MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

BITSTORED
SISTEM DE STOCARE SECURIZATĂ A FIŞIERELOR

LUCRARE DE LICENŢĂ

Absolvent: **Diana BEJAN**
Conducător ştiinţific: **Senior Lector Eng. Cosmina IVAN**

2019

MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

DECAN,
Prof. dr. ing. Liviu MICLEA

DIRECTOR DEPARTAMENT,
Prof. dr. ing. Rodica POTOLEA

Absolvent: **Diana BEJAN**

BITSTORED
SISTEM DE STOCARE SECURIZATĂ A FIŞIERELOR

1. **Enunţul temei:** Crearea unui sistem de stocare a fisierelor in cloud, acesta fiind disponibil sub forma de aplicaţie web. Aplicaţia realizează stocarea fişierelor în forma criptată, pentru a oferi protecţie sporită a datelor, şi compresată pentru utilizarea eficientă a spaţiului de stocare al utilizatorului. De asemenea sistemul oferă un mecanism de semnare a fişierelor prin steganografie.
2. **Conţinutul lucrării:** Pagina de prezentare, Cuprins, Introducere, Obiectivele Proiectului, Studiu Bibliografic, Analiză şi Fundamentare Teoretică, Proiectare de Detaliu şi Implementare, Testare şi Validare, Manual de Intalare şi Utilizare, Concluzii ,Bibliografie, Anexe.
3. **Locul documentării:** Universitatea Tehnică din Cluj-Napoca, Departamentul Calculatoare
4. **Consultanţi:** Senior Lector Eng. Cosmina Ivan
5. **Data emiterii temei:** 1 ianuarie 2019
6. **Data predării:** 12 iulie 2019

Absolvent: _____

Coordonator ştiinţific: _____

MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

Declaraţie pe proprie răspundere privind autenticitatea lucrării de licenţă

Subsemnatul(a) _____,
legitimat(ă) cu _____ seria _____ nr. _____
CNP _____, autorul lucrării _____
_____
_____
elaborată în vederea susţinerii examenului de finalizare a studiilor de licenţă la Facultatea de Automatică şi Calculatoare, Specializarea _____ din cadrul Universităţii Tehnice din Cluj-Napoca, sesiunea _____ a anului universitar _____, declar pe proprie răspundere, că această lucrare este rezultatul propriei activităţi intelectuale, pe baza cercetărilor mele şi pe baza informaţiilor obţinute din surse care au fost citate, în textul lucrării şi în bibliografie.

Declar, că această lucrare nu conţine porţiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislaţiei române şi a convenţiilor internaţionale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în faţa unei alte comisii de examen de licenţă.

În cazul constatării ulterioare a unor declaraţii false, voi suporta sancţiunile administrative, respectiv, *anularea examenului de licenţă*.

Data _____

Nume, Prenume _____

Semnătura

MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

Cuprins

Capitolul 1

Introducere - Contextul proiectului

Creşterea volumului datelor poate fi o problemă uşor de ignorat, deseori acest lucru se întâmplă făa utilizatorii să fie conştienţi. Motivul creşterii este faptul că utilizatorii tind să creeze şi să distribuie volume tot mai mari de date. Începând cu crearea unui document Word pe calculator şi continuând cu o poză distribuită pe telefon, utilizatorii încarcă spaţiile de stocare cu tot mai multe date. Această problemă poate rămâne neobservabilă până în momentul în care spaţiul de stocare este epuizat.

IDC[1](International Data Corporation) a definit trei tipuri de locaţii principale în care are loc digitalizarea şi unde este sunt create cantităţi masive de date: nucleu (centre de date tradiţionale si de tip cloud), muchie (infrastructuri de tip sucursală), şi obiectivele finale (PC-uri, telefoane si dispozitive IoT). Sumarizarea tuturor acestor date, în momentul în care sunt create, capturate sau replicate, se numeşte *Global Datasphere*, acest proces se confruntă cu o creştere spectaculoasă a datelor. IDC estimează că volumul de date din sfera globală va creşte de la 33 Zettabytes[1] în 2018 până la 175 Zettabytes in 2025, evoluţia se poate observa în Figura 1.1.

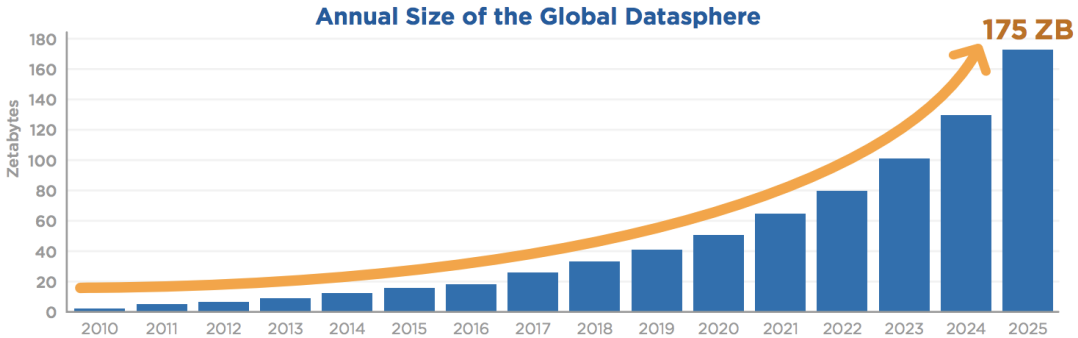

Figura 1.1: Creşterea volumului de date 2010-2025

[1]1 Zetta byte echivalent cu 2^{70} bytes

Securitatea datelor este o problemă foarte mare, cu care se luptă zilnic dezvoltatorii de servicii *cloud*. Multe dintre problemele de securitate se datorează expunerii nivelului de stocare, datele ar trebui să fie stocate în medii securizate, care oferă criptare eficientă şi sigură. Aceste vulnerabilităţi pot fi înlăturate doar prin indeplinirea strictă a unor standarde de securitate avansată.

Creşterea volumului de date, pe de altă parte duce şi la creşterea volumului fizic de *hardware* necesar. Compresia datelor este o operaţie care permite stocarea aceluiaşi volum de date la un preţ mult mai redus[3]. Datorită compresiei un utilizator poate stoca un volum mai mare de date decât ii permite teoretic *hardware*-ul. Pe lânga faptul că se reduce spaţiul de stocare a datelor, compresia contribuie şi la micşorarea timpului necesar pentru transmisie sau pentru operaţiile I/E[4].

Un alt beneficiu al compresiei este creşterea nivelului de securitate. Acest fapt se datorează alterării datelor, astfel chiar şi un algoritm de criptare primitiv devine mult mai complicat de spart prin forţă brută, în ecuaţie se mai adaugă şi decompresia datelor, care este o operaţie costisitoare, nu se mai pot determina asocieri între simboluri, între secvenţe de caractere şi cuvintele dintr-o limbă. Cu toate că se adaugă un timp mare de procesare, în cazul unor date sensibile, această întârziere este una motivată din punctul de vedere al securităţii datelor.

Creşterea încrederii utilizatorilor în sistemele de stocare, creşterea volumului de date, inclusiv şi a celor de sensibilitate înaltă, şi creşterea numărului de atacuri cibernetice crează o nouă provocare pentru sistemele de stocare cloud existente: prestarea serviciilor cu accent pe securitatea datelor. Acest aspect crează şi oportunităţi noi pentru sistemele la început de drum, le oferă o piaţă de desfacere enormă şi un număr ridicat de clienţi care sunt gata să plătească preturi relativ ridicate pentru securitatea datelor sale.

Pentru oferirea securităţii datelor, anual se cheltuie în jur de $100 miliarde, în 2019 această sumă a ajuns la $124 miliarde[1]. Atenţia industriei IT ar trebui să se concentreze în jurul securităţii cibernetice, cu referire mai mult la valoarea datelor, nu la volumul sau sursa lor. Soluţiile sunt şi trebuie să fie conduse de viziunea şi gândirea oamenilor, dar validarea acestora ar trebui să fie condusă de inteligenţa datelor.

În trecutul recent utilizatorii erau responsabili pentru managementul şi stocarea datelor sale, însa dependenţa şi încrederea lor în servciile cloud, în special din cauza conectivităţii, performanţei si confortului, continua să crească, acest lucru duce la noi provocări pentru furnizorii de servicii cloud. Mediul afarcerilor urmăreşte centralizarea managementului datelor, pentru a putea oferi securitate, analiză de date, experienţă utilizator cât mai bună (prin comunicare între dispozitive, IoT, viteză, personalizarea profilului). Responsabilitatea pentru managementul datelor utilizatorilor şi business-urilor duce la o creştere continuă a centrelor de date ale furnizorilor de servicii Cloud. Ca rezultat importanţa serviciilor cloud creşte considerabil, iar utilizatorii nu doar permit acest lucru, ci se şi aşteaptă la o creştere cât mai spectaculoasă.

Volumul de date cât mai mare stocat în cloud este scopul industriei de management al datelor. Pentru a supravieţui într-o lume care tinde a fi condusă de inteligenţa artificială şi sistemele autonome, sistemele de cloud au nevoie să se perfecţioneze tot mai mult pentru

a ţine pasul cu evoluţia lumii digitale.

În calitate de clienţi, oamenii doresc să obţină acces rapid şi simplu la datele lor indiferent de oră şi locaţia în care se află. Sistemele cloud sunt provocate să ofere servicii performante de acces, care nu vor expune datele utilizatorilor.

Organizaţiile şi utilizatorii au început să îşi schimbe destinaţia de stocare a datelor de la infrastructuri fizice spre *cloud*-ul public, alţii însă au început sa îsi dezvolte propriile soluţii de stocare, astfel profitând de toate beneficiile *cloud*-ului, însă securitatea datelor rămâne cea mai mare problemă, mai ales din cauza lipsei de control asupra infrastructurii fizice[2]. Toate sistemele încercă să se bazeze pe modelul CIA[2], acest model a fost creat pentru a ghida seturile de reguli pentru securitatea informaţiilor într-o organizaţie, triada este prezentată în Figura 1.2. Principalele provocări pentru triada CIA sunt: managementul big data, managementul sistemelor IoT, securitatea sistemelor IoT.

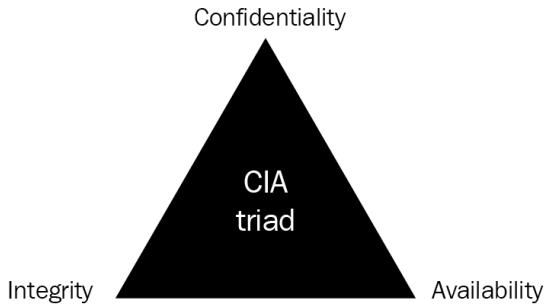

Figura 1.2: Triada CIA

Confidenţialitatea se referă la protejarea datelor de la accese neautorizate. Integritatea se referă la protecţia datelor de la modificări neautorizate, orice modificare poate însemna o pierdere considerabilă pentru utilizator sau organizaţie. Disponibilitatea denotă faptul că informaţiile vor fi accesibile doar pentru utilizatorii care au drept de acces, o încălcare a acestei reguli va provoca pierderea unor date. Toate cele 3 aspecte sunt esenţiale pentru securitatea datelor, însă acestea sunt uneori complicat de oferit.

Aspectele enumerate sunt un avantaj competitiv pentru indrustria IT şi sunt o provocare pentru construirea unei culturi sănătoase a stocării şi managementului datelor.

[2]https://devqa.io/confidentiality-integrity-availability/

Capitolul 2

Obiectivele Proiectului

În acest capitol este prezentată tema proiectului, obiectivele şi cerinţele funcţionale esenţiale ale proiectului.

2.1 Formularea temei

Prin acest poroiect, se urmăreşte implementarea unei platforme de *Cloud Storage*, aceasta trebuie să ofere o bună protecţie a datelor, iar datele să nu fie accesibile dezvoltatorilor sistemului, atacatorilor sau oricărei altei instituţii care colectează date. Sistemul va fi menit pentru stocarea datelor sensibile, sub răspunderea directă a utilizatorilor, de asemenea sistemul poate fi privit ca o platformă fără cunoştinţe despre datele stocate, nimeni altul decât proprietarul fişierelor nu poate decripta datele.

Conceptul sistemului va fi bazat pe arhitectura clasică *client-server*, aceasta este prezentată în Figura 2.1.

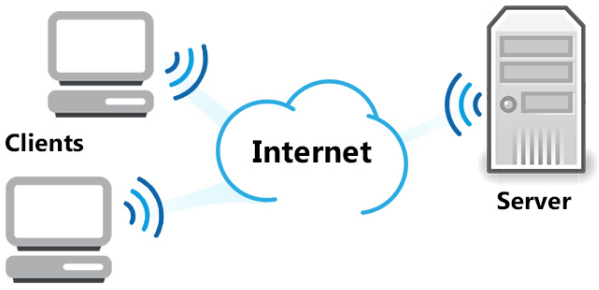

Figura 2.1: Arhitectura Client-Server

Serverul la rândul său va avea o **arhitectură bazată pe microservicii**, fiecare microserviciu va avea structură organizată pe nivele. La nivel de aplicaţie,

responsabilităţile sunt partiţionate între server şi client. La nivelul serverului, funcţionalităţile esenţiale sunt repartizate între servicii, astfel fiind respectat principiul *Single-responsability* din SOLID[5]. La nivel de microserviciu funcţionalităţile sunt divizate utilizând **arhitectură layer**.

Serverul se va ocupa de prelucrarea şi stocarea datelor. Funcţionalităţile cheie sunt: **compresie**, **criptare**, **steganografie**, **monitorizare** şi **stocare**. Partea de stocare a datelor este împărţită în 2: fişiere şi date personale. Datele utilizator vor fi stocate în MongoDB[1], iar cele despre fişiere in Couchbase[2], va fi aproape imposibil să asociezi un fişier cu identitatea unui utilizator fără accesul la ambele baze de date. Va mai exista o a 3-a bază de date, pentru care se utilizează Redis[3], în care sunt stocate logurile din sistem.

Aplicaţia va trebui să fie proiectată astfel încât să poată satisface cereri de la sute sau mii de utilizatori concomitent, fără a implica întârzieri mari de răspuns. Aplicaţia va trebui să fie rezistentă la un nivel mare de utilizare şi să se autoscaleze, doar pentru microserviciile care au un număr mare de apeluri şi nu le pot face faţă în mod rapid.

Sistemul va oferi utilizatorilor funcţionalităţile de bază ale unui sistem de stocare în *cloud*: încărcare fişier, descărcare fişier, creare fişier nou, grupare fişiere şi management fişiere. Pe lângă funcţionalităţile enumerate, sistemul va oferi un nivel de **securitate** înalt prin **criptarea** tuturor datelor şi eficienţă de utilizare a spaţiului prin compresia datelor. Cheia de criptare/decriptare nu va fi stocată nicăieri în server sau client, utilizatorul va avea în totalitate responsabilitatea de a îsi păstra fişierele sigure şi de a putea recupera datele stocate în cloud, din păcate este imposibil să i se ofere asistenţă în cazul pierderii parolei, deoarece regenerarea ei prin forţă brută ar putea dura între câteva ore şi câţiva ani.

De asemenea la descărcarea unui fişier, utilizatorul va avea posibilitatea de a ascunde mesaje în imagini utilizând **steganografia**, sau de a aplica **marcaje** vizibile pe imagini. Această funcţionalitate poate fi folosită atunci când se descarcă un fişier, pentru a determina dacă datele au fost expuse din sistem şi identitatea celui care le-a expus. Steganografia fiind o tehnica de ascundere a datelor, cel mai des acest lucru este realizat prin alterarea celui mai putin semnificativ bit, fapt care nu cauzează schimbări semnificative în percepţia vizuală, dar poate conţine informaţii valoroase pentru calculator.

Pentru client, se doreşte implementarea unei **aplicaţii web**. Aceasta va trebui să interacţioneze cu utilizatorii prin interfaţa web şi cu serverul prin apeluri de tip gRPC[4], pentru a asigura o viteză mai bună de transmisie şi răspuns. Clientului nu îi vor fi cunoscute decât 3 dintre serviciile dezvoltate: serviciul de autentificare, servicul de management utilizatori şi serviciul de management fişiere. Conexiunea între client şi server va fi securizată utilizând protocolul HTTPS. Astfel se va asigura că datele transmise vor fi protejate împotriva atacurilor de tipul *man-in-the-middle*.

[1] https://docs.mongodb.com/
[2] https://www.couchbase.com/
[3] https://redis.io/
[4] https://grpc.io/

2.2 Obiectivele proiectului

- Aplicaţia va fi contruită utilizând arhitectura bazată pe microservcii pentru partea de server.
- Sistemul va fi livrat în Google Cloud utilizând tehnologiile Docker şi Kubernetes.
- Sistemul va oferi utilizatorilor experienţa deplină a unui sistem de stocare în Cloud: încărcare fişier, descărcare fişier, modificare fişier, grupare fişiere.
- Sistemul va oferi utilizatorilor un nivel înalt de securitate a datelor personale prin folosirea unor algoritmi de criptare eficienţi. Cheile nu vor fi stocate în sistem, iar datele vor fi aproape imposibil de decriptat de către atacatori.
- Folosirea spaţiului de stocare va fi eficientizată prin folosirea unor algoritmi de compresie eficienţi, care permit reducerea volumului de date de mai mult de 2 ori în cazul unor date uzuale.
- La cererea utilizatorului, se va putea efectua marcarea imaginilor cu mesaje ascunse prin utilizarea steganografiei.
- Aplicaţia va fi disponibilă sub forma de client web.

2.3 Cerinţe

În această secţiune sunt descrise şi enumerate cerinţele principale ale sistemului. Acestea se referă atât la funcţionalităţile propriu-zise, cât şi la experienţa utilizatorului.

2.3.1 Cerinţe funcţionale

În ingineria software, cerinţele funcţionale definesc funcţiile sistemului sau comportamentul său, unde funcţiile sunt descrise ca o specificaţie sau set de acţiuni dintre intrare şi ieşire.

Aplicaţia va oferi funcţionalităţi diferite, în funcţie de rolul utilizatorului. Se poate face distincţia între 2 tipuri de utilizatori: utilizator primar şi administrator. Diagrama rolurilor sistemului este prezentată în Figura 2.2.

Pornind de la obiectivele proiectului, pot fi determinate următoarele cerinţe funcţionale:

CF1 Ca utilizator neînregistrat, doresc să îmi pot crea un cont nou.

CF2 Ca utilizator, doresc să mă autentific în sistem utilizând numele de utilizator şi parola, care au fost indicate la momentul înregistrării.

CF3 Ca utilizator, doresc ca sesiunea mea să fie păstrată şi după închiderea *browser*-ului.

CF4 Ca utilizator, dorec să îmi opresc sesiunea curentă prin deautentificare.

CF5 Ca utilizator, dorec să am posibilitatea de a îmi bloca temporar contul, fără a pierde accesul la date.

CF6 Ca utilizator, dorec să am posibilitatea de a îmi debloca contul.

CF7 Ca utilizator, dorec să am posibilitatea de a îm sterge contul şi toate datele stocate.

CF8 Ca utilizator, doresc să pot modifica datele mele de utilizator, cum ar fi parola, nume, poză de profil.

CF9 Ca utilizator, doresc să pot primi o confirmare a înregistrării pe mail-ul indicat la momentul înregistrării în sistem.

CF10 Ca utilizator, doresc să pot vizualiza profilul meu.

CF11 Ca utilizator, doresc să pot vizualiza fişierele mele din *Drive* şi să pot naviga prin ierarhia de directoare.

CF12 Ca utilizator, doresc să pot crea un fişier nou, care să aibă sau nu conţinut.

CF13 Ca utilizator, doresc să pot crea un director nou.

CF14 Ca utilizator, doresc să pot încarca un fişier în sistem.

CF15 Ca utilizator, doresc să pot modifica un fişier care se află în spaţiul meu de stocare.

CF16 Ca utilizator, doresc să pot şterge un fişier din sistemul meu de fişiere.

CF17 Ca utilizator, doresc să pot schimba directorul în care se află un fişier.

CF18 Ca utilizator, doresc să pot crea o copie a unui fişier într-un alt director, fără a îmi creşte spaţiul de stocare utilizat.

CF19 Ca utilizator, doresc să pot descărca un fişier care se află în spaţiul meu de stocare.

CF20 Ca utilizator, doresc să pot codifica mesaje în fişierul descărcat.

CF21 Ca utilizator, doresc să pot vedea dacă în fişierul pe care l-am încărcat sunt codificate mesaje.

CF22 Ca utilizator, doresc să mi se ofere date privind volumul de stocare economisit datorită funcţionalităţilor sistemului.

CF23 Ca utilizator, doresc să pot aplica mesaje vizuale pe imaginile descărcate.

CF24 Ca administrator, doresc să pot vizualiza conturile tuturor utilizatorilor.

CF25 Ca administrator, doresc să pot bloca contul oricărui utilizator.

CF26 Ca administrator, doresc să pot debloza contul oricărui utilizator.

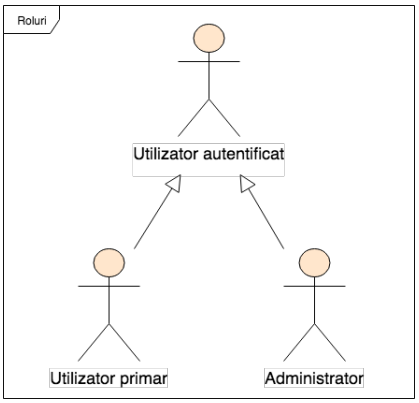

Figura 2.2: Tipurile de utilizatori ale sistemului

2.3.2 Cerinţe non-funcţionale

Cerinţele non-funcţionale se referă mai mult la calitatea şi experienţa de utilizarea a sistemului, decât la functiionalităţile şi capabilităţile specifice. Aceste cerinţe descriu cum ar trebui să fie sistemul, nu ce ar trebui să facă acesta.

CNF1 Codul sursă al sistemului ar trebui să fie scris şi menţinut la cel mai înalt nivel.

(1) Codul trebuie să respecte principiile SOLID.

(2) Dependenţele la librării trebuie înnoite frecvent, pentru a asigura fixarea eventualelor probleme din versiunile precedent.

(3) Codul ar trebui să fie bine testat, cu o acoperire mai mare de 80%.

(4) Nu ar trebui să se folosească soluţii copiate de pe forumuri de programare, acestea ar putea conţine vulnerabilităţi.

(5) Codul trebuie să fie bine documentat şi uşor de citit şi modificat.

CNF2 Sistemul trebuie să ofere un nivel înalt de securitate a datelor.

(1) Datele personale ale utilizatorilor vor fi stocate într-un mediu securizat, acestea vor fi criptate în avans şi vor fi decriptate doar la cererea proprietarului legitim.

(2) Fişierele vor fi criptate utilizând algoritmi compecşi, iar datele despre cheile de criptare nu vor fi stocate în sistem.

(3) Nu se va face o asociere directă între identitatea utilizatorului şi datele stocate în sistemul de fişiere.

CNF3 Sistemul trebuie să utilizeze eficient spaţiul de stocare.

(1) Datele vor fi comprimate cu ajutorul unor algoritmi ce au o rată de compresie foarte mare.

(2) Nu se vor stoca date redundante sau duplicate.

CNF4 Sistemul va avea o interfaţă utilizator inteligibilă, uşor de utilizat şi care nu creează ambiguitate în modul de utilizare.

CNF5 Sistemul va fi rezistent la căderile unor anumite servicii şi îsi va putea reveni ulterior, fără ca utilizatorul să ştie acest lucru.

CNF6 Clientul trebuie să fie suportat de mai multe browsere şi să ofere aceeaşi interfaţă web.

Capitolul 3

Studiu Bibliografic

În acest capitol vor fi prezentate rezultatele studiului bibliografic asupra arhitecturii monolitice, arhitecturii bazate pe microservicii şi tehinicilor de asigurare a securităţii în cloud.

3.1 Caracteristicile arhitecturii monolitice

Monolit însemnă ”dintr-o bucată”. O aplicaţie monolitică este o aplicaţie software în care diferite componente au fost combinate într-un singur program. Componentele programului sunt interconectate şi interdependente, spre deosebire de abordarile modulare care oferă un nivel de cuplare scăzut. Pentru ca programul să fie compilat sau executat fiecare componentă trebuie să fie prezentă şi definită şi să existe legăturile cu fiecare componentă. Componentele aplicaţiei pot fi:

- Autorizarea - responsabilă pentru autoriazarea utilizatorului.
- Prezentarea - responsabilă pentru tratarea apelurilor HTTP şi servirea răspunsurilor.
- Logica de business.
- Componenta de acces la baza de date.

Un monolit poate fi considerat un pattern arhitectural sau un stil de dezvoltare a aplicaţiilor (sau un anti-pattern, dacă privim din perspectiva dezavantajelor). Pattern-urile sunt de obicei grupate în categorii sau seturi, pentru a fi mai uşor de asociat. Categoriile de bază pentru arhitectura monolitică sunt:

- Modul - unităţile de cod sunt separate în module şi sunt compilate împreună producând un singur artefact.
- Alocare - Toate componentele sistemului sunt compilate, livrate şi configurate în acelaşi timp, ca un singur artefact, toate având aceeaşi versiune, indifernt de câte ori au fost modificate. Numărul versiunii este egal cu numărul de livrari al artefactului

- Runtime - Există o singură instanţa aplicaţiei care execută toate sarcinile.

