuthList[i] = id.(string)
41     }
42     existingRecord["DoctorAuthorizationList"] = updatedAuthList
43
44     updatedRecordJSON, err = json.Marshal(existingRecord)
45     if err != nil {
46         return "", err
47     }
48

```

```

49     err = ctx.GetStub().PutState(patientId, updatedRecordJSON)
50     if err != nil {
51         return "", fmt.Errorf("Failed to write to
52
53     world state. %v", err)
54     }
55
56     return string(updatedRecordJSON), nil
57 }

```

Listing 4.7: Functia de a garanta acces

Capitolul 5

Rezultate

În acest capitol voi explora rezultatele obținute din implementarea aplicației web bazată pe Hyperledger Fabric, o platformă blockchain puternică și securizată. Rezultatele vor fi bazate pe evaluarea performanței, analiza costurilor și prezentarea unei comparații între Hyperledger Fabric și Ethereum, un alt sistem blockchain popular. Prin intermediul acestui capitol, se vor demonstra aspecte esențiale privind eficiența și potențialul aplicației, precum și avantajele oferite de utilizarea Hyperledger Fabric în comparație cu alte soluții blockchain.

5.1 Analiza performanței aplicației

Performanța este un aspect esențial în dezvoltarea și utilizarea unei aplicații web de succes. Evaluarea și optimizarea performanței sunt cheia pentru a oferi o experiență de utilizare excelentă, garantând un timp de răspuns rapid și eficiență în utilizarea resurselor. Această secțiune, se va concentra asupra performanței aplicației mele Hyperledger Fabric și se vor prezenta rezultatele obținute în urma măsurărilor. Se va examina performanța aplicației atât în condițiile în care un singur utilizator interacționează cu aceasta, cât și în situații în care 50 de utilizatori operează simultan în cadrul aplicației. Am efectuat măsurători și am înregistrat diverse metrice pentru a obține o imagine completă a performanței. Prin intermediul acestor măsurători, se propune să se evalueze timpul de răspuns, utilizarea rețelei, operațiile de citire de pe disc și utilizarea CPU-ului. Aceste metrice vor ajuta la înțelegerea eficienței și scalabilității aplicației, atât în cadrul utilizării normale, cât și în situații de vârf cu mai mulți utilizatori. Rezultatele obținute în această secțiune vor fi utile pentru a identifica eventualele zone de îmbunătățire și pentru a evidenția punctele forte ale aplicației.

5.1.1 Performanța aplicației pentru un utilizator

În această secțiune, se va evalua performanța aplicației când un singur utilizator 5.1 interacționează cu ea timp de o oră. Rezultatele obținute sunt următoarele: Numărul de requesturi trimise: 2265 Requesturi pe secundă: 0.72 Timpul mediu de răspuns: 395 ms. Rata de eroare: 0 Utilizarea rețelei: 2.9 Mib. Operațiile de citire de pe disc: 2.73 citiri/sec. Utilizarea CPU-ului: 51.16

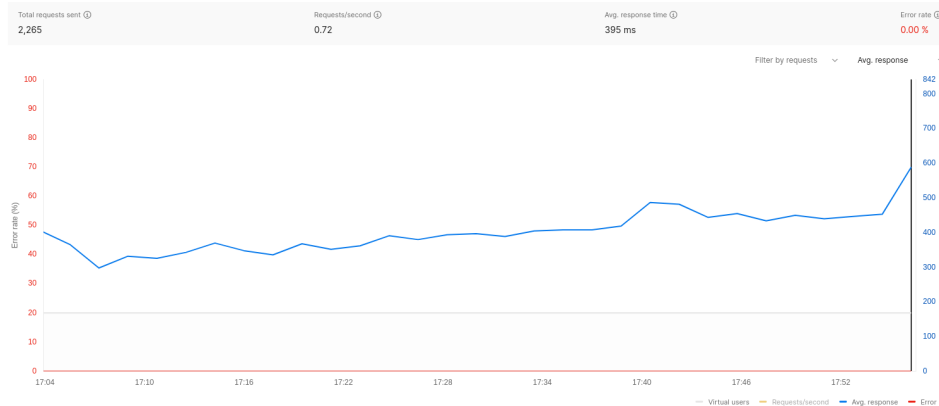


Figura 5.1: Api pentru un user

5.1.2 Performanța aplicației pentru 50 de utilizatori

În această secțiune, se va evalua performanța aplicației când 50 de utilizatori 5.2 interacționează cu ea timp de 5 minute. Rezultatele obținute sunt următoarele:

Numărul de requesturi trimise: 1137 Requesturi pe secundă: 4.27 Timpul mediu de răspuns: 12,450 ms. Rata de eroare: 3.8% erori. Utilizarea rețelei: 90.6 Mib. Operațiile de citire de pe disc: 15.25 citiri/sec. Utilizarea CPU-ului: 61.15% utilizare.

