

MINISTERUL EDUCAȚIEI NAȚIONALE



UNIVERSITATEA TEHNICĂ

DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

BITSTORED
SISTEM DE STOCARE SECURIZATĂ A FIȘIERELOR

LUCRARE DE LICENȚĂ

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

2019



**FACULTATEA DE AUTOMATICĂ ȘI 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 FIȘIERELOR**

1. **Enunțul temei:** Crearea unui sistem de stocare a fișierelor în cloud, acesta fiind disponibil sub forma de mobile (iOS și Android) și de aplicație web. Aplicația realizează stocarea fișierelor în forma criptată, pentru a oferi protecție sporită a datelor, și compresată pentru utilizarea eficientă a spațiului de stocare al utilizatorului. De asemenea, sistemul oferă un mecanism de restabilire a datelor printr-un sistem de logare avansat.
2. **Conținutul lucrării:** Pagina de prezentare, Cuprins, Introducere, Obiectivele Proiectului, Studiu Bibliografic, Analiză și Fundamentare Teoretică, Proiectare de Detaliu și Implementare, Testare și Validare, Manual de Instalare și Utilizare, Concluzii, Bibliografie, Anexe.
3. **Locul documentării:** Universitatea Tehnică din Cluj-Napoca, Departamentul Calculatoare
4. **Consultanți:** Senior Lector Eng. Cosmina Ivan
5. **Data emiterii temei:** 1 ianuarie 2019
6. **Data predării:** 12 iulie 2019

Absolvent: _____

Coordonator științific: _____



UNIVERSITATEA TEHNICĂ

DIN CLUJ-NAPOCA

**FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL CALCULATOARE**

**Declarație pe proprie răspundere privind
autenticitatea lucrării de licență**

Subsemnatul(a)

_____, legiti-
mat(ă) cu _____ seria _____ nr. _____
CNP _____, autorul lucrării _____

elaborată în vederea susținerii examenului de finalizare a studiilor de licență la Facul-
tatea de Automatică și Calculatoare, Specializarea _____
din cadrul Universității Tehnice din Cluj-Napoca, sesiunea _____ a an-
ului universitar _____, declar pe proprie răspundere, că această lucrare este
rezultatul propriei activități intelectuale, pe baza cercetărilor mele și pe baza informațiilor
obținute din surse care au fost citate, în textul lucrării și în bibliografie.

Declar, că această lucrare nu conține porțiuni plagiate, iar sursele bibliografice au
fost folosite cu respectarea legislației române și a convențiilor internaționale privind drep-
turile de autor.

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

În cazul constatării ulterioare a unor declarații false, voi suporta sancțiunile admin-
istrative, respectiv, *anularea examenului de licență*.

Data

Nume, Prenume

Semnătura

Cuprins

Capitolul 1	Introducere - Contextul proiectului	1
Capitolul 2	Obiectivele Proiectului	4
2.1	Formularea temei	4
2.2	Obiectivele proiectului	6
2.3	Cerințe	6
2.3.1	Cerințe funcționale	6
2.3.2	Cerințe non-funcționale	8
Capitolul 3	Studiu Bibliografic	10
3.1	Caracteristicile arhitecturii monolitice	10
3.2	Caracteristicile arhitecturii orientate pe microservicii	12
3.3	Microservicii în Cloud	14
3.4	Securitatea în sistemele informatice	15
3.4.1	Amenințări	16
3.4.2	Criptografia	17
3.4.3	Compresia	19
3.4.4	Steganografia	19
3.5	Sisteme similare	20
3.5.1	Metodologia de analiză	21
3.5.2	CloudMe	22
3.5.3	Dropbox	23
3.5.4	CrashPlan	25
3.5.5	ICloud	26
3.5.6	Google Drive	28
3.5.7	OneDrive	29
3.5.8	pCloud	30
3.5.9	sync.com	31
3.5.10	Concluzii și plasarea sistemului	32