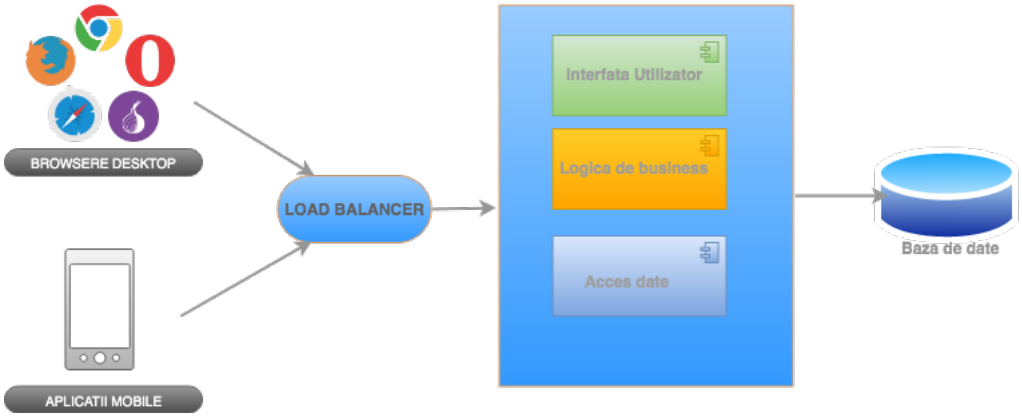

Figura 3.1: Arhitectura monolitică

Avantajele arhitecturiii monolitice sunt:

- Uşor de dezvoltat - la începutul unui proiect este mult mai uşor să dezvolţi o arhitectură monolitică.
- Uşor de testat. De exemplu, se pot implementa teste end-to-end prin simpla rularea a aplicaţiei şi testarea ei cu un tool specializat.
- Uşor de pus în funcţiune, este necesară doar copierea pe un server şi rularea programului.
- Scalabilă pe orizontală prin rularea mai multor instanţe.

Arhitectura monolitică este abordarea tradiţională care este folosită în multe sisteme, care sunt construite ca o aplicaţie autonomă. Chiar dacă este prezentă în multe aplicaţii existente şi încă este folosită pentru dezvoltarea aplicaţiilor noi, limitările şi problemele existente în acest mod de abordare duc la creşterea popularităţii arhitecturii bazate pe microservicii. **Dezavantajele** arhitecturii monolitice sunt:

- Mentenanţa - dacă o aplicaţie este prea mare este foarte greu să faci schimbări rapide şi să nu afectezi funcţionarea corectă a altor componente.
- Dimensiunea aplicaţiei duce creşterea timpul de start-up.
- Toată aplicaţia va trebui restartată atunci când se face o schimbare de cod.
- Este foarte complicat de scalat.
- O problemă în una dintree componente poate afecta întreaga aplicaţie.

- Este complicat să adopte tehnologii si framework-uri noi, deoarece prea multe lucruri trenuie schimbate în acelaşi timp.
- Spre finalul ciclului de dezvoltare complexitatea de a scrie cod devine mai mare, iar raportul timp-eficienţă devine tot mai mare.

Această arhitectură are şi plusuri şi minusuri, dar, dat fiind faptul că de fiecare dată când se rescrie o porţiune de cod este necesară recompilarea întregului program, arhitectura monolitică duce la întârzieri destul de mari, cauzate de compilările repetate a întregului program.

3.2 Caracteristicile arhitecturii orientate pe microservicii

Microserviciile[6] sunt nişte entităţi mici, independente, autonome, create să funcţioneze împreună, fiecare dintre ele este focusat pe un singur lucru şi are scopul de a-l face bine. Arhitectura bazată pe microservicii este un stil arhitectural care structurează aplicaţia ca o colecţie de servicii indepenedente şi modulare, care sunt uşor de testat, de întreţinut şi de înţeles. Acest tip de abordare duce la creşterea agilităţii prin îmunătăţirea productivităţii şi scăderea timpului de dezvoltare a produsului. Microserviciile au demonstrat că sunt un sistem de nivel superior, în special pentru aplicaţii mari care sunt dezvoltate de mai multe echipe. Pe lângă beneficiile enumerate mai sus, microserviciile mai oferă următoareele avantaje:

- Sunt mentenabile.
- Sunt scalabile.
- Sunt uşor de testat.
- Au nivel de cuplare joasă.
- Sunt independente.
- Sunt rezistente la căderi.
- Sunt organizate în funcţie de capabilităţile şi funcţionalităţile de business.
- Oferă independenţă dezvoltatorilor.
- Pot fi dezvoltate independent.
- Modificările pot fi aplicate uşor, deoarece nu mai necesită recompilarea întregului produs.

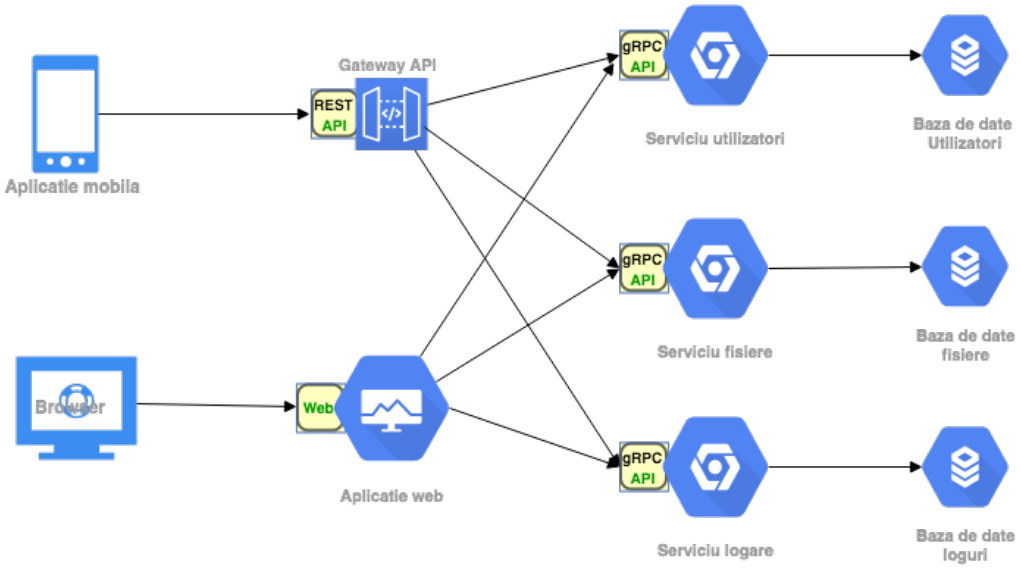

Figura 3.2: Arhitectura bazată pe microservicii

Arhitectura bazată pe microservicii permite integrarea şi livrarea continuă a unui produs complex şi de volum mare, deasemnea promovează diversitatea tehnologiilor utilizate într-un proiect. În prezent, tot mai multe companii au început să folosească arhitecturi bazate pe microservicii pentru produsele lor. Câteva dintre aceste companii sunt[7]:

- Netflix
- eBay
- Amazon
- Twitter
- PayPal
- SoundCloud
- Gilt
- The Guardian

Netflix, eBay şi Amazon sunt cunoscute pentru arhitecturile lor diverse, care au evoluat de la *Monolit* la *Microservicii* cu scopul de a putea face faţă unor volume imense de date.

Totuşi, ca oricare altă soluţie, arhitectura bazată pe microservicii are o serie de dezavantaje[8]:

- Se adaugă complexitate din cauza creării unor sisteme distribuite.
- Programatorul trebuie să implementeze comunicarea între servicii.
- Testarea interacţiunii este destul de complexă.
- IDE-urile şi tool-urile existente au un număr scăzut de funcţionalităţi care ajută la dezvoltarea aplicaţiilor distribuite.
- Un sistem format din microservicii are un nivel mai ridicat al consumului de resurse.

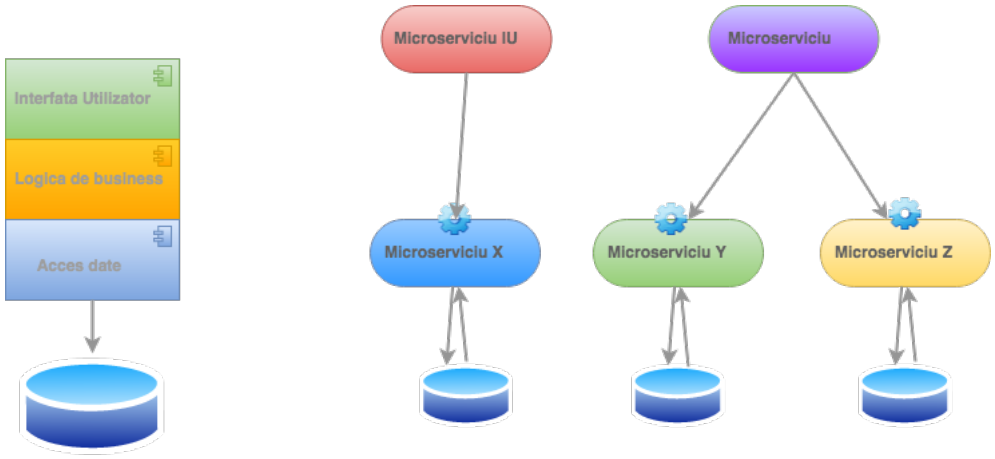

Figura 3.3: Diferenţa dintre arhitecturi

Chiar dacă mulţi dezvoltatori sunt reţinuţi în vderea unei schimbări, sau ezită să încerce o abordare diferită, beneficiile microserviciilor sunt mult mai importante şi mai semnificative decât dezavantajele, în cazul multor apliicaţii[9]. Arhitecturile modulare reduc riscul schimbarilor nedorite sau neanticipate dintr-o componentă în urma modificării altei componente. În concluzie, microserviciile sunt o parte miscării din industria IT care permite o colaborăre mai simplă între echipe. Microserviciile nu sunt doar o tehnologie folosită în prezent, ele sunt un o cultură despre procesul de dezvoltare software[10].

3.3 Microservicii în Cloud

Resursele în *Cloud* sunt disponibile atunci când sunt necesare. Comparativ cu o infrastructură clasică, nu există o limită practică a acestora. Diferite medii de dezvoltare şi versiuni de servicii pot co-exista în mod temporar sau permanent. Programatorul nu mai este nevoit să ghească sau să calculeze cerinţele şi capacitatea de consum a sistemului. La cerere resursele pot fi scalate sau diminuate fără intervenţie în partea fizică a sistemului.

Faptul că plăteşti doar ceea ce utilizezi reduce considerabil preţul experimentării, dar şi creşte posibilităţile de experimentare. Noi funcţiionalităţi pot fi integrate, pot fi oprite şi restartate cu noi parametri în cazul unui eşec. Datorită cloudului se pot efectua numeroase experimente fără riscuri, acets lucru constituind cheia de succes a inovaţiei. Acest fapt se potriveşte ideal cu conceptul de *microservicii*, oferind posibilitatea de a atinge un nivel înalt de agilitate.

Programabilitatea cloud-ului permite automatizarea proceselor de dezvoltare şi livrare. Integrarea contnuă este o parte a ciclului de viaţă a servicilor în cloud. Livrarea continuă, pe de altă parte, introduce provoci noi a complexităţi de administrarea a multiplelor servicii în paralel.

Gândirea, perfecţionarea continuă, livrarea continuă, managementul, monitorizarea şi întreţinerea API-urilor este o responsabilitate complexă şi consumă extrem de mult timp. Sistemele Cloud oferă suport pentru acestea şi uşurează viaţa dezvoltatorilor.

Arhitectura bazată pe microservicii este o abordare distribuită, dezvoltată pentru a rezolva limitările arhitecturii monolitice clasice. Microserviciile facilitează scalarea aplicaţiilor sau a anumitor module ale aplicaţiei. Totuşi sunt o provocare din punctul de vedere a complexităţii arhitecturale şi a operării sistemului. Serviciile cloyd contribuie la reducerea acestei complexităţi prin oferirea unor funcţionalităţi preimplementate de management al serviciilor.

3.4 Securitatea in sistemele informatice

Securitatea aplicaţiilor cuprinde măsurile luate pentru a asigura şi a îmbunătăţi securitatea sistemului. Deseori securitatea este asigurată prin găsirea, înlăturarea şi prevenirea vunlnerabilităţilor sistemului. Sunt utiilizate mai multe tehnici pentru a obţine acest lucru, acestea pot fi aplicate la diferite etape ale ciclului de dezvolatare, cum ar fi: *design, dezvoltare, livrare, perfecţionare sau mentenanţă*. Sunt utilizate diferite abordări pentru a găsi diferite subseturi ale vulnerabilităţilor de securitate, acestea au un impact diferit prin cost, timp, efort şi procentul de vulnerabiliităţi ce pot fi detectate[11]. Tehnicile esenţiale sunt:

- *Whitebox* - se referă la analiza securităţii sau a codului. Această abordare poate fi adoptată doar de un inginer care înţelege deplin codul, şi poate observa problemele prin analiza manuală şi vizuală a codului sursă.

- *Blackbox* - reprezintă auditul în securitate. Operaţia poate fi efectuată de un specialist în securitate, utilizând doar executabilul, fără necesitatea de analiza codul sursă. Inginerul se concentrează pe încercările de a găsi cazuri netratate care pot duce la compomiterea sistemului.

- *Revizuirea design-ului* - înainte de scrierea codului se analizează vulnerabilităţile arhitecturale şi ale dependenţelor externe ale sistemului.

- *Utilizarea tool-urilor* - în prezent există un număr mare de tooluri automate care pot fi utilizate pentru detectarea problemelor de securitate, însă acestea au un număr mai mare de fals pozitive decât în cazurile de testare manuală.

- *Platforme de vulnerabilităţi coordonate* - sunt aplicaţii care oferă recompense hackerilor experimentaţi pentru găsirea de probleme de securitate.

Utilizarea acestor tehnici în mod adecvat îmbunătăţeşte calitatea sistemului prin înlăturarea vulnerabilităţilor. Acest proces este în totalitate responsabilitatea echipelor de dezvoltare şi testare.

În timp ce stocarea în sistemele cloud este convenabilă şi oferă posibilitatea de a accesa datele indiferent de locaţie şi timp de pe aproximativ orice dispozitiv cu conexiune la internet, securitatea sistemelor de stocare este o problemă prioritară a organizaţiilor IT şi a departamentelor de securitate. Beneficiul principal al adoptării stocării în cloud este asigurarea securităţii si integrităţii datelor cu caracter senzitiv.

Dezvoltatorilor sistemelor de cloud le aparţine în totalitate responsabilitatea pentur securitatea aplicaţiilor lor. Aceştia implementează în sistemele lor toate funcţionalităţile esenţiale pentru securitate, acestea fiind: *autentificarea, autorizarea, controlul accesului şi criptarea*[12]. De aici în colo, fiecare companie are responsabilitatea de a adăuga noi nivele de protecţie pentru date şi de a restricţiona cât mai mult accesul la datele sensibile.

3.4.1 Ameninţări

Administratorii de sistem şi dezvolatatorii de servicii software sunt mereu la straja securităţii aplicaţiilor, însă sunt numeroase probleme care par netriviale, acestea pot compromite datele apliicaţiei. Principalele amenintări pentru securitatea aplicaţiilor sunt:

- *Utilizatorii* - utilizatorii aplicaţiilor noastre sunt cea mai mare ameninţare pentru propria lor integritate şi pentru datele lor. Deseori aceştia nu realizează cât de important este să acceseze doar partea sigură a internetului. Nerespectarea unor reguli de bază a navigării pe internet poate compromite parole, chei de acces, chei de criptare, orice efort din partea dezvoltatorilor de a păstra securitatea va eşua în acest caz.

- *Greşeli elementare de structurare sau scriere a codului* - în ciuda multiplelor avertismente şi a anilor de educaţie încă există cod cu greşeli elementare de securitate. Problemele triviale sunt: *SQL-injection* şi *Cross-site scripting*. O recomandare în această direcţie este utilizarea unor librării specializate pentru SQL şi pentru randarea, validarea şi filtrarea datelor provenite de la utilizatori, acestea tratează cazurile menţionate mai sus şi aplicaţia nu poate fi atacată în acest mod.

- *Utilizarea unor librării învechite* - este recomandat să se utilizeze cele mai noi versiuni ale unor librării şi aplicaţii, aceste nu conţin de obicei erorile şi problemele de securitate cunoscute. O recomandare în această direcţie este utilizarea unor

programe software care analizează dependenţele sistemului şi avertizează dezvoltatorii despre eventualele vulnerabilităţi.

- *Setarea unor permisiuni de acces greşite* - prin setarea permisiunilor de acces greşite, utilizatorul poate obţine prea multă libertate şi control asupra aplicaţiei. Este recomandat ca permisiunile să fie la nivelul minim necesar pentru utilizarea aplcaţiei conform modelului de cazuri de utilizare.

- *Hackerii şi crackerii*[1] - persoanele care doresc să obţină profit sau date preţioase vor încerca mereu să strice aplicaţia, problema nu poate fi înlăturată complet, dar procesul poate fi făcut mai complex prin îndeplinirea unor norme de securitate mai avansate.

- *Lipsa obfuscării sau criptării datelor* - datele stocate în formă citibilă creează o facilitate pentru atacatori, aceştia nu mai au nevoie de muncă suplimentară pentru obţinerea datelor valoroase odată ce au obţinut control asupra sistemului.

Măsurile esenţiale pentru securitate vor fi discutate în secţiunile ce urmează: criptografia, compresia şi steganografia.

3.4.2 Criptografia

Criptografia[2] - este utilizată pentru obfuscarea şi ascunderea mesajelor. Există numeroase metode de criptare a datelor începând cu Cifrul lui Caesar, una dintre cele mai primitive metode de criptare existente, şi terminând cu AES[3] şi RSA[4], care sunt metodele de criptare standardizate, considerate aproape invincibile în momentul de faţă. Totuşi criptografia nu este o soluţie generală pentru securitate, aceasta trebuie privită mai mult ca o ustensilă. Adversarul principal al criptografiei este - criptanaliza, ştiinţa destinată descifrării mesajelor criptate prin analiza datelor de intrare şi ieşire ale unui algoritm[13].

Pentru criptarea şi decriptarea fişierelor a fost selectat algoritmul *twofish*[14]. Acesta este un cifru cu cheie simetrică, blocurile de date au o lungime de până în 128 biţi, iar cheile utilizate pot avea lungimi de până la 256 biţi. Algoritmul a participat în concursul AES, a fost printre ultimii finalişti, dar nu a fost selectat pentru standardizare. Eficienţa şi siguranţa acestui algoritm este apropiată de cea a actualului AES, acesta fiind mai încet pentru chei de 128 de biţi, dar mai rapid pentru chei de 256 de biţi. Algoritmul se bazează pe *reţele Feistel*, operaţia fundamentală a algoritmului este maparea datelor de intrare în funcţie de cheie în date de ieşire, funcţia fiind non lineară şi non-surjectivă. Un alt element cheie al algoritmului este *S-box*[5] - acesta constă dintr-un set de substituţii non-lineare folosite în majoritatea cifrurilor pe blocuri. Alte operaţii folosite sunt mapările

[1] https://www.docsity.com/en/hackers-and-crackers/4166947/
[2] https://cryptography.io
[3] Advanced Encryption Standard
[4] Rivest–Shamir–Adleman
[5] https://www.sciencedirect.com/topics/mathematics/s-box

cu ajutorul matricilor MDS[6], transformările Pseudo-Hadamard şi operaţia fundamentală XOR. În Figura 3.4 este reprezentată o rundă din algoritm. Avantajele principale ale algoritmului sunt: *performanţa*, *flexibilitatea*, *simplicitatea* şi *eficienţa*.

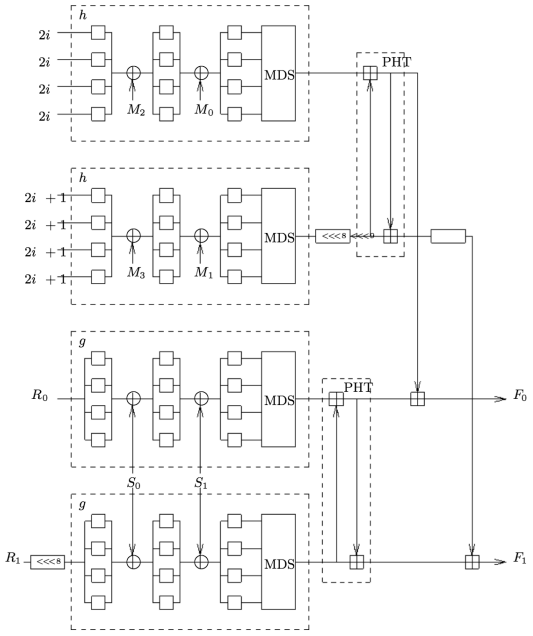

Figura 3.4: Runda de criptare TwoFish

Algoritmul selectat pentru criptarea parolelor este **Argon2**[15]. Acest algoritm a fost ales câştigătorul competiţiei *Password Hashing Competition*[7] în Iulie 2015. Există 3 versiuni al acestui algoritm:

- *Argon2d* maximizează rezistenţa la atacurile care folosesc GPU. Acesta accesează memoria într-o ordine care depide de caracterele parolei, şi reduce riscul atacurilor cauzate de dependenţele memorie-timp
- *Argon2i* reduce riscul atacurilor bazate pe canale auxiliare.
- *Argon2id* versiune hibridă a celor 2 menţionate anterior, are avantajele ambelor versiuni şi este recomandată de a fi utilizată.

Toate 3 versiuni permit configurarea a 3 parametri de control: timp de execuţie, volum de memorie cerută şi gradul paralelismului.

[6]https://cryptography.fandom.com/wiki/MDS_matrix
[7]https://password-hashing.net

Nu exista date oficiale sau publice cu referire la atacuri soldate cu succes asupra Argon2d sau Argon2di. Spre deosebire de algoritmii similari, cum ar fi bcrypt sau PBKDF2, argon2 este foarte eficient, foarte sigur şi simplu de configurat.

3.4.3 Compresia

În procesarea de date, compresia[16] este o operaţie care presupune encodarea datelor utilizând mai puţini biţi decât în setul de date iniţial. Compresia poate fi de 2 tipuri: *cu pierderi* sau *fără pierderi*. Compresia fără pierderi reduce numărul de biţi prin identificarea şi eliminarea redundanţei statistice. Compresia cu pierderi se bazează pe reducerea dimensiunii datelor prin eliminarea informaţiei redundante sau nesemnificative, la efectuarea procesului invers datele nu vor fi egale cu cele iniţiale.

Avantajele compresiei sunt[17]:

- **Reducerea dimensiunii datelor** - este utilă la transmitere şi stocare, deoarece reduce timpul de procesare şi scade costul de stocare
- **Reducerea entropiei datelor** - în cazul criptării, procesul de atacuri de forţă brută devine mai complicat şi nu se mai poate face analiză statistică a datelor.

Unul dintre cei mai eficienţi algoritmi de compresie este *zlib*[8]. Acesta este un algoritm open-source, nepatentat, disponibil pe toate platformele şi pentru diverse tipuri de date, de la text la video.

Compresia a forst şi rămâne a fi centrul unei schimbări masive în ingineria calculatoarelor. Motivul fiind o simplă teorie economică: fişierele compresate sunt fişiere mai mici, datorită acestui fapt: sunt mai uşor de transmis, mai uşor de prelucrat, mai uşor de stocat şi, desigur, mai ieftine. Distribuitorii plătesc mai puţin să le distribuie, iar clienţii plătesc mai puţin să le consume. În lumea în care calculele sunt echivalente cu banii, compresia reprezintă cea mai viabilă soluţie economică.

3.4.4 Steganografia

Steganografia[18] este o tehnică de protecţie a datelor, aceasta oferă soluţii eficiente pentru controlul accesului la date private. Datele din imagini sunt disponibile doar pentru oamenii care cunosc modul de extragere a lor.

Steganografia bazată pe modificarea celui mai puţin semnificativ bit este o metodă simplă şi eficientă de encodare a informaţiilor ascunse. Vizual nu se observă diferenţa dintre imaginea iniţială şi imaginea care conţine date encodate în cel ai nesemnificativ bit. La o analiză statistică se pot decoda datele şi algoritmul nu mai este fezabil. Pentru a obţine o securitate mai ridicată, a se utilizează o metodă în care nu se ascund datele în intreaga imagine, ci doar în pixelii de pe margine. Pentru complicarea analizei statistice se adaugă encodarea datelor în zig-zag.

[8]https://zlib.net

De-a lungul timpului s-au dezvoltat foarte multe metode de steganografie, unele eficiente, altele mai puţin eficiente, dezavantajul cel mai mare al steganografiei rămâne faptul că odată ce metoda a fost descoperită, aceasta nu mai poate fi considerată de încredere. O soluţie de creştere a eficienţei este criptarea mesajului înainte de encodare.

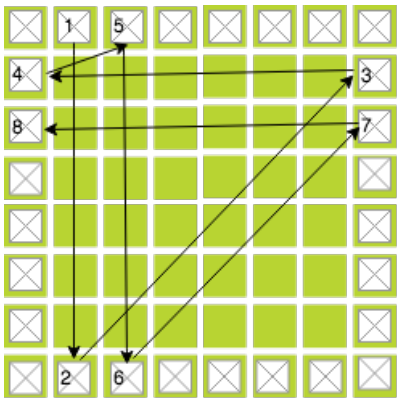

Figura 3.5: Exemplu de encodare a unui byte

După cum se poate observa din Figura 3.5, pentru encodarea unui octet se execută 8 paşi, biţii sunt encodaţi alternativ în marginile imagini. Această metodă nu poate fi considerată 100% sigură, însă reduce riscul descoperirii datelor în cazul unei analize statistice clasice, datele se pot afla doar ştiind ordinea exactă de codificare.