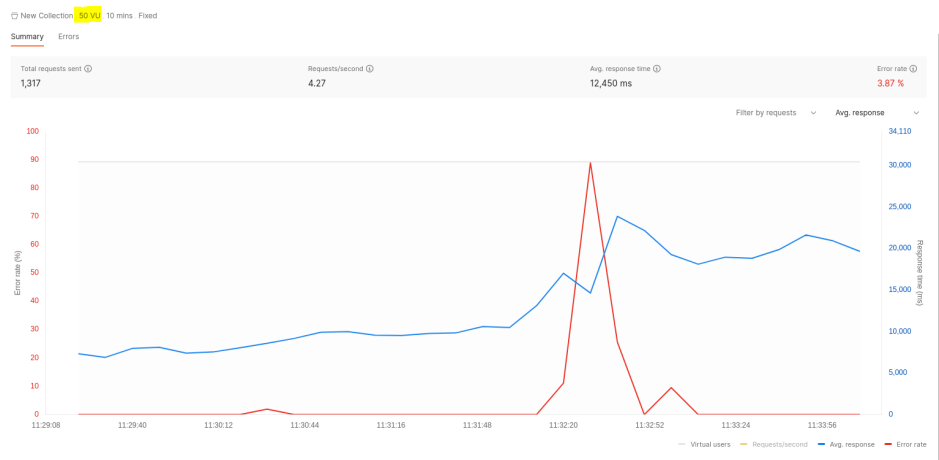


Figura 5.2: Api pentru 50 useri simultan

5.2 Costul Aplicației

Costul aplicației web Hyperledger Fabric poate varia în funcție de mai mulți factori, inclusiv utilizarea infrastructurii de cloud¹ și resursele necesare pentru a asigura funcționarea adecvată a aplicației. În cadrul acestui capitol, se vor explora principalele componente care pot influența costurile aplicației. Pentru a testa costul Aplicației s-a ales providerul de cloud de la Microsoft , Azure.

Costul Mașinilor Virtuale

Pentru a rula aplicația web Hyperledger Fabric, este necesară utilizarea mașinilor virtuale pentru a găzdui și executa componentele acesteia. Costul mașinilor virtuale poate varia în funcție de specificațiile dorite și de furnizorul de servicii cloud utilizat. Pentru aplicație s-a selectat dimensiunea Standard-D4s-v3 pentru o mașina virtuală, care dispune de 4 unități de procesare și 16 GB de memorie RAM. Costul lunar al acestei mașini virtuale este de 130.89 \$. În funcție de cerințele specifice, s-ar putea să fie necesară utilizarea mai multor mașini virtuale pentru asigurarea scalabilității și redundanței aplicației.

Costul Stocării

Aplicația web Hyperledger Fabric poate necesita stocarea datelor, cum ar fi tranzacțiile blockchain sau alte informații relevante. Costul stocării poate varia în funcție de volumul de date și de tipul serviciului de stocare utilizat.

Costul Utilizării Kubernetes

Pentru gestionarea și orchestrarea aplicației web Hyperledger Fabric, se poate utiliza Kubernetes, o platformă open-source pentru administrarea și scalarea containerelor. Kubernetes în sine este gratuit, însă utilizarea sa implică costuri asociate cu resursele infrastructurii utilizate în cadrul clusterului Kubernetes.

Aceste costuri pot include: Costul mașinilor virtuale pentru nodurile Kubernetes, costul stocării persistente, costul rețelei și al altor factori relevanți pentru infrastructura utilizată.