Capitolul 4	Analiză și Fundamentare Teoretică	34
4.1	Arhitectura conceptuală a sistemului	34
4.2	Cazuri de utilizare	35
4.2.1	Modele de cazuri de utilizare	37
4.3	Tehnologii	37
4.3.1	Golang	37
4.3.2	gRPC	38
4.3.3	Vue.JS	39
4.3.4	MongoDB	40
4.3.5	Couchbase	41
4.3.6	HTML, CSS, Bootstrap	41
4.3.7	JSON Web Token(JWT)	42
4.3.8	Docker și Kubernetes	42
4.3.9	Google Cloud	43
4.3.10	Git	44
Capitolul 5	Proiectare de Detaliu și Implementare	45
5.1	Arhitectura serverului	45
5.1.1	Descriere generală	45
5.1.2	Orchestrarea microserviciilor	46
5.1.3	Microserviciul de autentificare	47
5.1.4	Microserviciul de criptare	49
5.1.5	Microserviciul de steganografie și marcarea	50
5.1.6	Microserviciul de compresie	50
5.1.7	Microserviciul de fisiere	54
5.1.8	Microserviciul de utilizatori	54
5.2	Arhitectura aplicației web	55
5.2.1	Descriere generală	55
5.2.2	Descrierea componentelor	55
Capitolul 6	Testare și Validare	56
6.1	Testarea serverului	56
6.1.1	Reguli de testare în Golang	56
6.1.2	Testarea serviciilor	57
6.2	Testarea clientului	58
Capitolul 7	Manual de Instalare și Utilizare	59
7.1	Cerințe sistem	59
7.2	Instalare și configurare	59
7.3	Instrucțiuni de utilizare	62
7.3.1	Arborele de navigare	62
7.4	Navigare	63

Capitolul 8 Concluzii	65
8.1 Rezultate obținute	65
8.2 Contribuții	65
8.3 Dezvoltări ulterioare	66
Bibliografie	67
Anexa A Secțiuni relevante din cod	69
Lista figurilor	71
Lista tabelelor	73
Anexa B Diagrame UML	74
Anexa C Glosar	75

Capitolul 1

Introducere - Contextul proiectului

În epoca contemporană se observă o tendință continuă a digitalizării și transformării digitale, fapt care aduce un impact enorm atât asupra marilor companii, atât și asupra utilizatorilor individuali. Lumea bazată pe date va fi permanentă, mereu în urmărire, mereu în stadiu de monitorizare - pentru că va fi mereu în stadiu de învățare.

IDC[1] a definit trei locații principale în care digitalizarea are loc și unde este creat conținutul de date: tip nucleu (centre de date tradiționale și de tip cloud), tip muchie (infrastructuri de tip sucursală), și obiectivele finale (PC-uri, telefoane și dispozitive IoT). Sumarizarea tuturor acestor date, în momentul în care sunt create, capturate sau replicate, se numește Global Datasphere, și aceasta se confruntă cu o creștere spectaculoasă. IDC (International Data Corporation) estimează că volumul de date din Global Datasphere va crește de la 33 Zettabytes¹ în 2018 până la 175 Zettabytes în 2025, evoluția se poate observa în Figura 1.1.

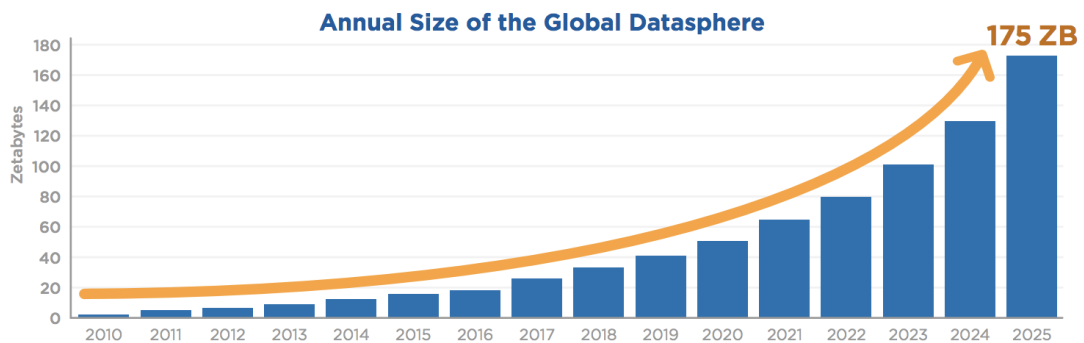


Figura 1.1: Creșterea volumului de date 2010-2025

În trecutul recent utilizatorii erau responsabili pentru datele lor, însă dependența și încrederea lor în serviciile cloud, în special din cauza conectivității, performanței și con-