3.5 Sisteme similare

Acest capitol reprezintă clasificarea si analiza sistemelor similare existente, bazată pe etapa de cercetare a proiectului. Sistemele au scop şi functionalităţi similare cu proiectul propus. Sistemele alese pentru comparaţie sunt:

- CloudMe
- Dropbox
- CrashPlan
- ICloud
- Google Drive
- OneDrive
- pCloud

- sync.com

Dropbox, Google Drive, ICloud si OneDrive au fost incluse în acest studiu deoarece sunt în top 10 cele mai populare servicii de cloud 2019 [19]. Acestea reprezintă un model pentru cum este văzut un cloud storage modern: simplu de configurat, simplu de utilizat şi disponibil la un preţ avantajos. pCloud si sync.com sunt în topul sistemelor cu cea mai înaltă recuritate de pe piaţă. pCloud este categorizat ca un sistem infraudabil şi nu a avut nici o expunere a datelor utilizatorilor. Însă aceste sisteme vin si cu preţuri de 3-4 ori mai mari decât sistemele clasice.

3.5.1 Metodologia de analiză

În ultimul deceniu tot mai mulţi utilizatori, atât business cât şi individuali, se basează pe stocarea fişierelor în Cloud. Cele mai importante criterii pe care se bazează utilizatorii sunt: securitatea, simplitatea de utilizare a sistemului, disponibilitatea si preţul de utilizare. Analiza sistemelor individuale de cloud a fost efectuată în modul următor:

Sunt sumarizate preţurile de utilizare a sistemului şi a diferitor opţiuni. Sunt analizate detaliile capabilităţilor tehnice şi organizaţionale ale părţilor client şi server ale sistemului. Informaţiile colectate sunt bazate pe secţiunile *Terms of Service* şi *Privacy Policy* ale documentaţiilor oficiale ale sistemelor[20]. Rezultatele analizei comparative au ca scop determinarea cerinţelor pricipale ale unui sistem de stocare cloud şi comparaţia sistemului elaborat cu cele existente. În secţiunile ce urmează se vor analiza următoarele funcţionalităţi:

Tabelul 3.1: Criterii evaluare sisteme similare

Copy	Backup	Sync	Sharing	Client-side Encryption	Server-side encryption	Compression	Watermarking

Pentru fiecare categorie din tabelul 3.1 se va acorda un punctaj conform următoarelor reguli:

✓✓ este echivalent pentru *foarte bine*, toate cerinţele obligatorii pentru funcţionalitatea respectivă au fost îndeplinite şi câteva dintre cele opţionale.

✓ este echivalent pentru *bine*, toate cerinţele obligatorii pentru funcţionalitatea respectivă au fost îndeplinite.

± este simbolul pentru *bine cu câteva vulnerabilităţi*, nu toate cerinţele esenţiale au fost îndeplinite.

✗ este echivalent cu *slab*, cel puţin o cerinţă obligatorie nu a fost îndeplinită.

✗✗ este echivalentul pentru *foarte slab*, adică mai multe dintre cerinţele obligatorii nu sunt indeplinite în funcţionalitatea respectivă sau funcţionalitatea lipseşte integral.

3.5.2 CloudMe

CloudMe[21] este un sistem de cloud standard, care vine cu un serviciu de sincronizare şi backup a fişierelor, după o analiză detaliata ne putem da seama că acest sistem a fost inspirat din arhicunoscutul **Dropbox** care este analizat în secţiunea 3.5.3.

În tabelul 3.2 sunt prezentate funcţionalităţile sistemului şi o notă a fiecărei funcţionalităţi în concordanţă cu evaluările primite pe pagina oficială a sistemului, precum şi a vulnerabilităţilor depistate în ultima perioadă.

Tabelul 3.2: CloudMe Funcţionalităţi

Copy	Backup	Sync	Sharing	Client-side Encryption	Server-side encryption	Compression	Watermarking
Da ✓	Da ✓	Da ✗✗	Da ±	Da ✓✓	Nu ✗✗	Nu ✗✗	Nu ✗✗

Tabelul 3.3 prezintă disponibilitatea **CloudMe** pe diferite platforme şi pretul acestuia.

Tabelul 3.3: CloudMe Platforme disponibile şi preţ

Web Client	Desktop client	Mobile Client	500GB Plan Price
Da	Da	Da	10€

Un fapt bun despre **CloudMe** este că acesta pune la dispoziţia utilizatorului un spaţiu de stocare gratuit de 3GB, iar pentru volume de date mai mari oferă opţiuni la preţul mediu al pieţei. Serviciul oferă funcţionalităţile de bază, însă nimic desebit în materie de securitate.

Interfaţa web a **CloudMe** este foarte simplă si clară, este evident cum sa încarci un fisier sau cum sa îl schimbi în alt folder.

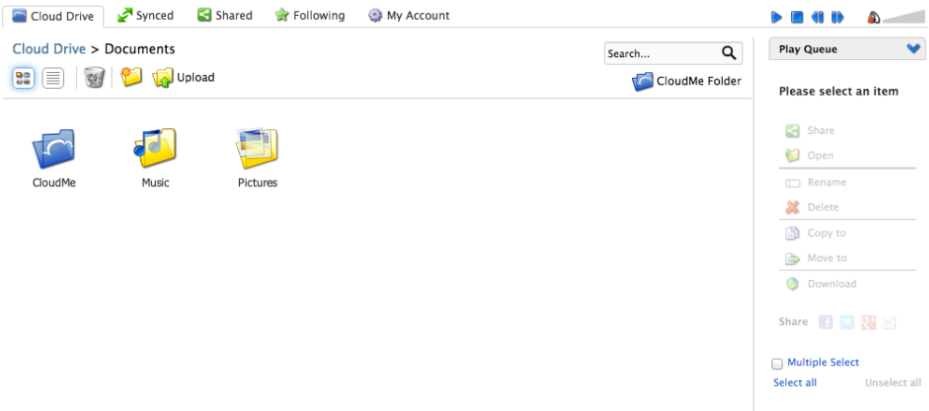

Figura 3.6: CloudMe - interfaţa web

CloudMe oferă sincronizarea aplicaţiilor pentru Windows, Mac, Linux, iOS şi Android. Pentru ca sincronizarea să poată avea loc utilizatorul primeşte un aşa numit "director albastru" care oferă utilizatorului sincronizare în timp real. De asemenea, **CloudMe** oferă opţiunea de a alege timpul la care se va întâmpla sincronizarea. Pentru a distribui fişiere sunt disponibile mai multe opţiuni, începând cu distribuire clasică, care permite utilizatorului să ofere acces la fişierele lui prin distribuirea unui link, şi ajungând la metode mult mai colaborative care le permit altor utilizatori, cărora li se oferă access, să modifice fişierele din cloud-ul altui utilizator sau să încarce fişiere noi.

Figura 3.7: CloudMe - planuri de preţ

CloudMe arhivează versiunile precedente ale fişierelor prin funcţionalitatea coşului de gunoi, care păstrează pentru 60 de zile toate fişierele care au fost şterse.

În privinţa securităţii **CloudMe** nu este o opţiune prea bună deoarece nu oferă criptarea datelor, acestea fiind vulnerabile pentru atacuri. Este posibil să încarci şi să descarci fişiere criptate, însă criptarea şi decriptarea acestora ramâne la latitudinea utilizatorului.

CloudMe este o aplicaţie foarte uşor de utilizat şi are o interfaţă foarte intuitivă. Acesta dispune de funcţionalităţile de bază ale unui sistem de stocare în cloud, însă nu este potrivit pentru stocarea fisierelor cu conţinut de date senzitiv din cauza lipsei criptării, de asemnea atunci căND fişierele sunt distribuite după un link este foarte greu să determini cine a făcut public un fişier cu caracter privat deoarece link-ul de acces poate fi uşor furat.

Un alt defect al acestui sistem este funcţionalitatea de *Sync*, motivul pentru care i-am oferit punctaj minim este că în ultimul an a avut multiple vulnerabilităţi, conform *CVE-2018-6892* [9] atacatorii se puteau conecta la clientul de "CloudMe Sync" prin portul 8888 şi trimiterea unor date maliţioase puteau cauza "buffer overflow", acest lucru le oferea control asupra execuţiei şi posibilitatea de a executa cod maliţios.

3.5.3 Dropbox

Dropbox a fost lansat in 2007 şi este definit ca unul dintre cele mai bune servicii de cloud pentru uz general[22]. Sistemul are peste 500 de milioane de utilizatori în toată lumea, fiind unul dintre cele mai competitive servicii de pe piaţă.

[9]https://nvd.nist.gov/vuln/detail/CVE-2018-6892

În tabelul 3.4 este prezentată o evaluare a sistemului pe baza unei analize de utilizare, dar şi pe baza evaluărilor oferite de *CloudWards*[22] şi *On the Security of Cloud Storage Services*[20].

Tabelul 3.4: Dropbox Funcţionalităţi

Copy	Backup	Sync	Sharing	Client-side Encryption	Server-side encryption	Compression	Watermarking
Da ✓	Da ✓	Da ✓	Da ✓✓	Nu ✗✗	Da ±	Nu ✗✗	Nu ✗✗

În tabelul 3.5 sunt prezentate opţiunile de clienţi disponibili pentru **Dropbox** şi oferta de preţ.

Tabelul 3.5: Dropbox Platforme disponibile şi preţ

Web Client	Desktop client	Mobile Client	500GB Plan Price
Da	Da	Da	18€

Dropbox este uşor de utilizat atăt folosind aplicaţia web, cât şi cele mobile sau desktop. În Figura 3.8 este prezentată interfaţa web a sistemului.

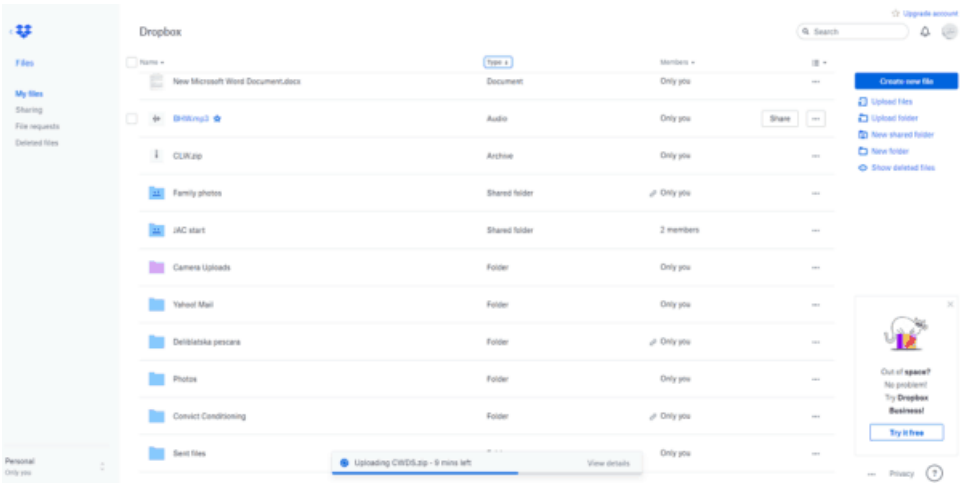

Figura 3.8: Dropbox - interfaţa web

Dropbox oferă un serviciu de sincronizare ”rapidă şi inteligentă”, cu diverse posibilităţi de customizare, însă această opţiune poate fi aplicată doar pe anumite directoare, lucru care restrânge libertatea utilizatorului.

Funcţionalitatea de partajare a fişierelor este în topul celor disponibile pe piaţă, deoarece oferă partajare atât de fişiere cât şi de directoare, cu parolă sau fără, cu

customizare de permisiune şi termen de expirare. Se pot trimite link-uri pe mail sau prin copierea directă, link-urile pot fi şterse şi fişierul nu mai este accesibil prin acel link. Această funcţionalitate este disponibilă atât din orice aplicaţie **Dropbox**.

O funcţionalitate a cărui autor este **Dropbox** este deduplicarea la nivel de bloc, prin împărţirea fişierului la încărcare în multiple porţiuni de dimensiune fixă şi prin scanarea dacă porţiunea există, acest lucru oferă o viteză de încărcare mai mare, dar prezintă un risc deoarece un atacator poate determina conţinutul unui fişier de anumit format prin încărcări repetate de conţinut diferit a anumitor porţiuni, de exemplu un fişier cu analize medicale.

La nivel de securitate **Dropbox** nu este cea mai bună soluţie, având un trecut destul de bogat în atacuri, suferind numeroase furturi de date. **Dropbox** salvează fişierele în format criptat, însă numeroase metadate care includ şi porţiuni de text sunt salvate in format original. Acest lucru nu este benefic pentru utilizatori deoarece datele lor pot fi compromise.

Dropbox a avut o evoluţie specaculoasă în ultimii 6 ani, adaugând numeroase funcţionalităţi şi devenind mult mai uşor de utilizat. Însă **Dropbox** rămâne o soluţie pentru utilizatorii care nu au date senzitive stocate în acest sistem. Există mai multe variante de compromitere a datelor, una dintre acestea este cauzată de implementarea FTS(Full-Text-Search)[10], prin păstrarea metadatelor pentru căutare în format necriptat. Altă vulnerabilitate este cauzată de deduplicarea la nivel de bloc ce se execută la nivel de întreg sistem în loc de nivel fişiere per utilizator.

O vulnerabilitate de securitate a fost descoperită în 2018 *CVE-2018-12271*[11] atunci când un atacator se putea loga pe un cont **Dropbox** cu orice amprentă arbitrară şi avea access la toate fişierele utilizatorului, un nivel suplimentar de securitate prin criptare cu cheie provenită de la utilizator ar fi prevenit acest lucru.

3.5.4 CrashPlan

CrashPlan este un sistem de stocare a fişierelor şi backup care a apărut pe piaţă în 2007. În prezent, sistemul a devenit unul foarte popular, zilnic acesta procesează peste 100 de miliarde de fişiere.

În tabelul 3.6 sunt prezentate funcţionalităţile sistemului şi o evaluare efectivă a acestor capabilităţi.

Tabelul 3.6: CrashPlan Funcţionalităţi

Copy	Backup	Sync	Sharing	Client-side Encryption	Server-side encryption	Compression	Watermarking
Da ✓	Da ✓	Nu ✗✗	Nu ✗✗	Da ✓✓	Da ✓✓	Nu ✗✗	Nu ✗✗

[10] https://www.techopedia.com/definition/17113/full-text-search
[11] https://www.cvedetails.com/cve/CVE-2018-12271/

Din evaluarea de mai sus se observă că sistemul nu oferă partajare de fişiere, în realitate această funcţionalitate a fost înlăturată recent, cauza fiind riscul de compromitere a datelor. De asemenea partajarea fişierelor şi menţinerea unui nivel de securitate crescut implică un efort considerabil, de aceea, pentru moment, serviciul a fost deactivat.

În tabelul 3.7 sunt prezentate platformele pe care este disponibil sistemul, se poate observa că preţul este mai mic comparabil cu adversarii analizaţi în secţiunile precedente, acest fapt se poate datora lipsei unor anumite funcţionalităţi.

Tabelul 3.7: CrashPlan Platforme disponibile şi preţuri

Web Client	Desktop client	Mobile Client	500GB Plan Price
Da	Da	No	10€

CrashPlan a fost caracterizat ca unul dintre cele mai bune sisteme pentru backup[23] datorită factorului că nu are dimensiune maximă a fişierelor pentru backup. Crashplan nu necesită implicare din partea utilizatorului pentru a executa copierea regulată a fişierelor. De asemenea, CrashPlan permite executarea operaţiunii de backup în acelaşi cont de utilizator a până la 10 calculatoare.

CrashPlan oferă mai multe nivele de securitate pentru fişiere, toate datele sunt criptate de la client până la server. Criptarea se execută utilizând o cheie de 448 biţi pentru utilizatorii unui plan plătit, pentru utilizatorii opţiunii gratuite se utilizează o cheie de 128 biţi. Cheile de criptare sunt generate utilizănd un sistem eficient de numere aliatoare. De asemena există opţiunea de a selecta o cheie privată de criptare, acesta nu va fi salvată niciodata sub formă de text şi nu va fi accesibilă nimănui.

Cel mai important lucru care face **CrashPlan** un sistem extrem de bun este performanţa şi eficienţa criptării care oferă o securitate excepţională a datelor, acestea nu pot fi decriptate fără cunoştinţa şi acordul utilizatorului. Datele utilizatorilor au fost compromise o singură dată, din cauza unui vulnerabilităţi ce permitea executarea codului la distanţă, acest lucru a devenit posibil din cauza unei vulnerabilităţi din clasa Java *DateRMI*, descrierea vulnerabilităţii poate fi găsită în *CVE-2017-9830* [12].

3.5.5 ICloud

ICloud este unul dintre cele mai populare şi utilizate sisteme de cloud conform CloudWards [24], acest fapt nu este datorat doar poziţiei monopolistice pe care o ocupă pe piaţă, ci şi funcţionalităţilor pe care le oferă.

Sistemul **ICloud** este preinstalat pe toate dispozitivele Apple, acest lucru este deseori privit ca motivul pentru care este atât de popular. Totuşi, conform analizei funcţionalităţilor, care este prezentată în tabelul 3.8 se poate observa că funcţionalităţile acestuia se ridică la un nivel destul de înalt.

[12] https://www.cvedetails.com/cve/CVE-2017-9830/

Tabelul 3.8: ICloud Funcţionalităţi

Copy	Backup	Sync	Sharing	Client-side Encryption	Server-side encryption	Compression	Watermarking
Da ✓	Da ✓	Da ✓✓	Da ±	Da ✓✓	Nu ✗✗	Nu ✗✗	Nu ✗✗

În tabelul 3.9 sunt prezentate platformele pe care este disponibil **ICloud**.

Tabelul 3.9: ICloud Platforme disponibile şi preţuri

Web Client	Desktop client	Mobile Client	500GB Plan Price
Da	Da	Da	5€

Însă acesta are şi câteva probleme legate de clientul Desktop, care are funcţionalităţi limitate, şi limitarea funcţionalităţii de partajare de fişiere.**ICloud** oferă funcţionalitatea de sincronizare cu Apple Photos. Iar funcţionalitatea de share poate fi accesată din orice fişier care se află într-un director sincronizat.

În 2014 a avut loc un furt de date de dimensiuni foarte mari, acesta a compromis reputaţia ICloud, însă atacul a fost făcut prin forţă brută şi "pescuirea datelor" de la viitoarele victime. În realitate apple oferă câteva funcţionalităţi de securitate care îl fac un sistem cu nivel de securitate peste media de pe piaţă. Însă Apple nu este un sistem cu "cunoştinţă zero", acesta stochează cheile de criptare în acelaşi loc cu fişierele criptate, asta íl face extrem de vulnerabil în cazul unui atac.

Spre deosebire de alte sisteme, cum ar fi *Google*3.5.6, este bine cunoscut că *Apple* nu colaborează cu guvernul sau companiile publicitare şi nu va oferi informaţii private despre clienţii săi.

ICloud nu este o soluţie genrală pentru stocarea fişierelor, acesta este potrivit pentru utilizatorii care nu au nevoie să păstreze date extrem de sensibile din cauza posibilităţii de decriptare prin brute force. Acesta nu este potrivit nici pentru utilizatorii care doresc o viteză ridicată de încărcare şi decărcare a fişierelor, însă Apple continuă să se perfecţioneze şi să crească viteza operaţiilor de reţea.

De asemenea iCloud este extrem de vulnerabil pentru executare de cod la distanţă conform datelor oferite de CVEDetails.com, vulnerabilităţile recent descoperite sunt *CVE-2018-20506*[13] şi *CVE-2018-4464*[14], acest fapt este datorat popularităţii sistemului crae îl face o ţintă importantă pentru hackeri.

3.5.6 Google Drive

Cu aproximativ un miliard de utilizatori, **Google Drive** este cel mai popular serviciu de *cloud* de pe piaţă, această popularitate nu este datorată doar faptului că este

[13]https://www.cvedetails.com/cve/CVE-2018-20506/
[14]https://www.cvedetails.com/cve/CVE-2018-4464/

preinstalat pe telefoanele Android, dar şi capabilităţilor de partajare şi vitezei înalte de decărcare şi încărcare a fişierelor.

În tabelul 3.10 sunt prezentate evaluări ale funcţionalităţilor sistemului.

Tabelul 3.10: Google Drive Funcţionalităţi

Copy	Backup	Sync	Sharing	Client-side Encryption	Server-side encryption	Compression	Watermarking
Da ✓	Da ✓✓	Da ✓	Da ✓	Da ✓✓	Da ✓	Da ±	Nu ✗✗

Deşi **Google Drive** oferă criptarea datelor, securitatea si intimitatea nu sunt punctele forte ale sistemului, acesta având antecedente de implicare în campaniile miltare de colectare a datelor şi spionării cetăţenilor. Compresia datelor, pe de altă parte, se referă la reducerea dimensiunii imaginilor, proces de compresie cu pierderi.

În tabelul 3.11 sunt prezentate opţiunile de client disponibile pentru **Google Drive**, de asemenea fiecare utilizator primeşte iniţial 10 GB de stocare gratuită.

Tabelul 3.11: Google Drive Platforme Disponibile şi Preţuri

Web Client	Desktop client	Mobile Client	500GB Plan Price
Da	Da	Da	15€

Google Drive este unul dintre cele mai bune sisteme pentru colaborare, ocupând locul 2, după Dropbox, prezentat în secţiunea 3.5.3. Punctul forte al colaborării oferite de Google Cloud este faptul că Office Suite este integrat în acest sistem.

Funcţionalitatea de sincronizare este extrem de performantă, dar nu oferă sincronizare la nivel de bloc de fisier. Partajarea de fişiere este una dintre cele mai folosite funcţionalităţi ale sistemului, însă acesta vine cu câteva vulnerabilităţi cauzate de lipsa criptării la partajare sau protejarea cu parolă. Partajarea este uşor de executat link-ul poate fi partajat prin email, Facebook, Twitter sau copiat şi salvat în destinaţia dorită.

Datele sunt criptate utilizând AES-128 atunci când sunt salvate pe disc şi prin TSL atunci când sunt transportate între client şi server. Oricum sistemul nu oferă ”zero-knowledge” şi oricine deţine are control asupra sistemului poate să citească datele (exemplu: programatorii sistemului). Totuşi pentru o protecţie mai bună Google oferă posibilitatea de autentificare în 2 paşi.

Google Drive vine cu foarte multe aplicaţii integrate şi posibiltăţi de colaborare integrate în sistem. Autentificare în 2 paşi, criptare a datelor şi viteză ridicată de încărcare şi descărcare, dar lipsa posibilităţii de criptare privată şi vulnerabilităţile cauzate de lipsa unei criptări destul de sigure pentru funcţionalitatea de partajare de fişiere fac **Google Cloud** sa nu fie cea mai bună soluţie pentru securitatea şi integritatea datelor.

3.5.7 OneDrive

OneDrive este un sistem de stocare în cloud dezvoltat de compania MIcrosoft, are un trecut destul de ambiguu în privinţa securităţii datelor, deşi şi-a perfecţionat securitatea, nu poate fi încadrat în topul celor mai sigure sisteme de securitatea[25], însă tinde spre acesta.

Analiza funcţionalităţilor sistemului este prezentată în tabelul 3.12.

Tabelul 3.12: OneDrive Funcţionalităţi

Copy	Backup	Sync	Sharing	Client-side Encryption	Server-side encryption	Compression	Watermarking
Da ✓	Da ✓✓	Da ✓	Da ✓	Nu ✗✗	Da ✗✗	Nu ✗✗	Nu ✗✗

Funcţionalităţile nu au primit o notă maximă din cauza ambiguităţii de utilizare a sistemului, în dependenţă de tipul fişierului selectat, meniurile arată diferit, uneori poate fi o problemă pentru utilizatori. De asemenea, sistemul nu este unul cu cunoştinţe zero, atacatorul poate obţine datele unui utilizator, odată ce s-a infiltrat în sistem, deoarele toate datele necesare pentru decriptare sunt deja prezente în sistem.

În tabelul 3.13 sunt prezentate datele cu privire la disponibilitatea sistemului pe diferite platforme, dar şi preţul unui abonament cu spaţiu de stocare de 500GB.

Tabelul 3.13: OneDrive Platforme Disponibile şi Preţuri

Web Client	Desktop client	Mobile Client	500GB Plan Price
Da	Da	Da	15€

OneDrive urmează nişte principii definite de Dropbox cu privire la standardele de sincronizare a fişierelor. OneDrive obişnuia să aibă probleme serioase cu privire la securitate, neavând nici criptare la nivelul nivelului de stocare. În prezent, ânsă sistemul oferă criptare la transmitere şi la stocare. Criptarea nivelului de stocare include 2 componente esenţiale: *BitLocker*, criptare la nivel de disc, şi un sistem de criptare per fişier a conţinutului. Fiecare fişier este securizat priin utilizarea unei chei AES unice de 256 de biti, de asemenea se foloseşte protocolul TLS pentru a preveni atacurile de tipul *man-in-the-midle* Dar minusul acestui sistem este că cheile sunt stocate pe acelaşi sistem, orice angajat poate eventual să citească datele utilizatorilor sau să le ofere companiilor de e-publicitate sau unor servicii secrete.

În concluzie, **OneDrive** este un sistem care a evoluat considerabil în ultima perioadă, însă are neajunsuri considerabile pe partea de securitate. De asemenea sistemul nu oferă funcţionalitate de comopresie, ceea ce creşte costulm de stocare a datelor, datele pot fi compresate de utilizator în prealabil, iar după descărcare acestea pot fi decompresate, operaţia însă este anevoioasă şi prezintă o problemă în cazul partajării fişierelor. Sistemul nu este potrivit pentru stocarea unor date sensibile şi este recomandat doar pentru stocarea unor date netriviale.