Cheltuielile specifice pot varia în funcție de furnizorul de servicii cloud, dimensiunea și cerințele clusterului Kubernetes.

Pentru aplicația în discuție, ar fi necesară crearea unei mașini virtuale pentru fiecare imagine de docker listată mai jos:

- `docker-peer0.hospital1.com`

¹o infrastructură virtuală în care resursele și serviciile sunt furnizate la cerere printr-o rețea, cum ar fi internetul

- `docker-peer0.hospital2.com`
- `docker-orderer.example.com`
- `Container orderer.example.com`
- `Container couchdb0`
- `Container couchdb1`
- `Container peer0.hospital1.com`
- `Container peer0.hospital2.com`

Dacă s-ar folosi aceeași dimensiune ca și în cazul mașinii virtuale utilizate pentru rularea aplicației pe o singură mașină virtuală, respectiv Standard-D4s-v3, costul total ar fi de 1047.12\$ pe lună 5.3, numai pentru a rula mașinile virtuale utilizând Kubernetes.

Alte Costuri Posibile

Pe lângă costurile menționate mai sus, este posibil să existe și alte costuri asociate cu dezvoltarea, mentenanța și administrarea aplicației web Hyperledger Fabric. Aceste costuri pot include cheltuieli cu licențe de software, servicii de securitate și altele.

Observație: Este important de menționat că costurile menționate în acest capitol sunt estimate și pot varia în funcție de provider-ul de cloud, fluctuațiile pieței și de cerințele specifice ale aplicației.

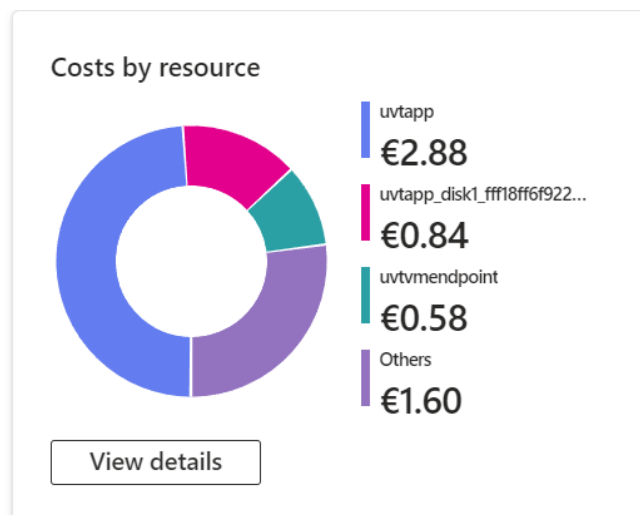


Figura 5.3: Estimare a costului pentru o zi

5.3 Interfața web a aplicației

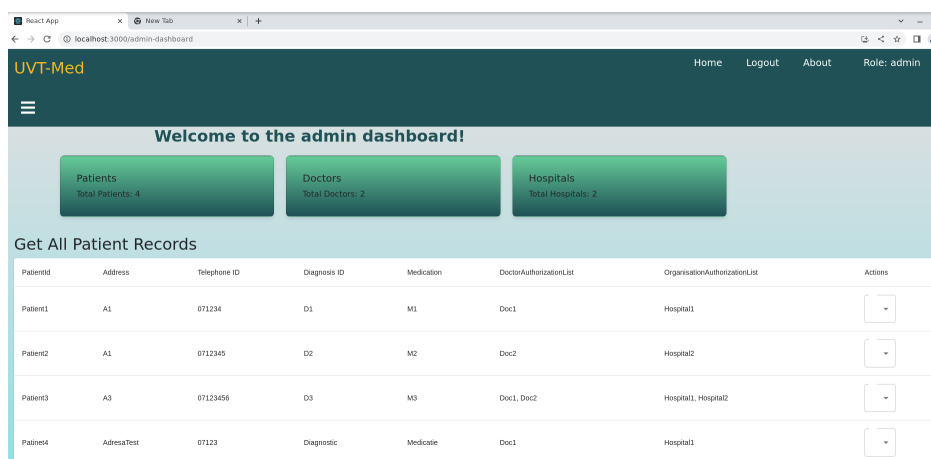
Interfața web a aplicației prezintă o gamă diversă de funcționalități și operații, oferind o experiență utilizator intuitivă și acces facil la informațiile și funcțiile cheie. În această secțiune, se va explora interfața web, evidențiind principalele operații și pagini ale aplicației.