3.5.8 pCloud

pCloud a fost fondat în 2013 şi în doar 3 ani a ajuns la peste 3 milioane de utilizatori, competitorii cei mai importanţi sunt Dropbox şi Copy. Chiar dacă este nou pe piaţă, spre deosebire de competitorii săi mari, acesta oferă funcţionalităţi de top şi este inclus în topul *Most Secure Cloud Storage 2019: Safety First*[25].

pCloud oferă funcţionalităţile de sincronizare, backup (se opoate aplica şi pe datele de pe Instagram sau Facebook), colaborare şi criptare avansată a datelor. O evaluare a acestor capabilităţi este oferită în tabelul 3.14.

Tabelul 3.14: Funcţionalităţi

Copy	Backup	Sync	Sharing	Client-side Encryption	Server-side encryption	Compression	Watermarking
Da ✓	Nu ✗✗	Da ✓	Da ✓	Da ✓✓	Da ✓✓	Nu ✗✗	Nu ✗✗

Sistemul este prezent pe toate tipurile de platforme, tabelul 3.15 şi are un preţ destul de ridicat, însă rezonabil pentru un sistem securizat de stocare.

Tabelul 3.15: Sisteme de operare

Web Client	Desktop client	Mobile Client	500GB Plan Price
Da	Da	Da	40€

Securitatea este unul dintre punctele forte ale sistemului, dezvoltatorii încearcă să ofere servicii la cel mai înalt nivel şi sa ofere funcţionalităţi care nu sunt prezente la competitori. Sistemul foloseşte chei de 256 biti şi TSL pentru transmisia de date. Securitatea însă vine şi cu un preţ, fişierele nu pot fi modificate în sistem, deoarece operaţiile de criptare şi decriptare ar trebui efectuate repetat, provocând costuri de prelucrare ridicate. De asemenea, sediul central al companiei se află în Elveţia, unde sunt cele mai stricte reguli de protecţie a datelor, astfel utilizatorul poate fi liniştit cu privire la integritatea datelor sale.

Datele încărcate sunt repartizate în funcţie de tipul de date: imagini, documente, muzica şi video.

Un dezavantaj al sistemului este că criptarea este o funcţionalitate care nu este oferită în pachetul de bază şi are un preţ mai ridicat, comparativ cu alte sisteme care oferă criptarea ca serviciu gratuit.

3.5.9 sync.com

Sync.com a fost fondat în 2011 de către Suhan Shan, Thoman Savundra, şi Darius Antia. Acesta a devenit deja un competitor serrios pentru Google Drive si Dropbox. Caracteristicile pentru care a ajuns atât de popular sunt analizate în tabelul 3.16.

Tabelul 3.16: Funcţionalităţi

Copy	Backup	Sync	Sharing	Client-side Encryption	Server-side encryption	Compression	Watermarking
Da ✓	Da ✓✓	Da ✓	Da ✓	Da ✓✓	Da ✓✓	Nu ✗✗	Nu ✗✗

Din păcate sistemul nu oferă compresia sau deduplicarea datelor, sistemul stocând volumul real de date pe care îl primeşte de la utilizator, plus câteva metadate create de algoritmii de compresie.

Tabelul 3.17: Sisteme de operare

Web Client	Desktop client	Mobile Client	500GB Plan Price
Da	Da	Da	15€

Criptarea datelor face ca sistemul să nu mai poată interpreta datele stocate, astfel sistemul nu oferă posibilitatea de deschidere şi vizualizare a fişierelor, doar decărcarea şi manipularea lor ulterioară cu alte aplicaţii. Însă există şi opţiunea de a stoca fişierele fără criptare. Caracteristicile cheie ale sync.com sunt :

- Cunostinţe zero despre date
- Criptare privată
- Uşor de utilizat
- Syncronizare
- Partajare de fişiere cu control asupra operaţiei

Sync este unul dintre sistemele care garantează securitate la nivelul cel mai înalt, acest sistem foloseşte pentru criptare RSA cu chei între 512 si 2048 biti. Sync nu păstrează cheile de criptare în sistem, daca utilizatorul îşi uită parola atunci datele lui nu vor putea fi recuperate niciodată. De asemenea se poate activa şi opţiunea de autentificare în doi paşi, pentru a oferi o protecţie şi mai bună.

Sistemul oferă funcţionalitatea de sincronizare, însă nu orice director poate fi selectat pentru efectuarea operaţieidin cauza faptului că se iau în considerare particularităţile fiecărei aplicatii. Sync ofera funcţionalitatea de partajare de fişiere care poate fi configurată adăugând parolă sau timp de expirare.

Nu putem nega că sync.com oferă o funcţionalitate de criptare complexă şi eficientă care oferă o securitate avansată, dar, partea negativă a acestui lucru este că atunci când dorim să vizualizăm un fişier sau săîl descărcăm, operaţia ar putea dura între câteva secunde şi cateva minute din cauza complexităţii adăugate de criptare.

3.5.10 Concluzii şi plasarea sistemului

Volumul de date stocate în cloud a crescut cu un factor de 40 în ultimii 10 ani, creşterea este constantă. Evoluţia tehnologică aduce un preţ mai mic pentru componentele hardware de stocare, dar şi cantităţi mai mari de date ceea ce împiedică scăderea preţului serviciilor. De asemnea, evoluţia tehnologică aduce un impact negativ şi asupra securităţii, un atac de forţă brută poate fi executat mult mai uşor pe un sistem mai performant.

După studiul efectuat, am determinat că nici un sistem nu oferă functionalitatea de compresie de fişiere, cu toate că unele sisteme oferă deduplicare, aceasta vine cu un impact asupra securităţii utilizatorilor. Sistemul propus oferă reducerea volumului de date prin algoritmi de compresie şi decompresie fără pierderi, spre deosebire de Google care efectuează compresia imaginilor în versiunea gratuită prin reducerea dimensiunii, se pierde calitatea imaginii şi este o experientă neplăcută pentru utilizatori. Analiza comparativă este prezentată în tabelul 3.18.

Tabelul 3.18: Comparaţie sisteme similare

Nume	Copiere	Partajare	Criptare la partajare	Criptare pe disc	Algoritmi adaptivi	Compresie	Stegano-grafie
CloudMe	✓	✓	✓	✗	✗	✗	✗
Dropbox	✓	✓	✗	✓	✗	✓	✗
CrashPlan	✓	✗	✗	✓	✗	✗	✗
ICloud	✓	✓	✗	✓	✗	✗	✗
GDrive	✓	✓	✗	✓	✗	✓	✗
OneDrive	✓	✓	✗	✗	✗	✗	✗
pCloud	✓	✓	✓	✓	✗	✗	✗
sync.com	✓	✓	✓	✓	✗	✗	✗
Bitstored	✓	✓	✗	✗	✓	✓	✓

Majoritatea sistemelor oferă criptarea datelor, unele au metode imposibil de spart, altele au metode mai simple şi puţin sigure. Sistemul propus va cripta datele utilizatorului utilizând algoritmi auto-calibrabili care evoluează în funcţie de sistemul pe care ruleaza aplicaţia sau de trecerea timpului. De asemenea sistemul va folosi cei mai noi si siguri algoritmi TwoFish şi PBFK2, care au fost modificaţi să folosească chei mai sigure de o lungime de 256 şi 512 biţi.

Un aport nou pe care îl aduce sistemul şi nu a fost observat la niciuna dintre aplicaţiile studiate este semnătura pe fişiere, pentru a detecta furtul de date prin extragerea codului - tehnica ce va fi folosită este steganografia.

Sistemul elaborat va fi mai mult un prototip, care poate fi dezvoltat, devenind un competitor pentru cele enumerate mai sus. Se pot adăuga funcţionalităţi suplimentare pe partea de partajare de fişiere.

Capitolul 4

Analiză şi Fundamentare Teoretică

4.1 Arhitectura conceptuală a sistemului

Această secţiune reprezinta arhitectura conceptuală a sistemului, prezentată în Figura 4.1.

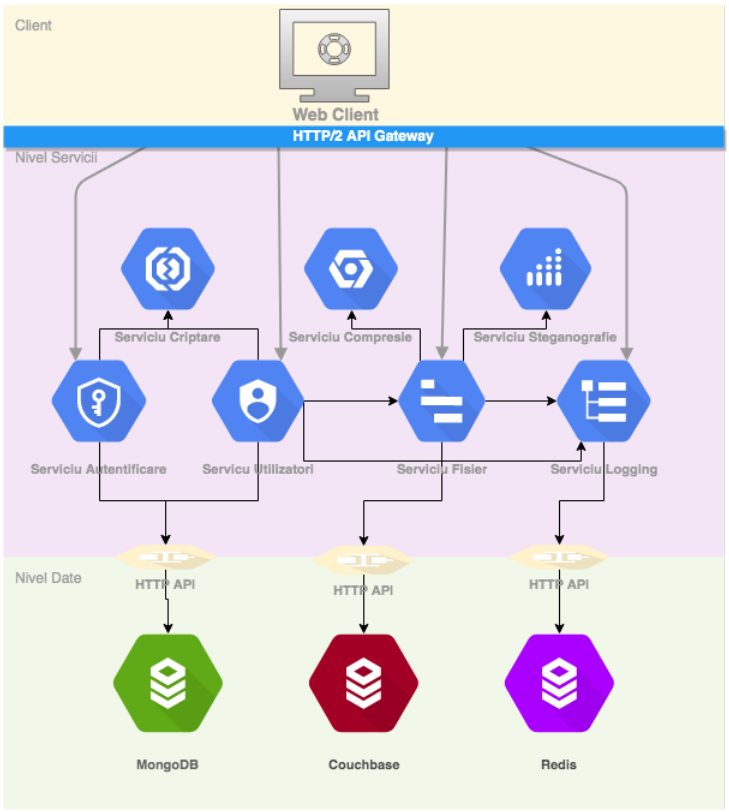

Figura 4.1: Arhitectura sistemului

Sistemul a fost împărţit în microservicii în fucţie de principalele funcţionalităţi ale sistemului. Divizarea sistemului în microservicii contribuie la *scăderea nivelului de cuplare* a componentelor şi *creşterea coeficientului de coeziune*.

Aplicaţia client este de asemenea un microserviciu care asamblează serviciile serverului şi expune funcţionalităţile pentru client. Arhitectura clientului este reprezentată în Figura 4.2.

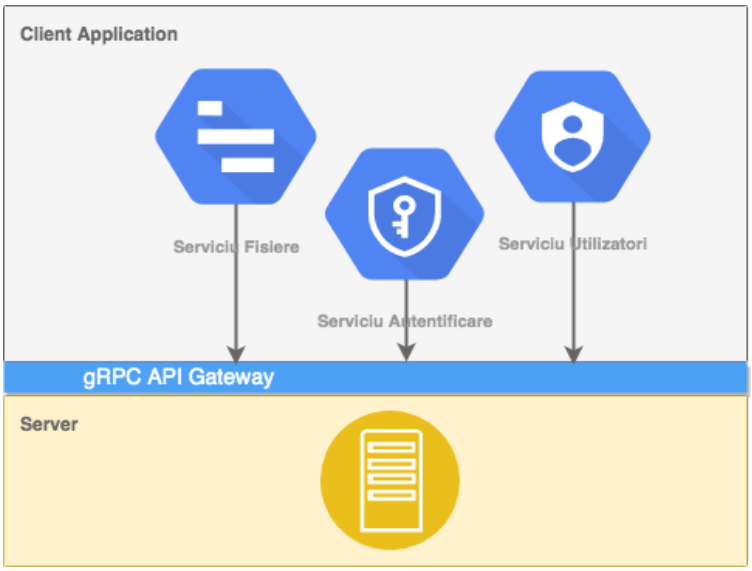

Figura 4.2: Arhitectura clientului

Serviciile din care este format clientul nu interacţionează între ele şi pot fi considerate entităţi separate. Acest lucru contribuie la scăderea numărului de erori cauzate de interacţiunea serviciilor.

4.2 Cazuri de utilizare

În această secţiune sunt descrise cazurile de utilizare ale sistemului, acestea sunt prezentate în Figura 4.3. Au fost identificate 2 tipuri de utilizatori:

- **Utilizator primar** - principalul client al sistemului, are dreptul de a crea, încărca şi descărca fişiere.
- **Administrator** - pe lângă atribuţiile unui utilizator ordinar, are dreptul de a vizualiza toţi utilizatorii primari şi de a le bloca sau debloca contul.

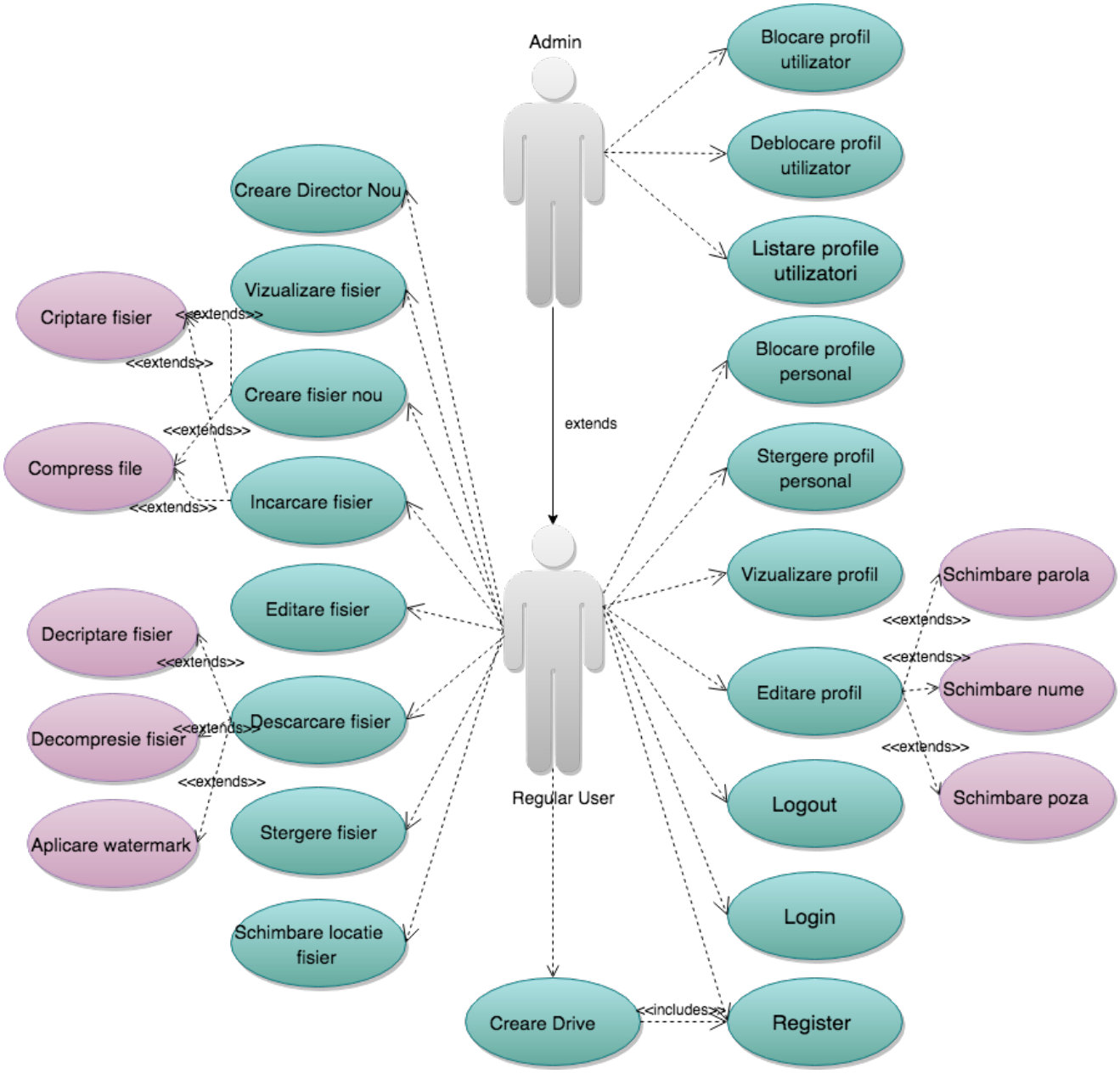

Figura 4.3: Cazuri de utilizare

Din diagramă se pot observa ramificările şi compoziţia din cadrul cazurilor de utilizare stabilite.

4.2.1 CU01 Încărcare fişier

Acest caz de utilizare se referă la încărcarea unui fişier în sistem. Figura 4.4 deprezintă diagrama de activitate pentru acest caz de utilizare.

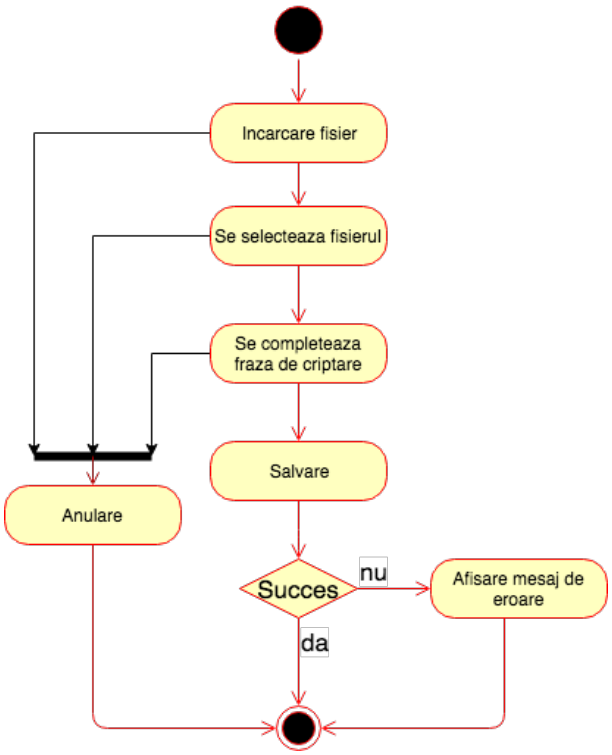

Figura 4.4: Diagrama de activitate ”Incărcare fişier”

4.2.1.1 Precondiţii

- Utilizatorul are un cont activat, care nu este blocat.
- Utilizatorul este autentifiicat.

4.2.1.2 Postcondiţii

- În Drive-ul utilizatorului a fost încărcat un fişier nou, acesta este criptat şi compresat dacă operaţia s-a finalizat cu succes.
- Nu s-a efectuat nici o scriere în cazul anulării operaţiei de către utilizator.

4.2.1.3 Scenariul principal

1. Se apasă butonul de încărcare fişier.

2. Se selectează fişierul care se doreşte a fi încărcat.
3. Se introduce fraza de criptare a fişierului.
4. Se apasă butonul de confirmare a încărcării.
5. Sistemul crează o înregistrare care conţine fişierul comprimat şi criptat.

4.2.1.4 Scenariul alternativ 1

Începutul scenariului

1. Apăsarea butonului de anulare.
   - Poate avea loc în oricare dintre paşii (1) - (3) ai scenariului.

Sfârşitul scenariului

1. Meniul de încărcare este închis.
   - Nu s-a executat nici o scriere în sistem.

4.2.1.5 Scenariul alternativ 2

Începutul scenariului

1. A intervenit o eroare în procesul de încărcare.
   - Poate apărea dacă nu a fost introdusă o frază de criptare validă.
   - Utilizatorul nu mai are spaţiu disponibil.
   - Directorul selectat nu există sau este corupt.
   - Formatul fişierului nu este suportat.

Sfârşitul scenariului

1. O eroare este afişată.
   - Nu s-a executat nici o scriere în sistem.

4.2.2 CU02 Creare fişier

Acest caz de utilizare se referă la crearea unui fişier în sistem. Figura 4.5 deprezintă diagrama de activitate pentru acest caz de utilizare.

4.2.2.1 Precondiţii

- Utilizatorul are un cont activat, care nu este blocat.
- Utilizatorul este autentifiicat.

4.2.2.2 Postcondiţii

- În Drive-ul utilizatorului a fost creat un fişier nou, acesta este criptat şi compresat dacă operaţia s-a finalizat cu succes.
- Nu s-a efectuat nici o scriere în cazul anulării operaţiei de către utilizator.

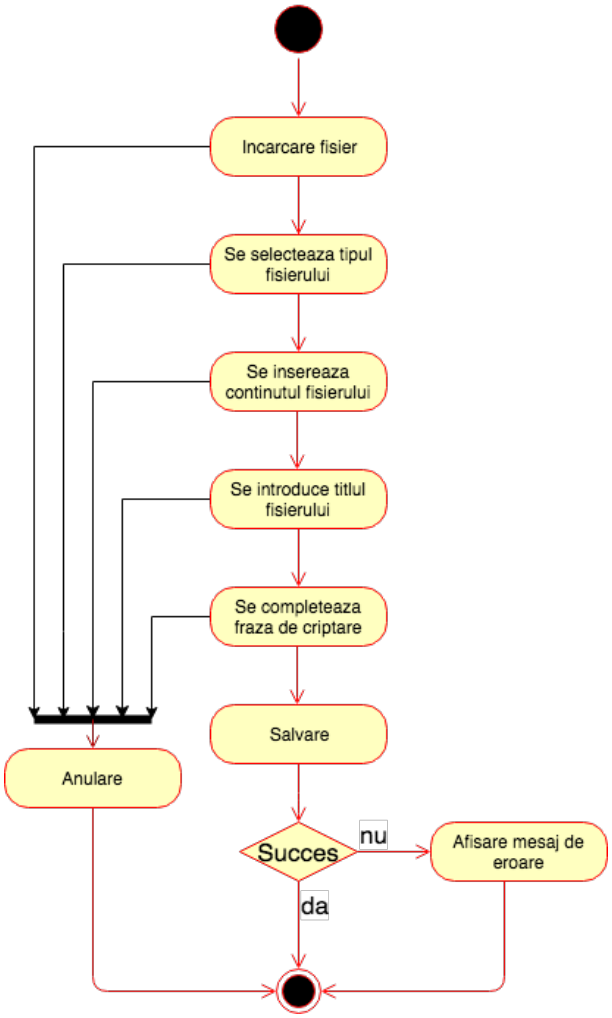

Figura 4.5: Diagrama de activitate ”Creare fişier”

4.2.2.3 Scenariul principal

1. Se apasă butonul de creare fişier.
2. Se selectează tipul fişierului ce va fi creat.
3. Se inserează în câmpul rezervat conţinutul dorit al fişierului.
4. Se introduce în câmpul rezervat numele fişierului.
5. Se completează fraza de criptare a fişierului.
6. Se apasă butonul de confirmare a creării.
7. Sistemul crează o înregistrare care conţine fişierul comprimat şi criptat.

4.2.2.4 Scenariul alternativ 1

Începutul scenariului

1. Apăsarea butonului de anulare.
   - Poate avea loc în oricare dintre paşii (1) - (5) ai scenariului.

Sfârşitul scenariului

1. Meniul de creare fişier este închis.
   - Nu s-a executat nici o scriere în sistem.

4.2.2.5 Scenariul alternativ 2

Începutul scenariului

1. A intervenit o eroare în procesul de încărcare.
   - Poate apărea dacă nu a fost introdusă o frază de criptare validă.
   - Utilizatorul nu mai are spaţiu disponibil.
   - Directorul selectat nu există sau este corupt.
   - Conţinutul fişierului conţine caractere nepermise.

Sfârşitul scenariului

1. O eroare este afişată.
   - Nu s-a executat nici o scriere în sistem.

4.2.3 CU03 Descărcare fişier

Acest caz de utilizare descrie operaţia de descărcare a unui fişier din Drive-ul utilizatorului. Diagrama de activitate pentru acest caz de uttilizare este prezentată în Figura 4.6.

4.2.3.1 Precondiţii

1. Utilizatorul are un cont activat, care nu este blocat.
2. Utilizatorul este autentifiicat.
3. Fişierul există şi are un format valid.

4.2.3.2 Postcondiţii

1. Pe calculatorul utilizatorului se crează un fişier cu conţinutul fişierului selectat.
2. În sistem nu s-a produs nici o schimbare.

4.2.3.3 Scenariul principal

1. Se navighează în arborele de fişiere până la directorul în care se află fişierul dorit.
2. Se selectează fişierul care se doreşte a fi descărcat.
3. Se apasă utonul de descărcare.
4. Parola de decriptare va fi introdusă în câmpul special.
5. Opţional se va introduce fraza de steganografie.
6. Se va confirma operaţia de descărcare.
7. Un nou fişier va fi creat pe calculatorul utilizatorului.

4.2.3.4 Scenariul alternativ 1

Începutul scenariului

(a) Apăsarea butonului de anulare a descărcării.

- Poate avea loc în oricare dintre paşii (1) - (3) ai scenariului.

Sfârştitul scenariului

(a) Meniul de descărcare fişier este închis.

- Nu s-a executat nici o scriere în sistem. Nu s-a creat nici un fişier pe sistemul utilizatorului.

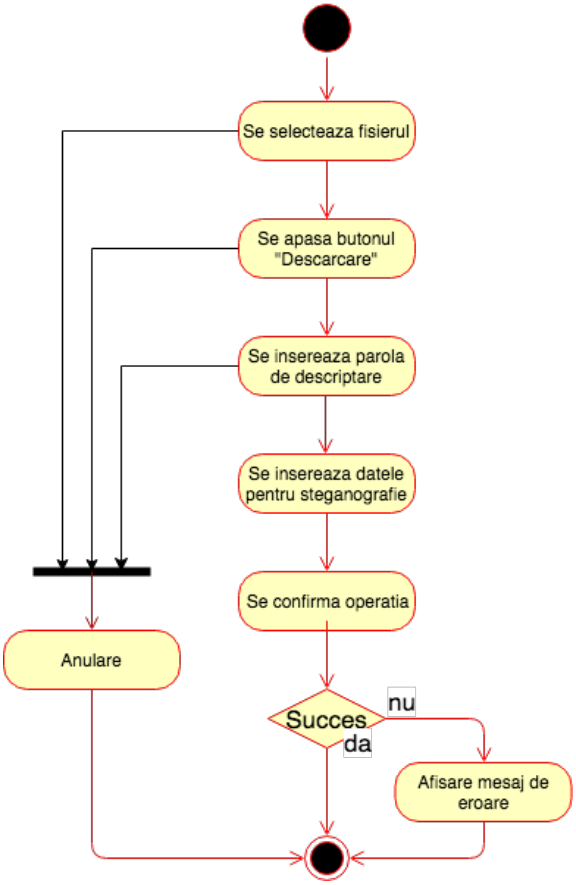

Figura 4.6: Diagrama de activitate ”Descărcare fişier”

4.2.3.5 Scenariul alternativ 2

Începutul scenariului

1. A intervenit o eroare în procesul de descărcare.
   - Poate apărea dacă nu a fost introdusă o frază de decriptare validă.
   - Utilizatorul nu mai are spaţiu disponibil pe calculator.
   - Fişierul selectat nu există sau este corupt.
   - Datele pentru steganografie nu sunt valide.

Sfârşitul scenariului

1. O eroare este afişată.

   - Nu s-a executat nici o scriere în sistem.

4.2.4 CU04 Schimare director fişier

Acest caz de utilizare descrie operaţia de schimbare a directorului părinte a unui fişier. Diagrama de activitate pentru acest caz de utilizare este prezentată în Figura 4.7.

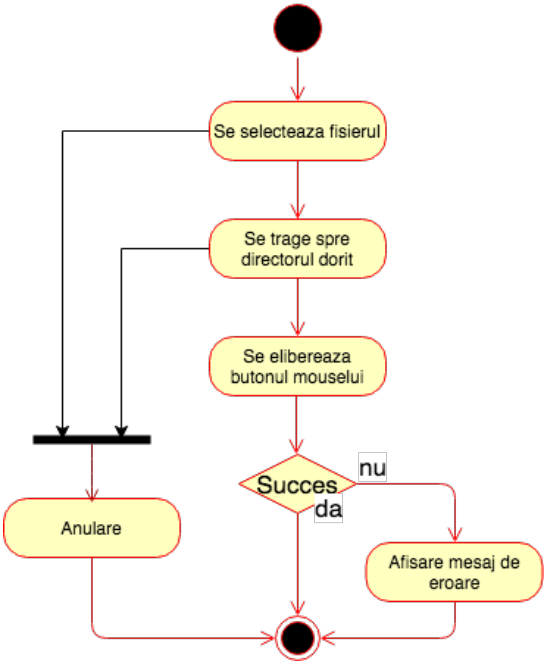

Figura 4.7: Diagrama de activitate "Schimbare director fişier"

4.2.4.1 Precondiţii

1. Utilizatorul este autentificat şi autorizat pentru această operţie.

2. Fişierul există.

4.2.4.2 Postcondiţii

1. În directorul sursă nu mai există referinţă la fişierul selectat, fişierul apare ca şi copil al directorului destinaţie.

4.2.4.3 Scenariul principal

1. Se navighează până în directorul în care se află fişierul pe care se va efectua operaţia.
2. Se selectează fişierul prin apăsarea butonului stâng al mouse-ului.
3. Se trage fişierul până când acesta se află deasupra directorului dorit.
4. Se eliberează butonul mouse-ului.
5. Părintele fişierului a fost schimbat.

4.2.4.4 Scenariul alternativ 1

Începutul scenariului

1. Se relaxează butonul mouse-ului fără ca fişierul să se afle deasupra unui director..
   - Poate avea loc în oricare dintre paşii (1) - (2) ai scenariului.

Sfârşitul scenariului

1. Operaţia este anulată.
   - Nu s-a executat nici o scriere în sistem. Nu s-a schimbat părintele fişierului.

4.2.4.5 Scenariul alternativ 2

Începutul scenariului

1. A intervenit o eroare în procesul de schimbare.
   - Fişierul a fost plasat deasupra unui alt fişier, operţia nu poate fi efectuată.
   - A intervenit o eroare în procesul de schiere din baza de date.

Sfârşitul scenariului

1. O eroare este afişată.
   - Nu s-a executat nici o scriere în sistem.

4.2.5 CU05 Ştergere fişier

Acest caz de utilizare se referă la operaţia de ştergere a unui fişier din Drive-ul utilizatorului. Diagrama de activitate pentru acest caz de utilizare este reprezentată în Figura 4.8.

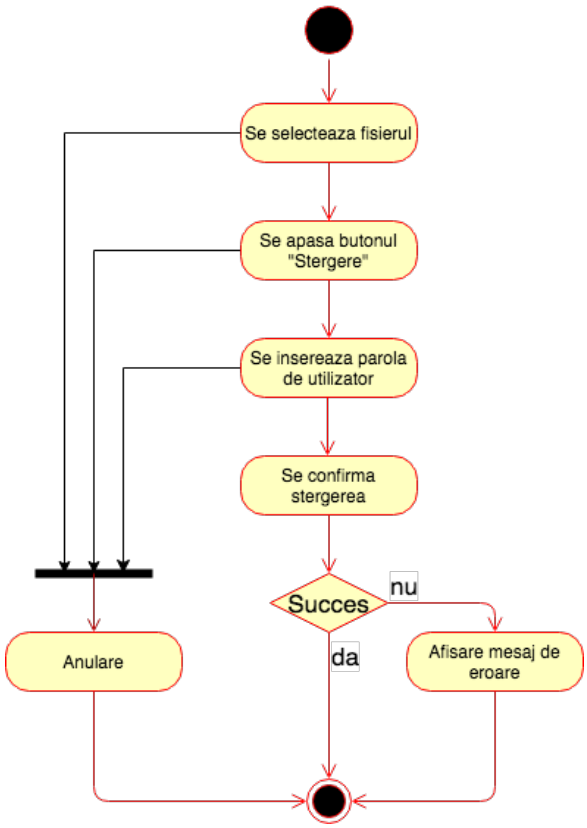

Figura 4.8: Diagrama de activitate ”Ştergere fişier”

4.2.5.1 Precondiţii

1. Utilizatorul are un cont activat, care nu este blocat.
2. Utilizatorul este autentifiicat.
3. Fişierul există şi are un format valid.

4.2.5.2 Postcondiţii

1. Fişierul nu va mai apărea în lista de fişiere a utilizatorului.

4.2.5.3 Scenariul principal

1. Se navighează până în directorul în care se află fişierul pe care se va efectua operaţia.
2. Se apasă butonul de ştegere care se află sub fişier.
3. În fereastra apărută se introduce parola de utilizator.
4. Se confirmă operaţia.
5. Fişierul este şters din sistem.

4.2.5.4 Scenariul alternativ 1

Începutul scenariului

1. Se apasă butonul de anulare a operaţiei
   - Poate avea loc în oricare dintre paşii (1) - (3) ai scenariului.

Sfârştitul scenariului

1. Operaţia este anulată şi fereastra de ştergere este închisă.
   - Nu s-a executat nici o scriere în sistem. Fişierul nu a fost şters.

4.2.5.5 Scenariul alternativ 2

Începutul scenariului

1. A intervenit o eroare în procesul de ştergere.
   - Fişierul nu există sau a fost şters înainte.
   - Parola introdusă este greşită.

Sfârştitul scenariului

1. O eroare este afişată.
   - Nu s-a executat nici o scriere în sistem.

4.3 Tehnologii

În acestă secţiune sumt descrise tehnologiile folosite pentru implementarea proiectului şi avantajele pentru care au fost selectate.

4.3.1 Golang

Golang este un limbaj de programare care a apărut în 2007, fiind şi un proiect *open source*. Este un limbaj care care ofera analiză statică a codului: *gofmt* - formatarea statică a codului, *golint* - stilizarea codului, *godoc* - documentarae codului, acestea oferă o siguranţă asupra codului scris. În Figura 4.9 este reprezentată emblema oficială a Golangului.

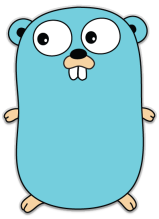

Figura 4.9: Golang

Go oferă un tool pentru testare integrat în limbaj, acesta a fost elaborat pentru simplitate şi eficienţă. Avantajul cheie este concurenţa, care este o particularitate integrată în limbaj, lucru care face limbajul să fie rapid, acest fapt este datorat *go rutinelor*, care au nevoie de un management mai simplu, efectuat la nivel de limbaj, nu platformă. Limbajul dispune de *Garbage Collector*, ceea ce permite eficienţa managementului de memorie şi resurse. Tipizarea este un alt avantaj al Golangului, acesta îl face rezistent la erori cauzate de erorile de conversie.

Limbajul oferă un API foarte simplu care poate fi folosit pentru orice tip de teste. Framework-ul este integrat în libaj şi permite crearea simplă a testelor tabelare, efectuarea verificărilor de acoperire a testelor. Limbajul permite analiza statică a codului, aceasta poate fi folosită pentru generarea automată a documentaţiei şi detecţia unor eventuale erori, cum ar fi: erori netratate, variabile neutilizate, metode neapelate şi lipsa verificărilor de tip înate de efectuarea unei conversii sau a unui *cast*.

Printre marele companii care utilizează Golang se numără: Google, YouTube, Facebook, Apple, Docker, Dropbox, Twitter, Netflix, Ethereum, Kubernetes, User, Couchbase, IBM şi altele. Dropbox, sistem descris în secţiunea 3.5.3, a migrat câteva dintre funcţionalităţile sale criptice spre Go în 2013. Sistemul a fost scris iniţial în Python, scopul trecerii a fost creştere performanţei, creşterea numărului de clienţi şi viteza de dezvoltare software.

Datorită performanţei, vitezei de procesare şi răspuns, Golang a fost selectat ca limbaj primar pentru acest proiect. Un alt motiv ar fi şi faptul că mai multe sisteme care efetuează operaţii cu volume de date destul de mari au optat pentru acest limbaj, lucru care denotă faptul că este potrivit pentru procesare de fişiere mari.

4.3.2 gRPC

gRPC este un *framework* RPC foarte performant, care poate rula pe orice platformă şi în orice mediu. Acesta poate conecta într-un mod eficient serviciile din diferite centre de date, acesta suportă balansarea, urmărirea, verificarea funcţionării corecte şi autentificarea. Acesta este de asmenea aplicabil în cazul sistemelor distribuite pentru conectarea dispozitivelor, aplicaţiilor mobile şi a browserelor la serviciile de back-end.

Avantajele *framework*-ului gRPC sunt:

- **Definirea simplă a serviciilor** - definirea este posibilă prin *Protocol Buffers*, care sunt un limbaj performant de serializare binară.
- **Simplu de utilizat şi scalat** - instalarea mediului de lucru se poate face în câţiva paşi simpli şi se poate scala la milioane de apeluri RPC pe secundă.
- **Compatibil cu multiple limbaje şi platforme** - se poate genera în mod automat codul pentru server şi client gRPC în numeroase limbaje, cum ar fi: Java, Go, C#, C++, JavaScript, Python.
- **Streaming bi-directional şi autentificare integrată** - se poate face streaming bi-direcţional şi dispune de autentificare integrată şi configurabilă cu transport prin HTTP/2.

Datorită avantajelor sale, acest tool este folosit de companii precum: Netflix, CoreOS, Carbon3D, Cisco, Google şi multe altele.

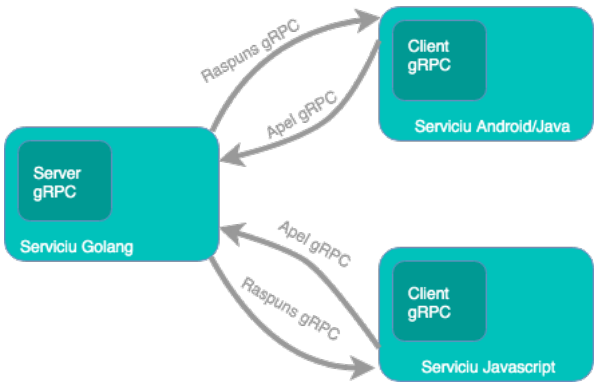

Figura 4.10: Exemplu de interacţiune gRPC

În gRPC o aplicaţie client poate apela metodele aplicaţiei server care rulează pe alt dispozitiv de parcă ar fi metode locale, modelul de interacţiune a clientului şi serverului este prezentat în Figura 4.10. Principiul principal al gRPC este definirea serviciilor, metodelor

şi parametrilor acestora prin *Protocol Buffers*, un exemplu de definiţie a unui serviciu este prezent în Anexa A.

Porning de la definiţia fişierului *.proto*, gRPC oferă compilatorul pentru fişierele protobuf care generează codul client şi sever. Utilizatorii gRPC apelează APi-urile de pe partea de client şi le implementează pe partea de server.

În concluzie, principalul motiv pentru care gRPC a fost selectat pentru proiectul respectiv este viteza de comunicare a 2 entităţi ce utilizeză această metodă. API-urile create cu ajutorul protocolului gRPC sunt mult mai performante decât cele create cu ajutorul metodei clasice REST, timpul de răspuns micşorându-se de la 15 ms la 1.71 ms pentru fiecare apel. Numărul mediu de apeluri variază de la 600 apeluri/minut la 1700 apeluri/minut.

4.3.3 Vue.JS

Vue.js este un framework progresiv de JavaScript, utilizat pentru construirea unor interfeţe utilizator şi aplicaţii web. Vue utilizeză template-uri cu sintaxă bazată pe HTML care permit randarea paginilor utilizând asocierea elementelor cu datele de instanţă ale componentei Vue a aplicaţiei.

Vue foloseşte un sistem reactiv bazat pe Javascript pur care permite optimizarea re-randării. Acest sistem foloseşte urmărirea dependenţelor, pentru a şti când trebuie re-randate componentele şi care dintre ele trebuie randate. La fel ca majoritatea framework-urilor de JS, Vue oferă suport nativ pentru TypeScript, fapt care în oferă şi mai multă flexibilitate.

Avantaje cheie ale Vue.js sunt:

- **Are de dimensiuni mici** - popularitatea framework-urilor Javascript depinde mult de dimensiune, cu cât este un framework mai mic cu atât va fi mai folosit.

- **Abordare simplă** - Vue este foarte simplu de integrat în orice proiect web existent. Acest lucru face Vue.js să fie foarte avantajos în dezvoltarea rapidă a aplicaţiilor web.

- **Flexibilitate** - aplicaţiile Vue pot fi rulate uşor din browser, fiecare schimbare în cod este detectată şi aplicaţia este recomplilată automat. Această abordare este deosebit de utilă în timpul testării aplicaţiilor.

- **Versatilitate în privinţa dimensiunii aplicaţiilor** - Vuex este un feature al Vue.js care este folosit pentru pentru managementul bazat pe stare şi funcţii de rutare. Acesta poate fi folosit pentru construirea unor funcţii mai largi şi mai complexe.

- **Integrare simplă** - framework-ul are nişte capabilităţi de integrare uimitoare cu aplicaţiile existente, fapt datorită căruia i-a crescut popularitate în comităţile de developeri.

Avantajele framework-ului l-au făcut să câştige popularitate încă de la aparţia acestuia, printre utilizatorii Vue.JS se numără Adobe, Gitlab, Netflix, Xiaomi, Codeship şi alţii. Acesta este foarte flexibil şi versatil, dar find foarte nou, are o comunitate destul de restrânsă de programatori, unele funcţionalităţi şi probleme nu sunt documentate, din acest punct dezvoltarea devine mult mai complicată.

4.3.4 MongoDB

MongoDB este o bază de date orientată pe documente, aceasta este disponibilă pe mai multe platforme. Avantajele cheie ale sistemului sunt performanţa, nivelul ridicat de disponibilitate şi scalabilitate. Baza de date este container-ul fizic pentru coloecţii. Colecţiile la rândul lor sunt un grup de documente. Documentele sunt un set de perechi cheie valoare. Documentele au o schemă dinamică şi nu trebuie să aibă aceeaşi structură.

Avantajele cheie ale MongoDB sunt:

- **Lipsa schemei** - MongoDB este o bază de date bazată pe documente, fapt care oferă programatorilor libertatea de a stoca şi grupa datele în orice mod. Structura unui obiect este foarte simplă şi clară.
- **Flexibilitate query-urilor** - Se pot executa filtrări complexe, datorită unui limbaj de *query* foarte versatil.
- **Este uşor de scalat** - scalarea bazei de date poate fi configurată foarte uşor.
- **Suport pentru diferite tipuri de date** - datorită structurii datelor, nu mai este necesară consersia sau maparea datelor aplicaţiei la un tip specific, acestea sunt convertite automat in BSON.
- **Este bine documentat** - fiind foarte popular, MongoDB dispune de o documentaţie foarte bine definită.
- **Indexarea** - se pot crea indecşi, aceştia contribuie la creşterea vitezei de căutare.

Deşi MongoDB nu dispune de câteva avantaje ale bazelor de date relaţionale, cum ar fi *exchange* şi *join*, însă scalabilitatea pe orizolntală şi adaptabilitatea schemei îi oferă un avataj deosebit şi îl fac un sistem foarte performant şi popular. Deasemenea două motive importante pentru care se potriveşte acestui proiect sunt: *nivelul de criptare suplimentar pe care îl oferă* şi *eficienţa în lucrul cu big data.*. Această bază de date a fost selectată pentru stocarea datelor utilizatorilor, deoarece oferă un suport simplu şi amplu.

4.3.5 Couchbase

Bazele de date NoSQL au schimbat procesul de dezvoltare a aplicaţiilor software din punctul de vedere a adaptabilităţii schemei dinamice şi a scalabilităţii. Comparând

între ele sistemele de baze de date NoSQL existente, Couchbase[26] este una dintre cele mai rapide. Este uşor de cofigurat şi are nişte funcţionalităţi puternice pentru stocarea datelor cu diverse structuri. Pentru extragerea datelor uilizează tehnica *map-reduce*, ceea ce face viteza de procesare să crească considerabil. În sistem, datele sunt stocate ca o pereche cheie-valoare, o citire a datelor se poate efectua într-uun singur pas în cazul în care se cunoaşte cheia obiectului, în cazul unor operaţii mai complexe, timpul de procesare creşte. Prin adăugarea indecşilor se poate efcientiza aproape orice operaţie.

Echivalentul temenului bază de date din terminologiile relaţionale, în ternCochbase, este numit *bucket*, acesta este un container logic pentru documentele grupate logic. Couchbase oferă câteva mecanisme de configurare, interacţiune şi monitorizare a clusterelor. Acestea sunt: interfaţă grafică web pentru administrator, administrarea utilizând REST API şi interfaţă de control din linia de comandă. Datorită acestor funcţionalităţi, adinistratorul, poate configura, edita şi expanda baza de date într-un mod simplu şi eficient.

Securitatea este foarte importantă în domeniul de stocare a datelor, Couchbase oferă diverse funcţionalităţi care permit configurarea unor controale şi verifcări care asigură integritatea şi securitatea sistemului. Se poate bloca accesul aplicaţiilor de la anumite adrese, sau se pot configura adresele şi domeniile care pot accesa sistemul. De asemenea, la fel ca şi MongoDB, Couchbase oferă un nivel suplimentar de criptare a datelor stocate.

Viteza, nivelul înalt de securitate, viteza de operare a datelor şi configurabilitatea fac Couchbase să fie cel mai potrivit sistem pentru stocarea fişierelor din aplicaţie. Un avantaj semnificativ sunt funcţionalităţile de procesare a datelor cu volum mare.

4.3.6 HTML, CSS, Bootstrap

HTML*(HyperText Markup Language)*[1] este blocul de bază pentru construirea paginilor Web. Acesta defineşte structura şi identitatea conţinutului web. Cuvântul *Hypertext* se referă la conexiunea unei pagini web cu alta prin *linking*. Elementele structurale a HTML sunt numite *tag*-uri, acestea trebuie structurate ca un arbore multicăi, rădăcina paginii web este tag-ul <*html*>.

CSS*(Cascading Style Sheets)*[2] este limbajul utilizat în web pentru a descrie prezentarea documentelor scrise în HTML sau XML. CSS descrie modul în care elementele vor fi afişate pe ecran, hârtie sau alt tip de media. Se poate configura modul de afişare, poziţionarea şi efectele elementelor.

Bootstrap este un framework de front-end care crează condiţii pentru o dezvoltare mai rapidă a paginilor web. Acesta include template-uri HTML şi CSS pentru design-ul elementelor tipografice, butoanelor, formurilor, tabelelor, modalurilor, imaginilor şi multor altor elemente. Datorită acestui framework se pot crea design-uri customizable şi foarte responsibile.

[1] https://developer.mozilla.org/en-US/docs/Web/HTML
[2] https://developer.mozilla.org/en-US/docs/Web/CSS

CSS, împreună cu HTML şi alte limbaje formează nucleul reprezentării paginilor Web. Fără CSS orice pagină web ar fi albă, fără posibilitatea de a avea un aranjament customizat al elementelor, doarc cele disponibile în HTML: listă, tabel, div-uri.

4.3.7 JSON Web Token(JWT)

JWT[3] este un standard(RFC 7519[4]) care defineşte transmisia compacta şi securizată a informaţiilor între 2 sisteme sub formă de obiect JSON. Această informaţie poate fi verificată şi considerată de încredere, deoarece are o semnătură digitală. JWT-urile pot fisemnate folosind un secret, prin metoda HMAC sau printr-un algoritm de cheie publica/privată folosind RSA.

Contextele în care sunt folosite JWT sunt:

- **Autorizare**: cel mai comun scenariu în care se foloseşte JWT. Imediat după logare, fiecare apel făcut de utilizator, va include un token, acesta va asigura accesul la rute, servicii, resurse.
- **Schimb de informaţii** - asets tip de token-uri sunt o cale sigură de schimb al informaţiei între 2 utilizatori sau două sisteme. Deoarece acestea pot fi semnate utilizând perechi de chei publice şi private - avem siguranţa că identitatea expediatorului este autentică.

Datorită acestor avantaje, JWT au fost alese pentru sistemul de autentificare. Stau la baza unui sistem cu un nivel de securitate înalt, astfel scopul proiectului poate fi atins.

4.3.8 Docker şi Kubernetes

Docker[5] este singura platformă compilare end-to-end, partajare şi rulare a aplicaţiilor bazate pe containere. Acesta poate efectua managementul unei întregi aplicaţii din momentul compilării pe calculatorul programatorului până la pasul în care ajunge pe cloud. Docker este cea mai rapidă soluţie pentru deplotementul unei aplicaţii spre producţie. Acesta oferă suport securizat pentru livrarea aplicaţiilor pe orice tip de cloud - de la cloud hibrid până la muchie.

Kubernetes[6] este un sistem de orchestrare open-source, folosit pentru management, plasarea, scalarea şi rutarea containerelor. Platforma Docker oferă un mediu Kubernetes securizat şi foarte performant, acesta este potrivit pentru atingerea unor scopuri comune în dezvoltarea software şi oferă maximă flexibilitate.

Atunci când se construieşte o arhitectură bazată pe microservicii este necesară agregarea tuturor microserviciilor. Kubenetes oferă numeroase abstractizări şi API-uri pentru a permite acest lucru.

[3]https://jwt.io
[4]https://tools.ietf.org/html/rfc7519
[5]https://www.docker.com
[6]https://www.docker.com/products/kubernetes

- Poduri - un grup de containere care conţin diverse microservicii ce pot fi conectate ca un întreg.

- Kubernetes oferă load-balancing, identificarea şi descoperirea microserviciilor pentru a putea fi izolate.

- Se poate configura nivelul de interacţiune între servicii.

Se poate efectua decuplarea aplicaţiei de maşina fizică, astfel mai multe microservicii pot convieţui pe acelaşi dispozitiv fizic fără a interfera. Astfel se reduce costul de consum al arhitecturii bazate pe microservicii.

Kubernetes a fost contruit pentru a schimba modul, în care aplicaţiile sunt construite şi consumate în cloud. Datorită combinaţiei dintre Docker şi Kubernetes sistemul poate avea un grad de *velocitate*, *eficienţă* şi *agilitate* mult mai înalt. Folosirea Docker şi Kubernetes creşte nivelul de eficienţa şi securitate a unei aplicaţii, acestea sunt indispensabile pentru construirae unui mediu securizat de stocare şi lucru.

4.3.9 Google Cloud

Costul resurselor hardware este foarte mare, iar sistemele sunt foarte greu se scalat. Sistemele cloud sunt soluţia cea mai buná pentru problema respectivă. Google Cloud[7] este o soluţii foarte populară pentru servicii în cloud. Principalele avantaje sunt: nivelul înalt de securitate, criptarea, protecţia identităţii şi multe servicii auxiliare pe care le oferă.

Motivele pentru care Google Cloud a fost ales ca soluţie pentru acest sistem sunt:

- **Preţ mai scăzut decât competitorii** - pentru o aplicaţie scalabilă, la început de drum, este o soluţie fezabilă şi eficientă.

- **Monitorizarea şi migrarea în timp real a maşinilor virtuale** - nici unul dintre marii competitori nu oferă această funcţionalitate. Avantajul oferit constă în faptul că reparaţia, scalarea şi updatarea părţilor hardware sau software se poate efectua foarte rapid.

- **Performanţă sporită** - poate suporta până la 60000 de apeluri concurente

- **Securitatea** - toate datele sunt criptate, toate accesele la componente pot fi efectuate doar după autorizare.

Utilizarea platformei Google Cloud oferă numeroase avantaje. Nivelul se securitate oferit, precum şi performanţa sporită îl transformă într-o soluţie potrivită pentru rularea unui sistem securizat de stocare.