5.3.1 Panoul de administrare

Panoul de administrare 5.4 reprezintă o secțiune esențială a interfeței web, destinată administratorilor sistemului. Aici pot fi efectuate diferite operații, cum ar fi obținerea numărului de pacienți, doctori și spitale, vizualizarea tuturor pacienților și dosarelor lor medicale într-o tabelă. În cadrul tabelii pacienților și dosarelor lor medicale, interfața web include o funcționalitate denumită **Actions**. Aceasta permite administratorilor să efectueze operații suplimentare asupra dosarelor medicale, inclusiv:

- **Istoric:** Prin intermediul butonului **Get History**, administratorii pot accesa istoricul complet al modificărilor efectuate asupra dosarului medical selectat. Această funcționalitate oferă transparență în procesul de gestionare a datelor medicale.
- **Șterge:** Utilizând butonul **Delete**, administratorii pot șterge complet dosarul medical selectat din sistem, împreună cu toate informațiile aferente.

Aceste funcționalități avansate permit administratorilor să gestioneze eficient și securizat datele medicale și să efectueze operații specifice asupra acestora.

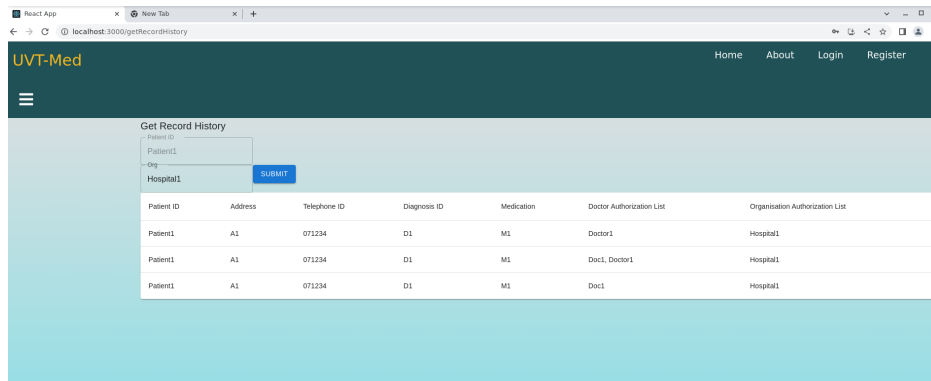


PatientId	Address	Telephone ID	Diagnosis ID	Medication	DoctorAuthorizationList	OrganisationAuthorizationList	Actions
Patient1	A1	071234	D1	M1	Doc1	Hospital1	
Patient2	A1	0712345	D2	M2	Doc2	Hospital2	
Patient3	A3	07123456	D3	M3	Doc1, Doc2	Hospital1, Hospital2	
Patient4	AdresaTest	07123	Diagnostic	Medicatie	Doc1	Hospital1	

Figura 5.4: Panoul de administrare

5.3.2 Pagina de Istoric pentru pacienți

Interfața web include și o pagină dedicată funcționalității **GetHistory**, destinată exclusiv pacienților. Aici, pacienții pot accesa istoricul *5.5* propriului dosar medical prin introducerea numelui spitalului și apăsarea butonului **Submit**.



Patient ID	Address	Telephone ID	Diagnosis ID	Medication	Doctor Authorization List	Organization Authorization List
Patient1	A1	071234	D1	M1	Doctor1	Hospital1
Patient1	A1	071234	D1	M1	Doc1, Doctor1	Hospital1
Patient1	A1	071234	D1	M1	Doc1	Hospital1

Figura 5.5: Pagina de Istoric pentru pacienți

Această pagină oferă pacienților posibilitatea de a vizualiza istoricul dosarului lor medical, oferindu-le un control mai mare asupra datelor lor și o înțelegere mai profundă a evoluției tratamentului și intervențiilor medicale anterioare.