[7] https://cloud.google.com

4.3.10 Git

Sistemele de control a versionării sunt o categorie de tool-uri cu ajutorul cărora se poate face managementul în timp al codului. Datorită acestor sisteme se poate monitoriza orice schimbare a codului sursă, se pot anula şi modifica diverse schimbări.

Git[8] este în momentul actual cel mai popular sistem de control al versionării. Avantajele pe care le oferă sunt: viteza de dezvoltare, managementul simplu al codului, securitatea şi siguranţa. *Github*[9] este una dintre cele mai populare platforme web care oferă servicii de versionare a codului. Acesta oferă posibilitatea de a avea un cont gratuit şi un număr nelimitat de proiecte hostate în mod gratuit.

[8] https://git-scm.com/about
[9] www.github.com

Capitolul 5

Proiectare de Detaliu şi Implementare

Acest capitol prezintă deciziile şi paşii de implementare parcurşi în ciclul de dezvoltare al proiectului. Sistemul propus este format din 3 subsisteme: aplicaţia web; serverul, care este format din alte subsisteme, şi bazele de date. Capitolul va oferi o descriere succintă a tuturor componentelor, incluzând şabloanele arhitecturale, şabloanele de design şi algoritmii folosiţi în dezvoltarea proiectului.

5.1 Arhitectura serverului

5.1.1 Descriere generală

Pentru dezvoltarea aplicaţiei de server, principala tehnologie folosită a fost Golang, pentru unul dintre module s-a folosit Python. Pentru structura proiectului am ales şablonul arhitectural bazat pe microservicii, şablonul şi avantajele au fost descrise în secţiunea 3.2. Secviciile din care este compus serverul sunt:

1. Authentication service (descris în 5.1.3)
2. Compression service (descris în 5.1.6)
3. Crypto service (descris în 5.1.4)
4. File service (descris în 5.1.7)
5. Watermarking service (descris în 5.1.5)
6. User service (descris în 5.1.8)

Arhitectura sistemului este prezentată în Figura 5.1

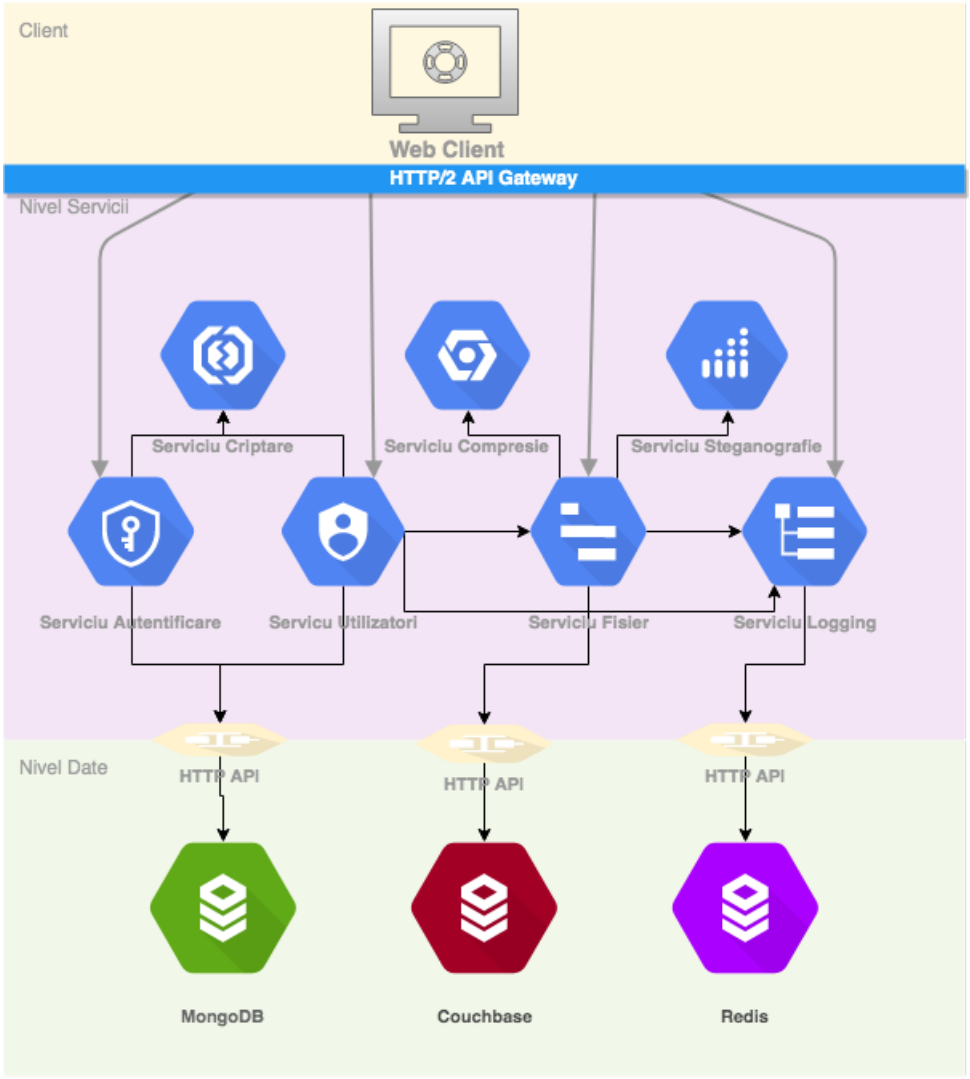

Figura 5.1: Arhitectura sistemului

De asemenea pentru dezvoltarea serverului au fost folosite 2 tipuri de baze de date: MongoDB şi Couchbase. Ambele fiind accesibile doar prin serverul dedicat. Pentru orchestrarea şi punerea în funcţiune a microserviciilor am folosit Docker si Kubernetes. Pentru maparea API-urilor am folosit envoy şi Docker.

5.1.2 Orchestrarea microserviciilor

Microserviciile sunt o modalitate de a împărţi funcţionalităţile dintr-un sistem. Acestea ne oferă flexibilitatea de a scala funcţionalităţi specifice şi de a fi agili în livrarea produselor. După ce funcţionalităţile au fost separate în subsisteme dedicate, întrebarea

următoare ar fi: cum să le ”lipim” la loc?

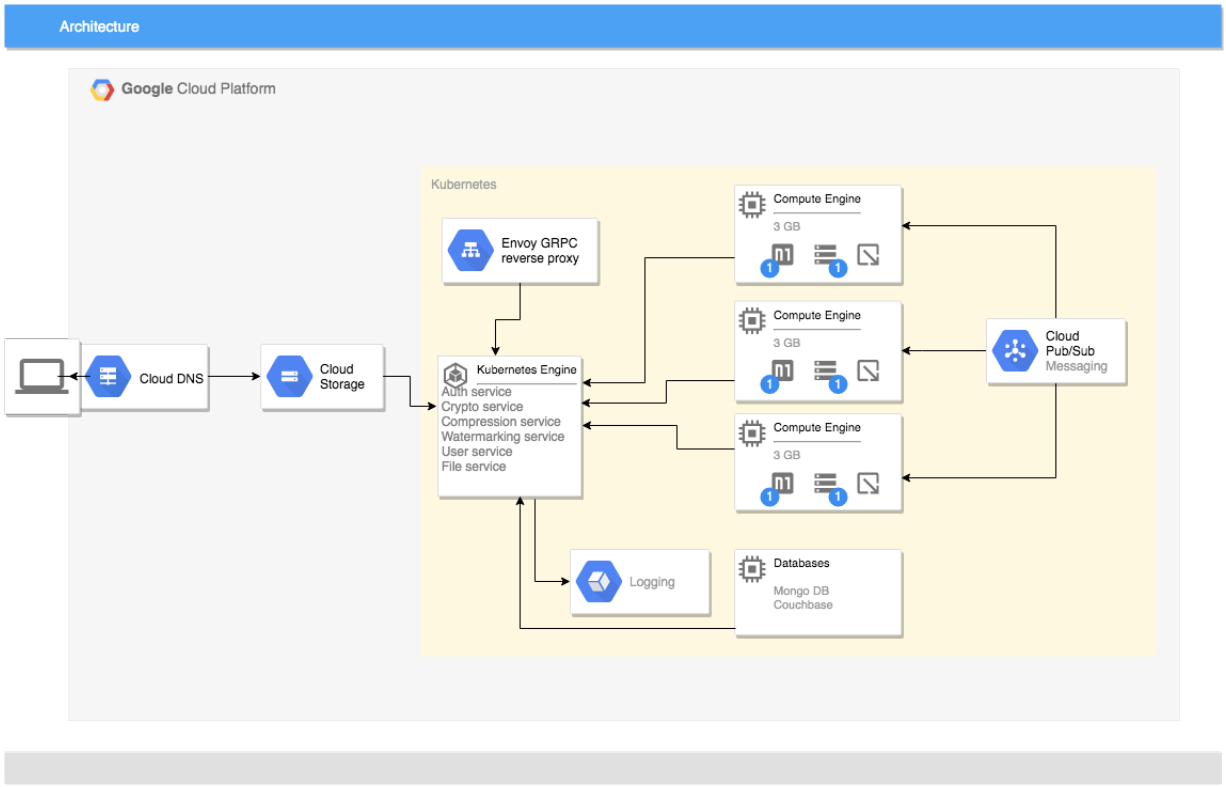

Figura 5.2: Diagrama orchestrării în Cloud

În procesul de stabilire a comunicării între servicii am avut o provocare destul de mare: să păstrez cuplarea la un nivel cât mai jos, în caz contrar ar putea apărea: multiple căderi, testare necalitattivă, dificultate crescută de înţelegere şi costuri crescute de consum.

5.1.3 Microserviciul de autentificare

Microserviciul de autentificare este responsabil pentru generarea şi validarea cheilor de sesiune. Serviciul este organizat pe nivele: acces date, validatori, logică de business şi nivel de API. Arhitectura serviciului este prezentată în Figura 5.3.

Figura 5.3: Arhitectura serviciului de autentificare

Nivelul de acces date efectuează operaţii pentru identificarea şi validarea utilizatorilor în funcţie de datele de autentificare pe care le-a oferit utilizatorul. Baza de date folosită este *MongoDB*, serviciul accesează acceaşi colecţie ca şi serviciul de utilizatori. Pentru operaţia de generare a unui Token de sesiune se caută utlizatorul cu credenţialele care au fost primite de la utilizator, dacă sunt valide se returnează un Token JWT în care sunt encodate datele utilizatorului. Pentru validarea tokenului, la acest nivel, se execută operaţia de verificăre dacă acesta este existent în baza de date şi nu este expirat.

Responsabilitatea *Validatorilor* este de a verifica dacă datele de intrare, cum ar fi parole, emailuri, token-uri sau alte date importante îndeplinesc nişte reguli stricte de format şi securitate.

La nivelul *Logicii de Business* se execută operaţii de encodare şi decodare a token-urilor, se gestionează accesul la resurse. De asemnea, la acest nivel se iau decizii legate de blocarea unor conturi în cazuri de încercări repetate de a folosi token-uri invalide.

Nivelul superior, *API gRPC* este responsabil pentru gestionarea cererilor utilizatorilor. Acest nivel conţine implementaea interfeţei de comunicare gRPC, fiecare metodă definită în protobuf este implementată şi poate fi apelată de un client gRPC.

În *Golang* nu există clase propriu-size, există structuri care, dintr-o perspectivă foarte generică, pot fi considerate clase. Structurile pot encapsula alte structuri sau alte tipuri de date, pot fi instanţiate printr-un constructor default care are număr variadic de parametri cu nume. În Figura 5.4 este prezentată diagrama de entităţi şi asocieri a serviciului.

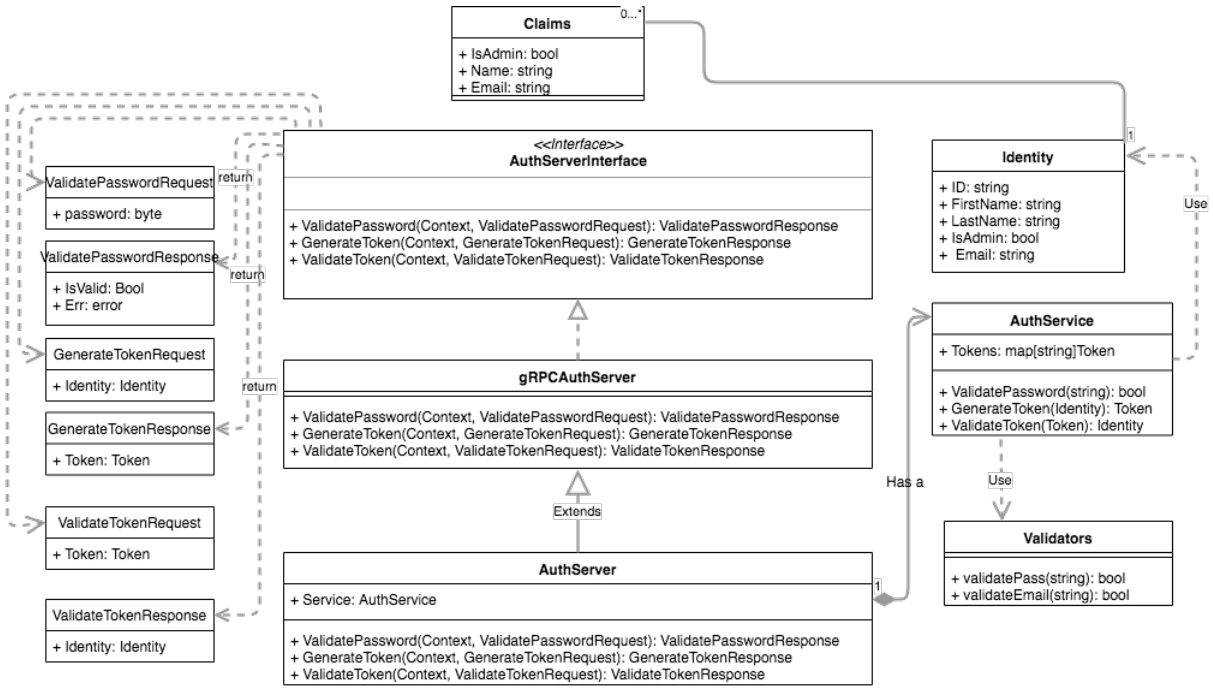

Figura 5.4: Diagrama de clase a serviciului de autentificare

Cea mai importantă clasă este AuthServer, acesta extinde gRPCAuthServer, care este o implementare a interfeţei definite în fişierul *service.proto*.

5.1.4 Microserviciul de criptare

Microserviciul de criptare are ca funcţionalitate primară efectuarea tuturor operaţiilor de criptare, decriptare şi hashing. Structura sistemului este prezentată în Figura 5.5, acesta are arhitectura organizată pe nivele.

Figura 5.5: Arhitectura serviciului de criptare

Nivelele care compun serviciul sunt: *gRPC API Level*, *Logica de business* şi *Utilităţile de criptare*. În continuare va fi prezentată descrierea fiecărui nivel în parte.

Nivelul de utilităţi de criptare encapsulează logica pentru 3 algoritmi: Argon2, twofish şi o versiune perfecţionată a pbkdf2. Pentru selecţia metodei de criptare a fost folosit şablonul de design *Strategy*, arhitectura nivelului este prezentată în Figura 5.6.

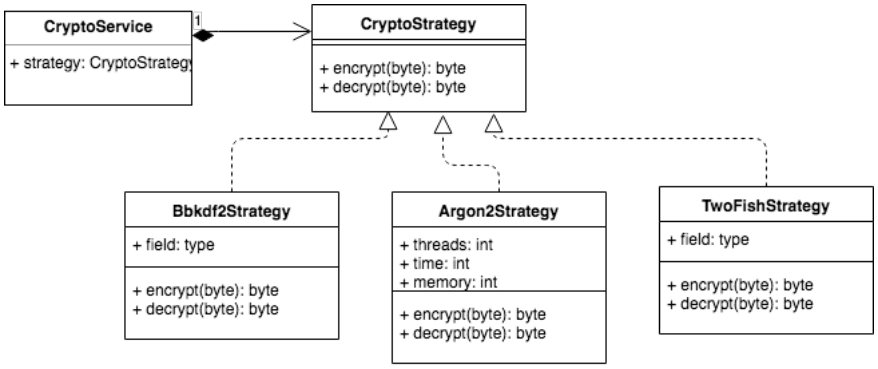

Figura 5.6: Şablonul arhitectural Strategy

Argon2 - este un algoritm de derivare a cheilor, acetsa a fost ales ca strandard pentru Hashing-ul parolelor în iulie 2015. Acesta este disponibil în 3 versiuni: Argon2d, Argon2i şi Argon2id, pentru acest proiect a fost selectat Argon2id deoarece acesta este mai complex şi nu are vulnerabilităţi descoperite. Avantajul algoritmului este configurabilitatea, se pot transmite 3 parametri de calibrare ai algoritmului: timp, memorie şi thread-uri.

Pentru configurarea automată a algoritmului a fost definit un FSM[1] care ajustează parametrii algoritmului pentru a obţine un anumit timp de execuţie, în Figura 5.7 este prezentată structura FSM-ului.

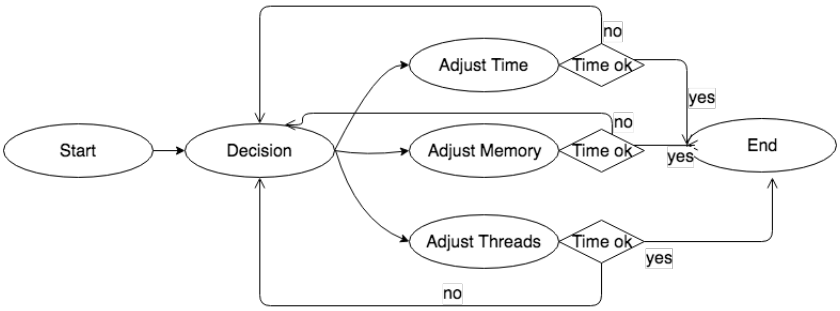

Figura 5.7: FSM-ul de calibrare a Argon2

[1]https://en.wikipedia.org/wiki/Finitestate_machine

Dacă timpul estimat este apropiat de timpul cerut, FSM va avea parametrii de ieşire necesari pentru calibrarea algoritmului, altfel s evor lua decizii de creştere sau scădere a coeficientului de timp. memorie sau thread-uri. Deoarece parametrii sunt avriabili în funcţie de hardware, aceştia vor fi salvaţi în baza de date împreuna cu hashu-ul obţinut, fapt care paote cauza probleme în cazul în care un atacator pune mâina pe baza de date.

TwoFish - este un algoritm cu blocuri simetrice de criptare , blocurile au dimensiune de 128 şi 256 biţi. Acesta a fost finalistul competiţiei *Advanced Encryption Standard contest*, dar nu a fost selectat ca şi câştigător. Avantajul faţă de AES constă în faptul că acesta este mai rapid pentru chei de 256 biţi. Librăria *crypto* din Golang nu conţine o implementare pentru chei de 256 de biţi, din acest motiv aceasta a fost implementată ca utilitar pentru acest proiect. Algoritmul a fost testat pentru consistenţă şi eficienţă, rezultatele de performanţă sunt asemănătoare cu cele din analiza algoritmului.

Nivelul de Logică de Business conţine în general validări ale parametrilor, calibrarea algoritmilor. În cazul în care parametrii primiţi nu sunt valizi sistemul aruna o eroare, iar comenzile nu sunt propagate la nivelul de mai jos.

Nivelul de gRPC API encapsulează implementarea interfeţei de serviciu care a fost definită în fişierul *.proto*. Nivelul este responsabil pentru parsarea şi extragerea parametrilor din apelurile de proceduri. De asemnea, acest nivel este responsabil pentru translatare API-urilor gRPC în REST.

Serviciul are responsabilităţile bine definite, separate şi conectate în mod transparent.

5.1.5 Microserviciul de steganografie şi marcare

Steganografia este ştiinţa ascunderii mesajelor, numele provenind de la cuvintele greseşti *steganos*, care înseamnă protejat, ascuns, şi *graphein*, care înseamnă a scrie. În Figura 5.9 este reprezentată arhitectura serviciului de steganografie şi watermarking.

Serviciul este ogranizat pe nivele: nivelul de gRPC API, nivelul de Logică de busuness şi nivelul de steganografie şi marcaje.

Nivelul de steganografie şi marcaje are ca responsabilitate înserarea unor date în imaginea sau textul dorit. Funcţionalitatea de steganografie are rolul de a însera un mesaj într-o imagine fără ca acesta să fie vizibil cu ochiul liber.

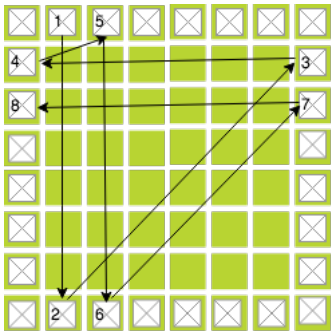

Figura 5.8: Exemplu de encodare a unui byte

Dezavantajele steganografiei constau în faptul ca pot fi analizate frecvenţele biţiolor şi se paote determina algoritmul folosit, iar odată descoperit acesta nu se poate folosi. Pentru a scădea riscul în cazul unei analize a biţilor a fost ales un algoritm care encodează date în cel mai puţin semnificativ bit folosind o tehnică de spiralare alternativă, tehnica de encodare este prezentată în Figura 5.8.

Funcţionalitatea de watermarking oferă posibilitatea de aplicare a unor mesaje text sau a altor imagini pe imaginea iniţială. Pentru aplicarea mesajelor text a fost definit un serviciu python care realizează conversia unui textla o imagine PNG cu ajutorul librăriei Pillow[2].

Figura 5.9: Arhitectura serviciului de marcaje

[2]https://pillow.readthedocs.io/en/stable/

Nivelul de logică de bussines are ca scop validarea datelor şi luarea deciziilor de conversie a datelor în funcţie de tipul acestora. La acest nivel se face conexiunea cu serviciul de python şi se prelucrează răspunsul acestuia. În cazul unei erori la acest nivel nu se va executa un apel la nivelul inferior, rezultatul operaţiei fiind o doar o eroare. La acest nivel este definit şi implementat clientul pentru aplicaţia de conversie a imaginilor.

Nivelul de API gRPC este responsabil pentru tartarea apelurilor de proceduri, acesta implementează interfaţa de server definită în fişierul *.proto*. La acest nivel se degineşte protocolul de maparea a procedurilor gRPC la funcţionalităţi care pot fi apelate prin REST.

5.1.6 Microserviciul de compresie

Microserviciul de compresie oferă 3 funcţionalităţi: compresie text, compresie PNG şi compresie JPEG. Arhitectura acestuia este prezentată în Figura 5.10.

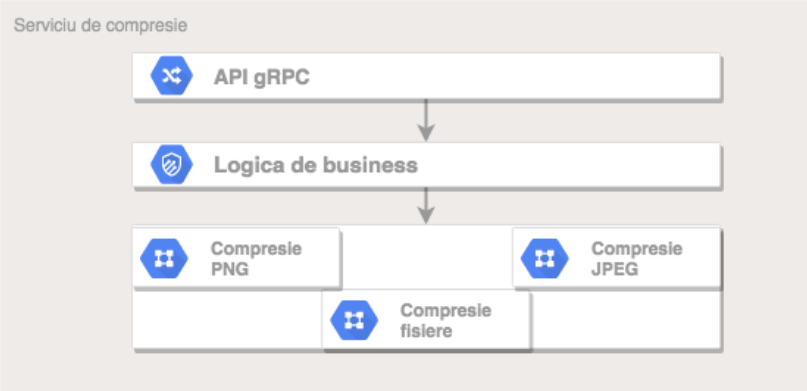

Figura 5.10: Arhitectura serviciului de compressie

Microserviciul de compresie este organizat pe 3 nivele: nivelul de API gRPC, nivelul de logică de business şi nivelul de utilitare de compresie.

Nivelul de API gRPC implementează protocolul de comunicare definit în fişierul protobuf, acesta este responsabil pentru parsarea parametrilor şi translatarea apelurilor gRPC la apeluri de tip REST. Tot la acest nivel sunt efectuate translatările de tip a datelor de intrare şi ieşire.

Nivelul de logică de business are ca scop definirea unui protocol de interacţiune a API-ului cu utilitarele de compresie. La acest nivel este implementat şablonul de proiectare *Strategy*[27]. Scopul pentru care a fost definit acesta este selecţia algoritmului de criptare dorit: compresie cu pierderi JPEG sau compresie fără pierderi utilizând algoritmul DEFLATE.

La nivelul de utilitare sunt implementaţi cei 3 algoritmi de compresie: Zlib pentru imagini, Zlib pentru texte şi compresia cu pierderi pentru imagini JPEG.

Compresia de text şi imagini PNG este bazată pe zlib care foloseşte algoritmul DEFLATE, acesta permite utilizarea unui număr minim de resurse pentru a obţine un rezultat cât mai bun. Algoritmul este unul fără pierderi. Rezultatul bun este obţinut datorită folosirii arborilor de codificare Huffman, deorece se crează un arbore Huffman optimizat pentru fiecare block de date. Totuşi folosirea Huffman este recomandată mai mult pentru mesaje mici, deorece aceasta introduce pentru fiecare block nişte instrucţiuni de decompresie. Compresia este obţinută prin 2 paşi:

- Potrivirea şi înlocuirea datelor duplicate
- Înlocuirea simbolurilor cu altele noi bazate pe frecvenţa de apariţie

În tabelul B.1 este prezentată performanţa algoritmului de compresie implementat. Se poate observa faptul că dimensiunea de după compresie a unui text ce conţine date repetitive poate fi de până la 40 de ori mai mică decât dimensiunea textului iniţial, iar în cazul textelor cu entropie mare dimensiune se înjumătţeşte. Datorită algoritmilor de compresie se poate obţine un coeficient câştig de spaţiu şi costuri între 2 şi 40, lucru care este favorabil atât pentru utilizator cât şi pentru dezvoltatorii software.Un alt avantaj adus de compresie este creşterea entropiei datelor, fapt datorită căruia criptarea devine mai eficientă iar algoritmul devine mult mai puţin vulnerabil al atacurile cibernetice. Dezavantajul principal al compresiei este complexitatea temporară adăugată, din cauza procesului se adaugă întârzieri detsul de mari care pot să deranjeze utilizatorul.