5.3.3 Pagina de înregistrare

Interfața web include, de asemenea, o pagină dedicată procesului de înregistrare a utilizatorilor *5.6*. Această pagină permite potențialilor utilizatori să-și creeze conturi în aplicație, furnizând informațiile necesare și respectând criteriile de securitate.

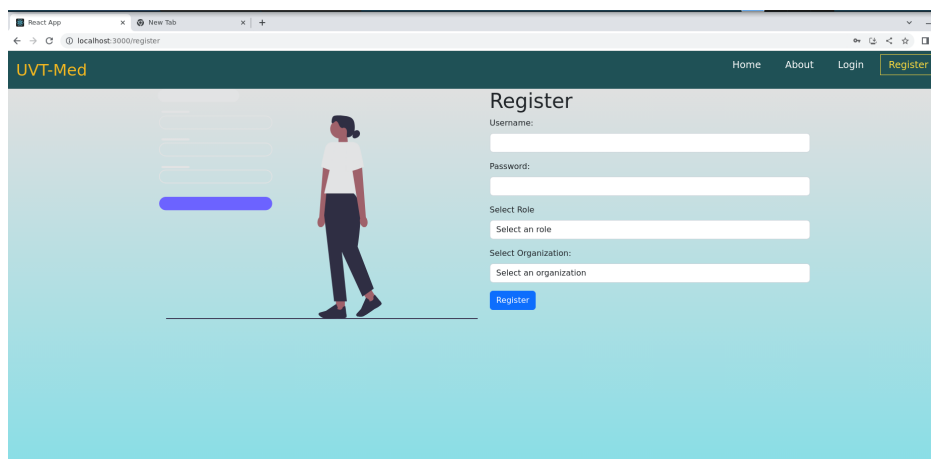


Figura 5.6: Pagina de înregistrare

Page de înregistrare facilitează procesul de aderare la aplicație și asigură că doar utilizatorii autorizați au acces la funcțiile și datele sensibile.

Prin intermediul interfeței web , aplicația oferă o experiență utilizator intuitivă și facilitează gestionarea eficientă a datelor medicale în cadrul rețelei Hyperledger Fabric. Imaginile furnizate ilustrează principalele funcționalități și pagini, evidențiind angajamentul față de securitate, transparență și accesibilitate.

5.4 Impactul și beneficiile aplicației

Aplicația aduce numeroase beneficii și are un impact semnificativ asupra gestionării și securizării datelor medicale. În această secțiune, voi explora aceste beneficii și voi face o scurtă comparație între Hyperledger Fabric și Ethereum.

5.4.1 Beneficiile aplicației bazate pe Hyperledger Fabric

Securitate îmbunătățită

Hyperledger Fabric utilizează un model de permisiuni și control al accesului care permite numai entităților autorizate să acceseze și să modifice datele medicale. Prin utilizarea unui sistem de consens distribuit și criptografie avansată, aplicația asigură integritatea și confidențialitatea datelor, reducând riscul de fraudă sau acces neautorizat.

Transparență și Veridicitate

Hyperledger Fabric oferă o transparență sporită prin înregistrarea completă și verificabilă a tuturor modificărilor aduse în rețeaua blockchain. Această caracteristică permite o auditare precisă și ușoară a datelor medicale, fiind utilă în procesele de conformitate și verificare.

Scalabilitate și performanță

Aplicația permite scalabilitatea orizontală prin adăugarea de noi noduri în rețea. Această abordare permite manipularea eficientă a unui volum crescut de date medicale și asigură o performanță ridicată, evitând congestia și întârzierile.

5.4.2 Comparație cu o aplicație Ethereum

În comparație cu o aplicație dezvoltată pe platforma Ethereum, aplicația pe Hyperledger Fabric aduce următoarele avantaje distincte:

Eficiență și performanță îmbunătățite

Aplicația aduce îmbunătățiri în ceea ce privește eficiența și performanța față de o aplicație dezvoltată pe platforma Ethereum. Hyperledger Fabric utilizează un algoritm de consens numit RAFT, care este cunoscut pentru eficiența și viteza sa în gestionarea tranzacțiilor. Acest algoritm permite rețelei să proceseze un număr mare de tranzacții într-un interval de timp scurt, asigurând o performanță ridicată și minimizând timpul de confirmare.

Pe de altă parte, Ethereum utilizează algoritmul Proof of Stake (PoS) pentru consens, care implică selectarea verificatorilor în funcție de cantitatea de criptomonede pe care le dețin. Deși PoS poate fi eficient în unele cazuri, poate apărea o întârziere în confirmarea tranzacțiilor și o limitare a capacității de scalabilitate în comparație cu algoritmul RAFT.

Confidențialitatea datelor îmbunătățită

Hyperledger Fabric oferă opțiuni avansate de control al accesului la date, permițând definirea de politici de confidențialitate la nivel de canal și de utilizator.

În schimb, Ethereum permite accesul public la toate datele înregistrate în blockchain-ul său. Aceasta face ca aplicația să fie mai potrivită pentru gestionarea datelor medicale sensibile.

Costuri și taxe

Hyperledger Fabric implică costuri asociate infrastructurii și resurselor hardware utilizate pentru implementarea și operarea rețelei. Nu există taxe de tranzacție standardizate, iar costurile pot varia în funcție de administrarea și mentenanța infrastructurii utilizate.

5.5 Dezavantaje ale aplicației bazate pe Hyperledger Fabric

Deși această aplicație bazată pe Hyperledger Fabric aduce numeroase beneficii și soluții inovatoare pentru gestionarea datelor medicale, există și câteva dezavantaje pe care trebuie să fie luate în considerare:

Complexitatea dezvoltării

Dezvoltarea unei aplicații bazate pe Hyperledger Fabric poate fi mai complexă în comparație cu alte soluții tradiționale. Înțelegerea conceptelor blockchain, a arhitecturii Fabric și a funcțiilor sale poate necesita o perioadă de învățare suplimentară

Hyperledger Fabric	
Obiectiv	Potrivit pentru întreprinderi ,aplicații B2B și B2C
Criptomonedă	Niciuna
Limbaj contracte inteligente	JavaScript, GO și Java
Mecanism de consens	Mecanism de consens modular.
Confidențialitate	Tranzacții confidențiale
Tranzacții pe secundă (tps)	>2000 tps
Ethereum	
Obiectiv	Potrivit pentru aplicații B2C
Criptomonedă	Ether
Limbaj contracte inteligente	Solidity
Mecanism de consens	Proof Of Stake (PoS)
Confidențialitate	Tranzacții transparente
Tranzacții pe secundă (tps)	Aproximativ 25 tps

Tabela 5.1: Comparație între Hyperledger Fabric și Ethereum

pentru dezvoltatori. Acest aspect poate aduce un cost adițional în timp și resurse pentru dezvoltarea și implementarea aplicației.

Dependența de infrastructură și resurse hardware

Aplicația necesită infrastructură și resurse hardware pentru a rula și a asigura funcționarea corespunzătoare.

Aceasta poate implica costuri suplimentare pentru achiziționarea și administrarea infrastructurii necesare, precum și pentru asigurarea performanței și scalabilității adecvate.

Limitări ale scalabilității

Deși Hyperledger Fabric oferă o capacitate de scalabilitate superioară față de alte platforme blockchain, există totuși unele limitări în ceea ce privește extinderea rețelei. În funcție de configurația și infrastructura utilizată, există un prag al numărului maxim de noduri pe care rețeaua Fabric le poate susține. Aceasta poate impune restricții în ceea ce privește creșterea rețelei și poate necesita revizuirea arhitecturii pentru a aborda aceste limitări.

Interfață utilizator și experiență

Uneori, dezvoltarea unei interfețe utilizator atractive și prietenoase pentru o aplicație bazată pe Hyperledger Fabric poate fi o provocare. Platforma în sine se concentrează mai mult pe funcționalitatea backend și pe securitatea datelor, iar construirea unei interfețe intuitive și ușor de utilizat poate necesita efort și experiență suplimentară în dezvoltarea interfețelor utilizator.