În cazul imaginilor cu o entropie mică, de exemplu cele care contin text sau au un procent de fundal foarte mare se obţine o rată de compresie foarte bună cu un factor cuprins între 20 şi 1000, fapt care se poate observa din Tabelul B.2. Acest lucru contribuie la scăderea preţului de stocare per octet, însă introduce un cost mediu spre mare de timp pentru imaginile foarte mari.

În cazul imaginilor cu o entropie mare compresia creşte dimensiunea fişierului prin adăugarea de metadate, iar dimensiunea datelor rămăne aceeaşi. În cazul unor imagini foarte mari de acest tip se poate introduce o întărziere de pănă la 35 de secunde fără a aduce impact asupra preţului de stocare. Însă aceste imagini se întâlnesc extrem de rar, aplicaţia ar fi afectată foarte puţin de acest fenomen, o analiză detaliată a acestui lucru este prezentată în Tabelul B.3.

5.1.7 Microserviciul de fisiere

Microserviciul de fişiere are o arhitectură organizată pe nivele, aceasta este prezentată în Figura 5.11. Responsabilitatea serviciului este tratarea tututor operaţiilor efectuate asupra fişierelor şi comunicarea cu serviciile de criptare, compresie şi steganografie.

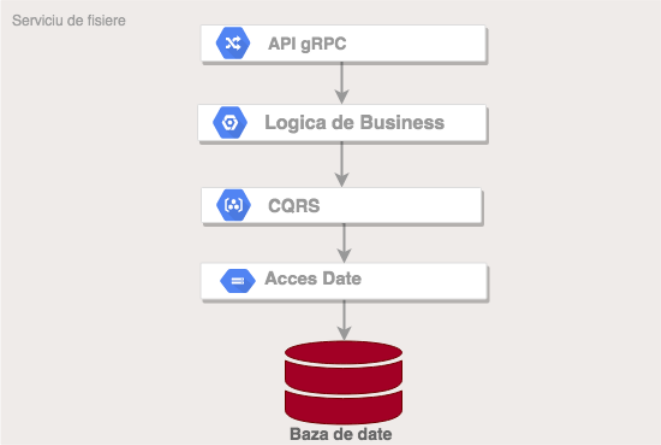

Figura 5.11: Arhitectura serviciului de fişiere

Serviciul este organizat pe 4 nivele: API gRPC, BLL, CQRS şi accesul la date. Fiecare nivel are funcţionalităţi bine definite şi separatae în aşa fel încât serviciul să respecte toate principiile SOLID[28].

Nivelul de logică de business are ca responsabilitate validarea datelor şi comunicarea cu microserviciile auxiliare, cu scopul de a efectua operaţiile de compresie, criptare şi steganografie. Pentru fiecare apel la un alt serviciu se crează o conexiune gRPC şi se instanţiază un client prin care se face comunicarea cu serviciul. Impactul operaţiei asupra performanţei este minim, deoarece apelurile se efectuează foarte rapid datorită protocolului binar definit în HTTP/2.

La nivelul CQRS*(Command and Query Responsibility Segregation)*[29] este implementat şablonul *Command-Event*. Scopul acestui nivel este definirea a 2 modele de date: modelul de scriere şi modelul de citire. Separarea celor 2 operaţii aduce un avantaj considerabil în viteza de procesare a datelor, însă a introdus complexitate de tratare a operaţiilor. Principala provocare a fost implementarea asincronă a creării şi tratării evenimentelor din sistem.

Golang, fiind un limbaj concurent, nu permite efectuarea operaţiilor în paralel, însă datorită go-rutinelor operaţiile concurente devin rapide şi eficiente. Pentru fiecare apel al utilizatorului se crează o comandă care este transmisă printr-un canal către nivelul de baze de date, ajuns în coada de procesare acesta declanşează un eveniment de interacţiune cu baza de date şi produce un răspuns acre este propagat înapoi către client. Avantajul principal al acestui nivel este abstractizarea operaţiilor, însă acesta introduce posibile riscuri de inconsistenţă a datelor.

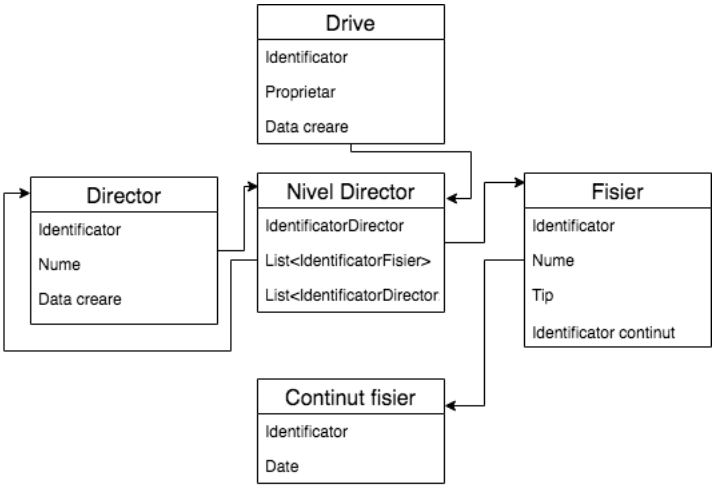

Figura 5.12: Structura organizării datelor serviciului de fişiere

Pentru nivelul de bază de date a fost folosit un pachet Go destinat pentru interacţiunea cu Couchbase. La acest nivel sunt efectuate operaţii complexe de scriere şi citire. Datele sunt decuplate cu scopul de a obţine viteze de citire şi transmisie cât mai mari. Modelul de date este prezentat în Figura 5.12, cu scopul de prezentare s-a ignorat aspectul non-relaţional al bazei de date, legăturile nu prezintă altceva decât legături logice între date.

Serviciul conţine o logică complexă de efectuare a operaţiilor şi de interacţiune cu alte servicii. Acest microserviciu expune un set de *endpoint*-uri REST şi gRPC apelabile din aplicaţia client, definiţiile endpoint-urilor sunt prezentate în Tabelul 5.1.

Fiecare endpoint poate fi apelat atât prin REST, cât şi prin gRPC. Acest lucrua fost realizat cu scopul de a fi compatibil cu mai multe tipuri de clienţi, deoarece gRPC nu este disponibil sau bine documentat pentru multe limbaje folosite în prezent o abordare REST ar putea soluţiona problema mult mai rapid. După rularea serviciului conform paşilor din Capitolul 7, serviciul va fi diponibil la adresa *localhost:5005* pentru apeluri REST şi la adresa *localhost:4005* pentru apeluri gRPC.

5.1.8 Microserviciul de utilizatori

Microserviciul de utilizator encapsulează toate funcţionalităţile pentru managementul conturilor de utilizatori. Pentru o mai bună separare şi nivel de cuplare mai scăzut, serviciul a fost structurat pe nivele, arhitectura serviciului este prezentată în Figura 5.13.

Nivelul de acces BD conţine metodele pentru scrierea şi citirea datelor. Pentru a utiliza MongoDB într-un mod cât mai simplu şi eficient a fost utilizat driver-ul oficial adaptat pentru Golang. Structura bazei de date este prezentată în Figura 5.14, După cum se poate observa, nu există nici o legătură între colecţia *User* şi colecţia *Session*.

METODĂ	URL REST	Parametri REST	Nume GRPC	Parametri gRPC
POST	/drive/create	userID	CreateDrive	CreateDriveRequest
POST	/file/create	content, name, secretPhrase, destinationID	CreateFile	CreateFile -Request
POST	/file/upload	content, name, secretPhrase, destinationID	UploadFile	UploadFileRequest
POST	/folder/create	name, destinationID	CreateFolder	CreateFolder -Request
GET	/folder/content	folderID	GetFolderContent	GetFolderContent -Request
GET	/file/content	fileID, secretPhrase	GetFileContent	GetFileContent -Request
GET	/drive/tree	driveID	GetFileTree	GetFileTree -Request
PUT	/file/update	fileID, content, secretPhrase	UpdateFile -Content	UpdateFileContent -Request
DELETE	/file/delete	fileID	DeleteFile	DeleteFileRequest
PUT	/file/rename	fileID, name	RenameFile	RenameFileRequest
PUT	/file/move	fileID, destination	MoveFile	MoveFileRequest
GET	/file/download	fileID, secretPhrase, watermarkingData steganographyData	DownloadFile	DownloadFile -Request
GET	/file/size	userID	ComputeSize	ComputeSize -Request
GET	/drive/id	userID	GetMyDriveId	GetMyDriveId -Request

Tabelul 5.1: Endpoint-urile serviciului de fişiere

Un document din colecţia de sesiuni conţine doar valoarea token-ului, motivul fiind faptul că pentru crearea unui token se folosesc datele personale ale utilizatorului: nume utilizator, identificator, rol şi nume, precum şi timpul de expirare a token-ului. Datele sunt decodate atunci când sunt necesare.

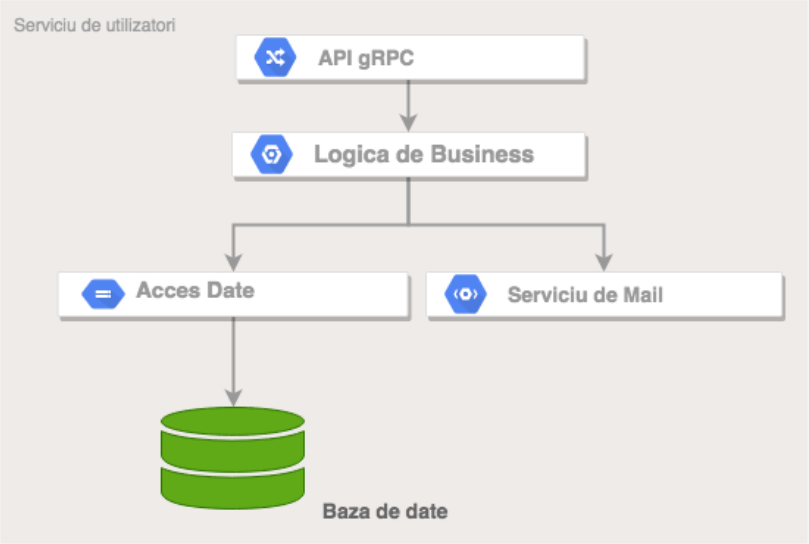

Figura 5.13: Arhitectura serviciului de utilizatori

Pentru înregistrare este necesar un e-mail valid, deoarece utilizatorul trebuie să îşi confirme înregistrarea prin accesarea unui link primit prin intermediul mail-ului la momentul completării formularului. Serviciul de mail a fost realizat utilizând pachetul *net/smtp* disponbil în Golang.

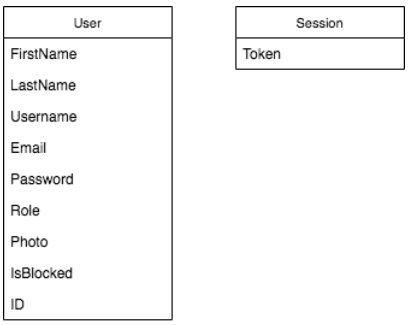

Figura 5.14: Structura bazei de date a serviciului de utilizatori

În Tabelul 5.2 sunt enumerate end-point-urile disponibile ale serviciului. Este prezentată atât descrierea procedurilor gRPC, cât şi cea a metodelor REST.

Nivelul de logică de business conţine toate validările rolurilor şi a permisiunilor utilizatorilor. La acest nivel se face conexiunea şi interacţiunea cu serviciul de criptare. Fiecare endpoint poate fi apelat atât prin REST, cât şi prin gRPC. Acest lucrua fost realizat cu scopul de a fi compatibil cu mai multe tipuri de clienţi ai aplicaţiei, în prezent o abordare REST ar putea soluţiona problema adăugării unui nou client mult mai rapid,

METODĂ	URL REST	Parametri REST	Nume GRPC	Parametri gRPC
POST	/account/create	username passsword email firstName lastName photo	CreateAccount	CreateAccountRequest
PUT	/account/update	token firstName lastName photo	UpdateAccount	UpdateAccount -Request
GET	/login	username password	Login	LoginRequest
GET	/logout	token	Logout	LogoutRequest
DELETE	/account/delete	token	DeleteAccount	DeleteAccountRequest
PUT	/account/lock	token	LockAccount	LockAccountRequest
PUT	/account/unclock	token	UnlockAccount	UnlockAccountRequest
GET	/acount/get	token	GetAccount	GetAccount -Request
GET	/account/list	token	ListUsers	ListUsers -Request

Tabelul 5.2: Endpoint-urile serviciului de utilizator

deaorece nu necesită configurări suplimentare. După rularea serviciului conform paşilor din Capitolul 7, serverul va fi diponibil la adresa *localhost:5008* pentru apeluri REST şi la adresa *localhost:4008* pentru apeluri gRPC.

5.2 Arhitectura aplicatiei web

Aplicaţia web a fost scrisă utilizând framework-ul Vue, aceasta are rolul de client pentru suita de servicii oferind utilizatorului posibilitatea de a utiliza aplicaţia din Browser. În Figura 5.15 este prezentată arhitectura clientului, acesta a fost structurat ca 3 microservicii separate. Fiecare microserviciu client comunică cu microserviciile serverului prin gRPC Gateway construită cu ajutorul Envoy.

Comunicarea cu serviciile din *backend* este efectuată prin protocolul gRPC. Pentru crearea clientului, fişierul *.proto* a fost compilat utilizând comenzile:

```
>set -ex
>mkdir -p ../../../../src/pb/
>protoc -I=.
        --js_out=import_style=commonjs,binary:./../../../../src/pb/
        --grpc-web_out=import_style=commonjs,mode=grpcwebtext:./../../../../src/pb/
         file_service.proto user_service.proto
```

La ieşire se generează câte 2 fişiere de Javascript pentru fiecare fişier Protobuf compilat. Acestea conţin definiţia parametrilor de intrare şi de ieşire ai metodelor, precum şi definiţia metodelor gRPC. Pentru apelarea metodelor serverului este necesară instanţierea unui client cu conexiune la aelaşi port ca şi serverul, iar prin apelul metodelor client se execută direct apelul metodelor server.

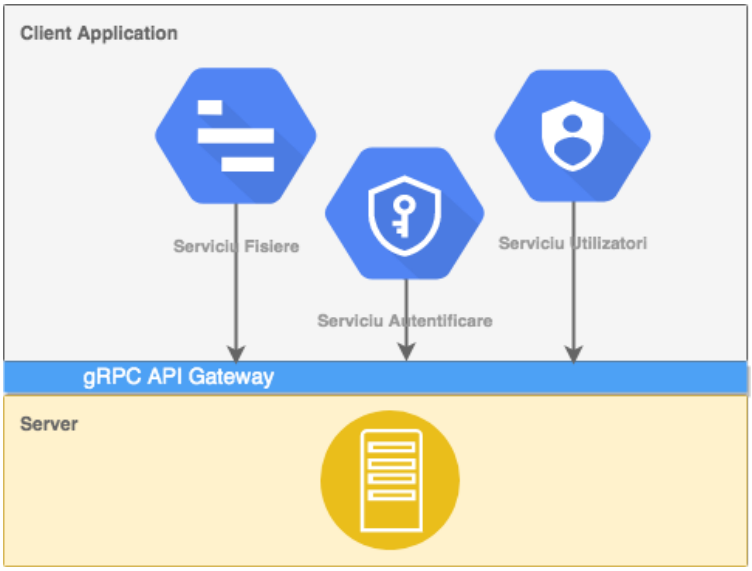

Figura 5.15: Arhitectura clientului

Pentru fiecare funcţionalitate au fost definite interfeţe separate, pentru definirea lor au fost folosite limbajele: Javascript, TypeScript, CSS şi HTML.

Capitolul 6

Testare şi Validare

Acest capitol conţine descrierea metodei de testare şi validate a sistemului. Sunt descrise toolurile care au fost folosite pentur testarea serverului şi a clientului. Deasemenea este analizatrocentul de acoperire a codului prin teste.

6.1 Testarea serverului

Pentru testarea serverului au fost scrise atât *Unit Teste*, cât şi *Teste de integrare şi interacţiune*. Pentru scrierea şi executarea testelor au fost folosite 2 *tool*-uri: *testing*, care este integrat în limbajul **Golang**, şi *testify*, care este o librărie *open-source* pentru testare în limbajul menţionat. Pentru o mai bună testare, înainte de dezvoltarea clientului *web*, s-au efectuat operaţii de testare utilizând fişierele *swagger* create pe baza API-ului serviciilor, şi utilizând tool-ul *Postman*.

În teste au fost acoperite toate cazurile de succes şi majoritatea cazurilor în care ar trebui să se producă o eroare, cum ar fi: parametri invalizi, parametri cu valori eronate sau precondiţii nesatisfăcute.

6.1.1 Reguli de testare în Golang

Pentru ca testele să poată fi executate automat, numele fiecărui fişier ar trebui să conţină sufixul *_test*. Fiecare metodă de test ar trebui să înceapă cu prefixul *Test*. Îndeplinirea acestor condiţii permite rularea automată a testelor într-un mediu de *Dezvoltare continuă*.

Scrierea unor teste bune nu este un lucru trivial, în multe situaţii sunt foarte multe cazuri de acoperit, ceea ce înseamnă că trebuie scrise multe teste, **Golang** rezolvă această problemă prin adăugarea de *teste tabelare*, acestea permit crearea testelor imbricate, care pot fi rulate în paralel, asfel fiind mai uşor de detectat *"data race"*. Acest tip de testare are un avantaj enorm deorece permite scrierea unui cod citibil şi foarte uşor de înţeles. Fiind integrate în limbaj, acest tip de teste pot fi create înainte de cod, imadiat după definirea semnăturilor de metode, ceea ce este ideal pentru o abordare *Test Driven Developement*.

De asemnenea, **Golang** permite adăgarea de funcţii ajutătoare, a căror ordine şi frecvenţă de rulare poate fi condiţionată, acestea pot fi precondiţii/postcondiţii globale sau locale.

6.1.2 Testarea serviciilor

Pentru fiecare serviciu au fost generate un numar de teste egal sau mai mare cu numărul de metode pe care le are serviciul, în fiecare test au fost definite seturi tabelare de date pentru a acoperi cât mai multe cazuri. În secţiunea de mai jos este prezentat un exemplu de test pentru iniţierea unui obiect.

```
func TestNewServer(t *testing.T) {
        type args struct {
                service *service.AuthService
        }
        // Definirea cazurilor de test si a parametrilor utilizati si valorile asteptate
        tests := []struct {
                name string
                args args
                want *server.AuthServer
        }{
                // Primul caz de test cu parametrii nuli
                {
                        name: "Nil",
                        args: args{
                                service: nil,
                        },
                        want: &server.AuthServer{
                                Service: nil,
                        },
                },
                // Al doilea caz de test, cu parametri valizi
                {
                        name: "Not nil",
                        args: args{
                                service: service.NewAuthService(),
                        },
                        want: &server.AuthServer{
                                Service: service.NewAuthService(),
                        },
                },
        }

        for _, tt := range tests {
                // Pentru fiecare caz de test din structura tabelara definita, se ruleaza un nou test,
                //in care se apeleaza metoda testata din cadrul pachetului.
                t.Run(tt.name, func(t *testing.T) {
                // Se verifica prin reflectie daca datele obtinute sunt egale cu cele asteptate
                        if got := server.NewServer(tt.args.service); !reflect.DeepEqual(got, tt.want) {
                                t.Errorf("NewServer() = %v, want %v", got, tt.want)
                        }
                })
        }
}
```

Rata de acoperire a codului a fost măsurată cu ajutorul unui *tool* care este integrat în limbaj, aceasta se numeşte *Go Cover*. În tabelul 6.1 este reprezentată rata de acoperire

cu teste a fiecărui serviciu în parte, unele părţi nu au putut fi acoperite, acestea conţinând tratări de erori care vin din alte librării externe ce au fost folosite, o situaţie de apariţie a acestora nu a putut fi simulată.

Tabelul 6.1: Rata de acoperire a testelor

Nume serviciu	Acoperire API	Acoperire BLL	Acoperire alte pachete
Serviciu autentificare	84%	81%	86%
Serviciu compresie	96%	95%	90%
Serviciu criptare	81%	90%	97%
Serviciu fişiere	90%	80%	99%
Serviciu staganografie	81%	77%	86%
Serviciu utilizatori	85%	83%	60%

Pentru fiecare metodă şi funcţionalitate a serverului au fost generate toate combinaţiile posibile de date, valide şi invalide, pentru a detecta cât mai multe probleme şi a le fixa, sau pentru a avea siguranţa că totul funcţionează cum este de aşteptat.

Testarea adecvată a avut un rol semnificativ în procesul de detecţie de erori sau comportament neaşteptat. Datorită testării au fost detectate probleme triviale de securitate, care puteau compromite identitatea şi datele utilizatorului.

S-au efectuat teste de performanţă pentru algoritmii de compresie pentru diferite tipuri de date. Datele au fost selectate după netropie şi după dimensiune, rezultatele testelor pot fi găsite în Tabelul B.3, Tabelul B.2 şi în Tabelul B.1. Din aceste rezultate se poate deduce faptul că algoritmii sunt eficienţi şi utili pe date uzuale, cu caracter repetitiv. În cazul datelor cu mare algoritmii măresc dimensiunea datelor de ieşire prin adăugarea meta datelor specifice.

Pentru algoritmii de criptare s-au efectuat teste de consistenţa a datelor, au fost testate următoarele aspecte:

- Datele decriptate sunt egale cu datele iniţiale ale algoritmului de criptare.
- Nu se păstrează pattern în date.
- La aplicarea algoritmului de hashing pe aceleaşi date avem mereu acelaşi rezultat, testul a avut 10000 de repetiţii
- Pentru hashing-ul diferitor seturi de date nu se obţin coliziuni, au fost testate circa 5000 de inputuri diferite.
- La aplicarea repetată a aceluiaşi algoritm pe date nu se obţine mesajul iniţial.
- La aplicarea algoritmului de hashing cu parametri diferiţi se obţin hash-uri diferite.

Toate testele de consistenţă ale algoritmilor au trecut cu succes, lucru care denotă ca algoritmii au fost implementaţi bine şi securitatea nu poate fi compromisă din cauza unor erori de implementare.

6.2 Testarea clientului

Pentru testarea aplicaţiei client s-au efectuat 3 tipuri de teste: teste manuale, e2e şi Unit teste.

Cu ajutorul Unit testelor s-a testat funcţionarea corectă a modulelor aplicaţiei şi a logicii de prelucrare a datelor. Pentru fiecare funcţionalitate au fost scrise teste care ar trebui săurmeze calea de succes şi teste care conţin date eronate şi ar trebui să producă eroare, fără a modifica starea aplicaţiei, şi fără a cauza erori în funcţionare.

Pentru partea de teste e2e a aplicaţiei client s-a folosit frmework-ul *Nightwath*, acesta este un *framework* scris în *Node.js* şi este o soluţie performantă pentru testarea aplicaţiilor web. *Nightwatch* este potrivit pentru proiectele care adoptă calea livrării continue. Cu ajutorul acestui framework am testat interacţiunea clientului cu serverul şi că toate acţiunile şi toate datele se propagă corect între module şi servicii.

Testele se bazează pe cazurile de utilizare şi sunt o simulare a acestora. În Figura 6.1 este reprezentat un test bazat pe cazul de utilizare descris în sectiunea 4.4.

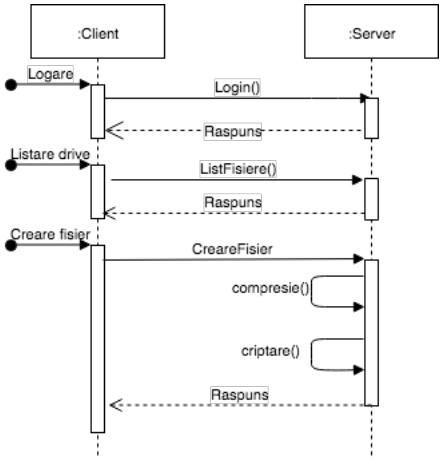

Figura 6.1: Diagrama de secvenţă test e2e

Prin testare manuală s-a testat modul de interacţiune a aplicaţiei cu utilizatorul. Acest lucru s-a efectuat prin parcurgerea căilor de evenimente posibile, erorile s-au difuncţiunile detectate devenind noi sarcini de îndeplinit în procesul de dezvoltare a proiectlui.

Capitolul 7

Manual de Instalare şi Utilizare

În acest capitol sunt prezentate cerinţele şi paşii de configurare a aplicaţie din perspectiva unui dezvoltator software şi paşii de utilizare din perspectiva unui client al sistemului.

7.1 Cerinţe sistem

Pentru a obţine un mediu optim pentru rularea aplicaţiei, calculatorul dezvoltatorului software trebuie să îndeplinească următoarele cerinţe:

- **CPU**: Intel Core i5 sau mai mult
- **Frecvenţă**: 1.8 GHz
- **RAM** : 8 GB
- **Sistem de operare**: Linux sau OS X
- **SSD/HDD**:256 GB
- **Browser**: Google Chrome, Safari sau Mozilla Firefox

Pentru clientul aplicaţiei web, cerinţele se limitează la cerinţele browser-ului şi cele de RAM.

7.2 Instalare si configurare

Atât aplicaţia client, cât şi cea server sunt hostate pe GitHub, acestea se află în repository-uri separate. Toate sub-proiectele necesare se găsesc reunite în aceeaşi organizaţie. Primul pas constă în clonarea fiecărui proiect pe maşina locală, fiecare

subproiect conţine un fişier în care sunt descrise cerinţele de configurare. În primul rând se vor configura microserviciile pentru back-end, iar apoi se va configura aplicaţia client.

În continuare vor fi descrise etapele de configurare ale ambelor părţi ale proiectului.

Pasul 1 **Obţinerea unei copii locale a proiectelor**

- Se va descărca şi instala GIT[1]
- Se va accesa **bitstored**, toate repository sunt publice şi disponibile în organizaţia respectivă
- Se vor clona toate proiectele listate în organizaţie într-un director local. Pentru clonare se pot utila clienţi de Git sau se poate rula comanda **git clone $REPO_URL** pentru fiecare subproiect

Pasul 2 **Instalarea şi configurarea bazelor de date**

- Se va instala şi configura **Couchbase**[2], conform paşilor din tutorialul de la adresa următoare *https://docs.couchbase.com/server/4.1/getting-started/installing.html*
- Se va crea un nou cluster, folosind scriptul din fişierul *server/repository/scripts/create_cluster.sh*
- Serverul de bază de date va fi accesibil la adresa *localhost:8091*, din interfaţa de administrator se pot gestiona datele, se pot crea noi clustere şi se pot vedea metrici.
- Pentru a putea utiliza Couchbase ORM din limbajul Golang, se va instala următoarea librărie

```
> go get -v github.com/couchbase/gocb
```

- Se va instala şi configura **MongoDB v4.x**[3], pentru instalare se vor urmări paşii de la adresa *https://docs.mongodb.com/manual/installation/*
- Opţional se va instala Robo3T[4], acest IDE permite vizualizarea, monitorizarea şi editarea înregistrărilor din baza de date Mongo
- Se va porni severul de MongoDB utilizând comenzile:

```
> docker run --rm -it -p 27017:27017 mongo
# sau pentru rularea pe masina fizica
> mongod
```

[1]https://git-scm.com/
[2]https://couchbase.com
[3]https://mongodb.com
[4]https://robomongo.org/

- O instanţă de MongoDB va fi accesibilă la adresa *localhost:27017*
- Pentru a fi accesibilă din Golang, se va instala un pachet MongoORM + Driver pentru Go, se va utiliza comanda ce urmează

```
> go get -v go.mongodb.org/mongo-driver/
```

Pasul 3 **Configurarea şi rularea aplicaţiei server**

- Se va instala **Golang**[5], versiune mai nouă de 1.12
- Se va instala **Visual Studio Code**[6] şi se vor adăuga extensii de Golang, sau se va instala **Vim** şi se va configura pentru suportul libajului Golang, sau se va instala editorul dedicat**Goland**[7]
- Se va descărca şi instala **Docker**[8]
- Se va instala Vue.JS[9] urmând paşii din următorul tutorial *https://vuejs.org/v2/guide/installation.html*
- Se va instala **Envoy**[10]
- Se va instala compilatorul de fişiere proto numit **protoc**[11]
- Înainte de rularea proiectului se vor instala, configura şi rula bazele de date necesare, conform descrierii de la Pasul 2
- Se va rula containerul de *Envoy*, se va deschide o sesiune de terminal în directorul rădă cină a proiectuluii şi se vor rula comenzile:

```
> cd envoy
> docker build -t bitstored_envoy
> docker run --rm -it -p 8081:8081 bitstored_envoy
```

- Se va rula fiecare serviciu utilizând comenzile de mai jos:

```
> cd $SERVICE_NAME
> docker build -t $SERVICE_NAME
> docker run  --network=host  $SERVICE_NAME
```

  O alternativă pentru acest proces este rularea containerului de docker din rădăcina proiectului, utilizând comenzile:

[5]https://golang.org/doc/install
[6]https://code.visualstudio.com/
[7]https://www.jetbrains.com/go/
[8]https://docs.docker.com/install/
[9]https://vuejs.org/
[10]https://www.envoyproxy.io/docs/envoy/latest/install/install
[11]http://google.github.io/proto-lens/installing-protoc.html

```
> docker build -t server
> docker run  server
```

Pasul 4 **Configurarea şi rularea aplicaţiei client**

- Se va deschide o sesiune nouă de terminal şi se va naviga în directorul aplicaţiei client
- Se va rula aplicaţia, utilizând comenzile:

```
> npm install
> npm run
```

- Aplicaţia Web va fi disponibilă local la adresa *localhost:8080*

7.3 Instrucţiuni de utilizare

În această secţiune se va face o prezentare generală a aplicaţiei, se va prezenta conţinutul web al fiecărei pagini care poate fi accesată de utilizatori, în funcţi de rol: utilizator neîngregistrat, utilizator autentificat sau administrator. Va fi prezentat arborele de naviigare a aplicaţiei, cu scopul de a obţine o viziune asupra cazurilor de utilizare.

7.3.1 Arborele de navigare

În Figura 7.1 modul în care paginile aplicaţiei sunt conectate între ele şi ce tip de utilizatori care le pot accesa.

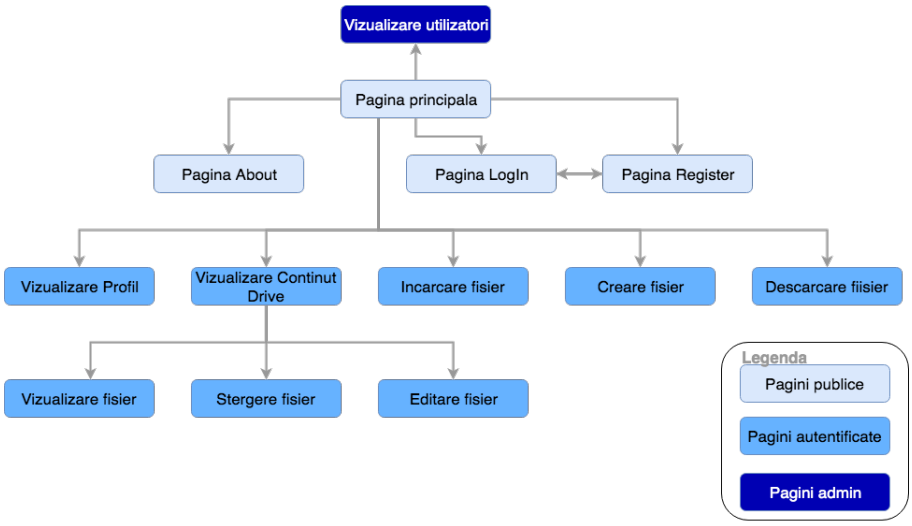

Figura 7.1: Arborele de navigare al aplicaţiei

Această schemă a fost utilizată în etapa de design, oferind informaţii generale despre modul de contruire a interfeţei utilizator. După cum se poate observa, numărul de pagini publice se rezumă la login şi register. Pentru a accesa oricare altă pagină, utilizatorul va trebui să se autentifice.

7.4 Navigare

Primul pas pentru utilizarea aplicaţiei este crearea unui cont. De la pagina de start, se va apăsa butonul register, conform Figurii 7.2.

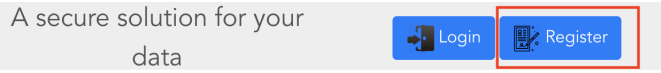

Figura 7.2: Buttonul de Register

După apăsarea butonului se va deschide o nouă pagină cu un formular, fiecare câmp este obligatoriu.

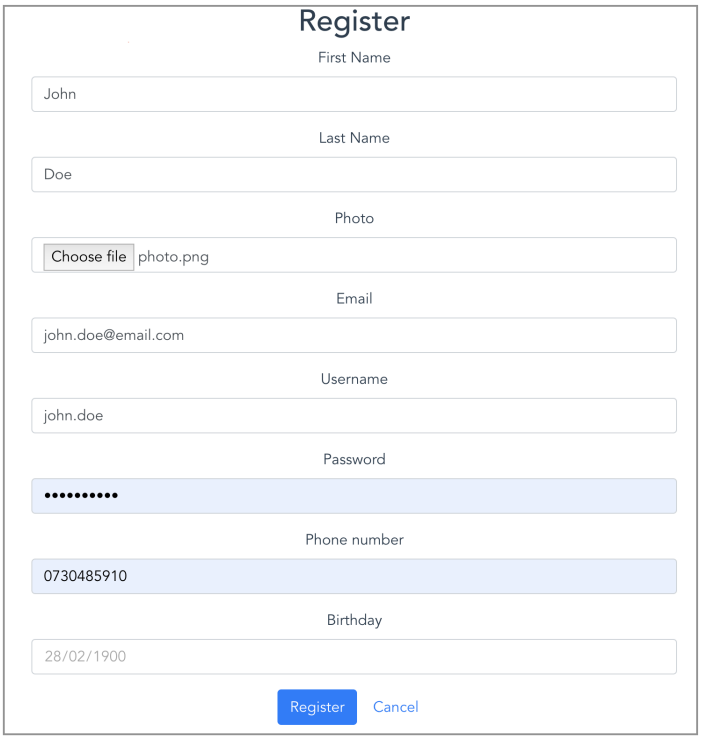

Figura 7.3: Exemplu înregistrare

Condiţia principală pentru ca o înregistrare să fie posibilă este ca email-ul, numele de utilizator şi numărul de telefon să nu fie înregistrate deja în sistem. În Figura 7.3 este prezentat un exemplu de date valide.

După ce utilizatorul este autentificat, acesta poate alege din meniul aplicaţiei spre ce pagină doreşte să navigheze. Pentru a efectua operaţii cu fişiere se va naviga la pagina *My Drive*. În Figura 7.4 este reprezentată pagina principală a Drive-ului.

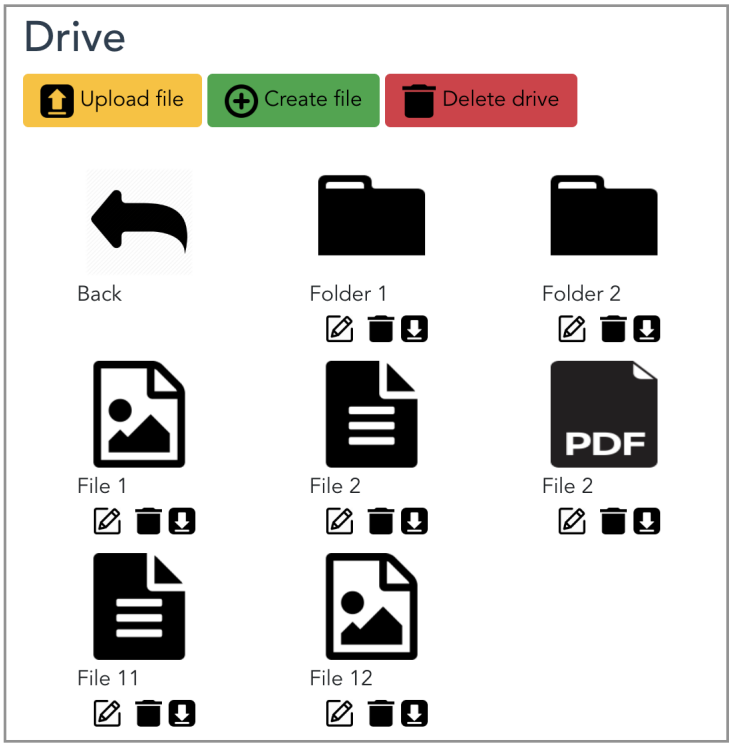

Figura 7.4: Vizualizare drive

Din pagina de mai sus utilizatorul poate selecta operaţia pe care doreşte să o efectueze: încărcare fişier, creare fişier sau creare director nou. De asemenea poate naviga în alte directoare printr-un click pe directorul dorit. Interfaţa este pe cât se poate de sugestivă, cu explicaţii pentru datele ce trebuie completate, iar navigarea în drive este similară cu navigarea în sistemul de fişiere al calculatorului.

Capitolul 8

Concluzii

În acest capitol vor fi descrise rezultatele obţinute, contribuţiile aduse şi ideile de dezvoltare ulterioară şi îmbunătăţire a aplicaţiei.

8.1 Rezultate obţinute

Scopul proiectului a fost dezvoltarea unei aplicaţii securizate de stocare a datelor. Cerinţele de bază au fost îndeplinite, aplicaţia având funcţionalităţile cheie ale unui sistem de stocare de fişiere clasic.

Comparativ cu sistemele analizate în Secţiunea 3.5, are un set de funcţionalităţi mai puţin numeros, dar implementarea acestora este bazată pe ideea menţinerii unui nivel înalt de securitate a datelor. Sistemul permite compresia eficientă a datelor, preţul de stocare scăzând astfel considerabil în comparaţie cu sistemele analizate în Secţiunea 3.5.

Aplicaţia obţinută oferă funcţionalităţi de încărcare, creare şi descărcare a fişierelor, acesta fiind un sistem cu zero cunoţinţe despre datele stocate în sistem, un atacator nu poate să pună mâna pe datele utilizatorului în lipsa parolei de decriptare, aceasta nefiind stocată în sistem.

Prin adăugarea compresiei fişierelor text şi a imaginilor s-a obtinut reducerea considerabilă a spaţiului de stocare în cazul fişierelor mari. De asemnea adăugarea compresiei a contribuit la creşterea securităţii aplicaţiei prin creşterea nivelului de complexitate pentru decriptare.

Funcţionalităţile de steganografie şi marcaje creează un sistem de adăugare a semnăturilor de autor şi este o măsură minimală de combatere a furtului de fişiere.

În etapă de dezvoltare curentă aplicaţia poate fi utilizată ca şi sistem de stocare sigur, vulnerabilităţile care pot apărea nu vor compromite datele fişierelor în mod normal, acets lucru se poate întâmpla doar din cauza imprudenţei utilizatorului.

8.2 Dezvoltări ulterioare

Proiectul are funcţionalităţile minime care îl încadrează în rândul aplicaţiilor de stocare a fişierelor. Pentru îmbunătăţirea modului de funcţionare se pot implementa următoarele idei:

- Deoarece operaţiile de criptare şi compresie, precum şi operaţiile inverse, sunt foarte costisitoare în cazul unor date mari ca volum şi cu rată de repetare mică, o soluţie care ar eficientiza procesul ar fi combinarea celor 2 operaţii în una singură. Acest efect se poate obţine prin adăugarea unor permutaţii pseudoaleatore în procesul de compresie al datelor[30].

- Se pot eficientiza funcţionalităţile de încărcare şi descărcare a datelor prin adăugarea operaţiei de despărţire în blocuri, acestea fiind transmise alternativ, operaţiile de transfer prin reţea sunt cunoscute a fi cele mai încete operaţii de manipulare a datelor. O astfel de abordare ar putea reduce extrem de mult operaţiile de transmisie, dar şi cele de criptare şi compresie, datorită faptului că operaţiile pe blocuri ar putea fi efectuate în paralel, în loc de serial cum este în versiunea curentă.

- Deoarece o parte dintre aplicaţiile existente sunt vulnerabile la execuţie de cod maliţios sau coruperi de date, o funcţionalitate care ar putea creşte nivelul de securitate al aplicaţiei ar fi adăugarea funcţionalităţii de antivirus. Astfel fiecare încărcare de fişier ar fi validată în prealabil şi fişierul ar ajunge în spaţiul de stocare doar după ce ar fi marcat ca unul sigur.

- Dat fiind faptul că mulţi hackeri utilizează atacurile prin forţă brută, aplicaţia poate fi expusă la acest tip de atacuri, o soluţie pentru acest pericol ar fi adăugarea autentificării în 2 paşi.

- O funcţionalitate de care poate beneficia aplicaţia este monitorizarea activităţii utilizatorilor şi analiza acesteia printr-un algoritm de inteligenţă artificială, astfel se pot bloca automat utilizatorii care au activităţi suspicioase sau se pot prezice viitoarele acţiuni sau procesul de creştere a volumului de date.

Bibliografie

[1] D. Reinsel, J. Gantz, and J. Rydning, "The digitization of the word from edge to core,' *IDC White Papper*, vol. 20, no. 18, pp. 1–27, 2018.

[2] P. Priyam, *Cloud Security Automation.* OReilly Media, Inc., 2018.

[3] P. Bruni, M. K. Boisen, G. D. Marchi, and F. Pinto, *Reduce Storage Occupancy and Increase Operations Efficiency with IBM zEnterprise Data Compression.* IBM, 2015.

[4] P. Lancett, "The advantages of file compression.' [Online]. Available: https://www.techwalla.com/articles/the-advantages-of-file-compression

[5] "Solid principles.' [Online]. Available: https://deviq.com/solid/

[6] N. Jackson, *Building Microservices with Go.* PacktPub, 2017, vol. 1.

[7] "An overview of monolithic vs microservices architecture.' [Online]. Available: https://www.bmc.com/blogs/microservices-architecture/

[8] C. Richardson, "Pattern: Microservice architecture.' [Online]. Available: https://microservices.io/patterns/microservices.html

[9] J. Lewis and M. Fowler, "Microservices,' *ThoughtWorks*, vol. 1, no. 1, pp. 1–20, 2014.

[10] D. Pacheco, *Building Effective Microservices.* Pakt Publishing, 2017.

[11] E. Griffor, *Handbook of System Safety and Security.* Syngress, 2016.

[12] J. D. Leon, *Security with Go.* Pakt Publishing, 2018.

[13] A. S. K. Pathan and S. Azad, *Practical Cryptography.* Auerbach Publications, 2014.

[14] B. Schneier, J. Kelsey, D. W. andDavid Wagner, C. Hall, and N. Ferguson, "Twofish: A 128-bit block cipher,' *Counterpane Systems*, vol. 1, no. 1, pp. 1–68, 1998.

[15] R. Zabicki, *Practical Security.* Pragmatic Bookshelf, 2019.

[16] A. Haecky and C. McAnlis, *Understanding Compression.* OReilly Media, Inc., 2016.

[17] K. Sayood, *Introduction to Data Compression, 4th Edition.* Morgan Kaufmann, 2012.

[18] C.-N. Yang and S. Cimato, *Visual Cryptography and Secret Image Sharing.* CRC Press, 2017, vol. 1.

[19] "Best cloud storage.' [Online]. Available: https://www.techradar.com/news/the-best-cloud-storage

[20] M. Borgmann, T. Hahn, M. Herfert, T. Kunz, M. Richter, U. Viebeg, and S. Vowe, *On the Security of Cloud Storage Services.* Fraunhofer Institute for Secure Information Technology SIT, 2012.

[21] "Cloudme documentation.' [Online]. Available: https://www.cloudme.com/

[22] "Cloudwards - best cloud storage.' [Online]. Available: https://www.cloudwards.net/best-cloud-storage/

[23] "Crashplan review.' [Online]. Available: https://www.bestbackups.com/blog/5210/crashplan-review-2/

[24] "Icloud drive review.' [Online]. Available: https://www.cloudwards.net/review/icloud-drive/

[25] "Most secure cloud storage 2019: Safety first.' [Online]. Available: https://www.cloudwards.net/most-secure-cloud-storage/

[26] H. Potsangbam, *Learning Couchbase.* PacktPub, 2015, vol. 1.

[27] E. Gamma, J. Vlissides, R. Johnson, and R. Helm, *Design Patterns Elements of Reusable Object Oriented Software.* Addison Wesley, 1994, vol. 1.

[28] G. M. Hall, *Adaptive Code Agile Coding with Design Patterns and SOLID Principles.* Microsoft, 2017, vol. 2.

[29] D. Betts, F. Simonazzi, G. Melnik, J. Dominguez, and M. Subramanian, *Exploring CQRS and Event Sourcing.* Microsoft, 2012, vol. 1.

[30] C.-E. Wang, "Cryptography in data compression,' *CodeBreakers Journal*, vol. 2, no. 3, pp. 1–20, 2005.

Anexa A

Secţiuni relevante din cod

A.1 Exemplu de fişier *protobuf*

```
syntax = "proto3";
package compression_service;
option go_package="pb";
import "google/api/annotations.proto";

service Compression {
    rpc CompressImage(CompressImageRequest) returns (CompressImageResponse) {
         option (google.api.http) = {
             get: "/compression/api/v1/image/compress"
        };
    }
     rpc DecompressImage(DecompressImageRequest)   returns (DecompressImageResponse) {
         option (google.api.http) = {
            get: "/compression/api/v1/image/decompress"
       };
    }
     rpc CompressText(CompressTextRequest)  returns (CompressTextResponse) {
         option (google.api.http) = {
             get: "/compression/api/v1/text/compress"
        };
    }
     rpc DecompressText(DecompressTextRequest)   returns (DecompressTextResponse) {
         option (google.api.http) = {
             get: "/compression/api/v1/text/decompress"
        };
    }
}
```

```
enum ImageType {
        PNG  = 0;
        JPEG = 1;
}
message CompressImageRequest {
    bytes image = 1;
}
messageCompressImageResponse {
    bytes image = 1;
}
message DecompressImageRequest {
    bytes image = 1;
}
message DecompressImageResponse {
    bytes image = 1;
}
message CompressTextRequest {
    bytes text = 1;
}
message CompressTextResponse {
    bytes text = 1;
}
message DecompressTextRequest {
    bytes text = 1;
}
message DecompressTextResponse {
    bytes text = 1;
}
```

Lista figurilor

Lista tabelelor

Anexa B

Tabele

Dimensiune fişier text (octeţi)	Tip date	Dimensiune după compresie (Best)	Dimensiune după compresie (Default)	Dimensiune după compresie (Fast)	Dimensiune după compresie (Doar Huffman)	Dimensiune după compresie (No)
99	Repetat	48 48.48%	48 48.48%	68 68.69%	68 68.69%	115 116.16%
99	Unic	94 94.95%	94 94.95%	128 129.29%	94 94.95%	115 116.16%
2022	Repetat	183 9.05%	183 9.05%	187 9.25%	1112 55%	2038 100.79%
1769	Unic	710 40.14%	710 40.14%	756 42.74%	978 55.29%	1785 100.9%
10240	Repetat	230 2.25%	233 2.28%	295 2.88%	5368 52.42%	10256 100.16%
5277	Unic	1724 32.67%	1724 32.67%	1918 36.35%	2835 53.72%	5293 100.3%

Tabelul B.1: Rata de compresie fişiere text

	Imagine mică	Imagine medie	Imagine mare	Imagine gigantă
Dimensiune Lăţime x Lungime	100 x 120	512 x 496	1024 x 1500	4000 x 4000
Dimensiune (octeţi)	36000	761856	4608000	48000000
Dimensiune compresie Best (octeţi)	236	1460	6646	56321
Raport dimensiune compresie Best	0.66 %	0.19 %	0.14 %	0.12 %
Timp execuţie compresie Best (s)	0,025696	0,460886617	2,69279	27,35
Dimensiune compresie Default (octeţi)	332	1825	7775	56321
Raport dimensiune compresie Default	0.92 %	0.24 %	0.17 %	0.12 %
Timp execuţie compresie Default (s)	0,025696	0,426049450	2,51625	26,207
Dimensiune compresie Fast (octeţi)	333	2057	9210	71688
Raport dimensiune compresie Fast	0.92 %	0.27 %	0.2 %	0.15 %
Timp execuţie compresie Fast (s)	0,015916	0,270366040	1,78344	16,9363
Dimensiune compresie No (octeţi)	36205	762756	4611603	48025301
Raport dimensiune compresie No	100.57 %	100.12 %	100.08 %	100.05 %
Timp execuţie compresie No (secunde)	0,004395	0,047150913	0,30246	2,97536

Tabelul B.2: Rata de compresie PNG entropie mică

	Imagine mică	Imagine medie	Imagine mare	Imagine foarte mare
Dimensiune Lăţime x Lungime	100 x 120	512 x 496	1024 x 1500	4000 x 4000
Dimensiune (octeţi)	36000	761856	4608000	48000000
Dimensiune compresie Best (octeţi)	36215	762931	4612658	48036298
Raport dimensiune compresie Best	100.6 %	100.14 %	100.1 %	100.08 %
Timp execuţie compresie Best(s)	0,055334	1,02189	5,769	65,3939
Dimensiune compresie Default (octeţi)	36215	762931	4612658	48036298
Raport dimensiune compresie Default	100.6 %	100.14 %	100.1 %	100.08 %
Timp execuţie compresie Default (s)	0,050935	0,989136	5,7694	62,03826
Dimensiune compresie Fast (octeţi)	36205	762756	4611603	48025301
Raport dimensiune compresie Fast	100.57 %	100.12 %	100.08 %	100.05 %
Timp execuţie compresie Fast (s)	0,030182	0,51095	3,0343	31,83958
Dimensiune compresie No (octeţi)	36205	762756	4611603	48025301
Raport dimensiune compresie No	100.57 %	100.14 %	100.08 %	100.05 %
Timp execuţie compresie No (secunde)	0,004997	0,048537	0,31703	3,02121

Tabelul B.3: Rata de compresie PNG entropie mare

Anexa C

Glosar

Termen	Definiţie
HTTPS	*Hyper Text Transfer Protocol Secure* - este o versiune securizată a protocolului de transmisie HTTP.
gRPC	*Google Remote Procedure Call* - framework pentru comunicare între servicii, utilizând protocolul HTTP/2.
e2e	*End to end* - metpdă de testare a aplicaţiei, presupune testarea unui întreg use-case
DOM	*Domain Object Model*
RPC	*Remote Procedure Call* - protocol de comunicare între 2 entităţi aflate la distanţă, se caracterizează prin faptul ca metodele sunt apelate de parcă ar fi locale.
Protocol buffers	*Interface description Language* - folosit pentru translatarea unor structuri şi interfeţe în mai multe limbaje.