Integrarea cu sisteme existente

Integrarea unei aplicații bazate pe Hyperledger Fabric cu sistemele existente poate fi un proces complex și consumator de timp. Acest lucru se datorează diferențelor de arhitectură, protocol și limbaje de programare între Hyperledger Fabric și alte sisteme software utilizate. Este necesară o planificare și o analiză atentă pentru a asigura o integrare adecvată și funcționarea corectă a aplicației.

5.6 Îmbunătățiri și direcții viitoare

În continuare, se vor explora câteva posibile îmbunătățiri și direcții viitoare având în vedere cerințele și nevoile utilizatorilor:

Deploy pe un Cloud provider și integrarea cu Kubernetes

Una dintre direcțiile viitoare ar fi să se realizeze deploy-ul sistemului pe un Cloud provider, cum ar fi Microsoft Azure. Aceasta ar permite scalabilitatea și disponibilitatea mai mare a sistemului, asigurând un nivel mai înalt de performanță și rezistență la eșecuri. De asemenea, sistemul poate fi integrat cu Kubernetes, un sistem de orchestrare a containerelor, pentru a facilita administrarea și scalarea automată a resurselor.

Adăugarea unei noi organizații pentru scalabilitate

Pentru a demonstra sistemul scalabil, se poate adăuga o a treia organizație în rețea. Aceasta ar permite extinderea sistemului la un număr mai mare de participanți și ar asigura o mai mare distribuție a puterii de calcul și a responsabilităților în rețea.

Prin adăugarea unei noi organizații, se poate crește capacitatea sistemului și putem facilita colaborarea între mai multe entități.

Adăugarea creării unui profil pentru pacient și doctor

Pentru a extinde funcționalitatea sistemului, poate fi implementată posibilitatea creării unui profil personalizat pentru pacient și doctor. Astfel, pacienții ar putea să-și gestioneze informațiile personale și să acceseze istoricul medical într-un mod centralizat și sigur.

De asemenea, medicii ar avea acces rapid la informațiile relevante despre pacienți, precum diagnosticul și tratamentul anterior, ceea ce ar îmbunătăți procesul de diagnosticare și tratament.

Posibilitatea adăugării de imagini pentru pacient la diagnostic

O altă îmbunătățire ar fi adăugarea posibilității de a atașa imagini relevante diagnosticului unui pacient.

Acest lucru ar permite medicilor să vizualizeze și să analizeze imaginile în contextul diagnosticului și tratamentului pacienților. Integrarea cu tehnologii de stocare și procesare a imaginilor medicale ar fi esențială în acest sens.

Altele

În afară de îmbunătățirile menționate mai sus, există și alte direcții viitoare care se pot explora. Acestea pot include: Implementarea unui sistem de notificări și alerte pentru pacienți și medici ,extinderea compatibilității cu diverse dispozitive și platforme, cum ar fi aplicații mobile ,integrarea cu tehnologii emergente, cum ar fi inteligența artificială și analiza datelor în timp real.

Acestea sunt doar câteva dintre direcțiile viitoare care se pot lua în considerare pentru îmbunătățirea și extinderea aplicației

Concluzii

Aceasta lucrare de licență oferă un cadru solid pentru înțelegerea unei aplicații web Hyperledger Fabric și beneficiilor sale în gestionarea dosarelor pacienților.

În această lucrare, s-a explorat implementarea unei aplicații web bazate pe platforma blockchain Hyperledger Fabric și s-a analizat performanța și costurile asociate, s-a efectuat o comparație între Hyperledger Fabric și Ethereum și s-a evaluat rezultatele obținute în contextul unei utilizări individuale și unei utilizări a 50 de utilizatori simultan. S-a explorat conceptul de blockchain și am evidențiat beneficiile sale în securitatea, transparența și imutabilitatea datelor. Hyperledger Fabric, ca platformă de tip blockchain, a fost prezentată ca soluție personalizată și flexibilă pentru nevoile aplicației .

Rezultatele obținute au evidențiat câteva beneficii majore ale aplicației . Aceasta facilitează gestionarea eficientă a dosarelor pacienților, cu acces rapid și simplificat la informațiile relevante. Aceste aspecte pot contribui la îmbunătățirea coordonării îngrijirii medicale, reducerea erorilor și îmbunătățirea experienței pacientului. În ceea ce privește costurile, s-au analizat principalele componente care pot influența costurile aplicației, cum ar fi mașinile virtuale, stocarea și utilizarea Kubernetes. S-au estimat costurile lunare utilizând serviciul Azure de la Microsoft și s-a observat că acestea pot varia în funcție de cerințele specifice ale aplicației.

În concluzie, implementarea aplicației web bazate pe Hyperledger Fabric a obținut rezultate pozitive în ceea ce privește performanța și eficiența. Utilizarea Hyperledger Fabric aduce avantaje semnificative în comparație cu alte soluții blockchain, cum ar fi Ethereum, în ceea ce privește securitatea, scalabilitatea și transparența. Cu toate acestea, există încă aspecte care pot fi îmbunătățite, cum ar fi optimizarea costurilor. Aplicația web dezvoltată în acest proiect are potențialul de a îmbunătăți gestionarea datelor medicale și de a asigura un mediu sigur și transparent pentru utilizatorii săi.

Bibliografie

- [ABB⁺18] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference*, pages 1–15, 2018.
- [AEVL16] Asaph Azaria, Ariel Ekblaw, Thiago Vieira, and Andrew Lippman. Medrec: Using blockchain for medical data access and permission management. In *2016 2nd international conference on open and big data (OBD)*, pages 25–30. IEEE, 2016.
- [BMST93] Navin Budhiraja, Keith Marzullo, Fred B. Schneider, and Sam Toueg. The primary-backup approach. 1993.
- [CBPS10] Bernadette Charron-Bost, Fernando Pedone, and André Schiper, editors. *Replication: Theory and Practice*, volume 5959 of *Lecture Notes in Computer Science*. Springer, 2010.
- [FA16] Victoria Fernandez-Anez. Stakeholders approach to smart cities: A survey on smart city definitions. In *International conference on smart cities*, pages 157–167. Springer, 2016.
- [Fab22a] Hyperledger Fabric. Hyperledger fabric - pre-requisites, 2022.
- [Fab22b] Hyperledger Fabric. Hyperledger fabric - test network, 2022.
- [Fab22c] Hyperledger Fabric. Hyperledger fabric glossary, 2022.
- [Hyp21] Hyperledger Fabric. Couchdb as state database - hyperledger fabric documentation, 2021.
- [KJPPM10] Bettina Kemme, Ricardo Jimenez-Peris, and Marta Patiño-Martínez. *Database Replication*, volume 2. 01 2010.
- [KWQ⁺12] Manos Kapritsos, Yang Wang, Vivien Quema, Allen Clement, Lorenzo Alvisi, and Mike Dahlin. All about eve: Execute-Verify replication for

- Multi-Core servers. In *10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*, pages 237–250, Hollywood, CA, October 2012. USENIX Association.
- [Lia20] Ying-Chang Liang. *Blockchain for Dynamic Spectrum Management*, pages 121–146. Springer Singapore, Singapore, 2020.
- [LSP19] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. In *Concurrency: the works of leslie lamport*, pages 203–226. acm.org, 2019.
- [Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized business review*, page 21260, 2008.
- [PM19] Maria Prokofieva and Shah J Miah. Blockchain in healthcare. *Australian Journal of Information Systems*, 23, Jul. 2019.
- [RR14] Wullianallur Raghupathi and Viju Raghupathi. Big data analytics in healthcare: promise and potential. *Health information science and systems*, 2:1–10, 2014.
- [Sar18] Simanta Shekhar Sarmah. Understanding blockchain technology. *Computer Science and Engineering*, 8(2):23–29, 2018.
- [SG19] Burcu Sakız and Ayşen Hiç Gencer. Blockchain technology and its impact on the global economy. *ON EURASIAN ECONOMIES 2019*, page 98, 2019.
- [Tan93] A S Tanenbaum. Distributed operating systems anno 1992. what have we learned so far? *Distributed Systems Engineering*, 1(1):3, sep 1993.
- [XTH⁺19] Junfeng Xie, Helen Tang, Tao Huang, F Richard Yu, Renchao Xie, Jiang Liu, and Yunjie Liu. A survey of blockchain technology applied to smart cities: Research issues and challenges. *IEEE Communications Surveys & Tutorials*, 21(3):2794–2830, 2019.