¹1 Zetta byte echivalent cu 2^{70} bytes

fortului, continua sa crească ceea ce duce la noi provocări pentru furnizorii de servicii cloud. Mediul afacerilor urmărește centralizarea managementului datelor, pentru a putea oferi securitate, analiză de date, experiență utilizator mai bună (prin comunicare între dispozitive, IoT, personalizarea profilului). Responsabilitatea pentru managementul datelor utilizatorilor și businessurilor duce la o creștere continuă a centrelor de date ale furnizorilor de servicii Cloud. Ca rezultat importanța serviciilor cloud crește considerabil, iar utilizatorii nu doar permit acest lucru si se așteaptă la o creștere cât mai spectaculoasă.

Volumul de date cât mai mare stocat în cloud este scopul industriei de stocare a datelor. Pentru a supraviețui într-o lume care tinde a fi condusă de inteligența artificială și sistemel autonome sistemele de cloud au nevoie să se perfecționeze tot mai mult pentru a ține pasul cu evoluția lumii.

În calitate de clienți, oamenii doresc să obțină acces rapid și simplu la datele lor indiferent de oră și locația în care se află. Sistemele cloud sunt provocate să ofere servicii performante de acces, care nu vor expune datele utilizatorilor.

Organizațiile și utilizatorii au început să își schimbe destinația datelor de la infrastructuri fizice spre *cloud*-ul public, alții însă au început sa își dezvolte propriile soluții 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 CIA², acest sistem este prezentat în figura 1.2.

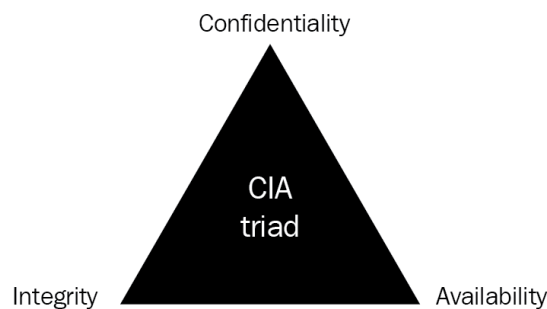


Figura 1.2: Triada CIA

Confidențialitatea se referă la protejarea datelor de la un access neautorizat. Integritatea se referă la protecția datelor de la modificări neautizate, orice modificare poate însemna o pierdere considerabilă pentru utilizator sau organizație. Disponibilitatea denotă faptul că informațiile vor fi accesibile doar pentru utilizatorii care au drept de acces, o încălcare a acestei reguli, din nou, se va provoca pierderea unor date. Toate cele 3 aspecte sunt esențiale pentru securitatea datelor, însă acestea sunt uneori complicat de oferit.

²Confidentiality, Integrity, and Availability

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

Creșterea volumului de date, pe de altă parte duce și la creșterea volumului fizic de *hardware* necesar. Compresia datelor este o operație care permite stocarea aceluiași volum de date la un preț mult mai redus[3]. Datorită compresiei un utilizator poate stoca un volum mai mare de date decât îi permite teoretic *hardware*-ul. Pe lângă faptul că se reduce spațiul de stocare a datelor, compresia contribuie și la micșorarea timpului necesar pentru transmisie sau pentru operațiile I/E[4].

Un alt beneficiu al compresiei este creșterea nivelului de securitate. Acest fapt se datorează alterării datelor, astfel chiar și un algoritm de criptare primitiv devine mult mai complicat de spart prin forță brută, în ecuație se mai adaugă și decompresia datelor, care este o operație costisitoare, nu se mai pot determina asocieri între simboluri, între secvențe de caractere și 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 securității datelor.

Creșterea încrederii în sistemele de stocare, creșterea volumului de date, inclusiv și a celor de sensibilitate înaltă, și a numărului de atacuri cibernetice crează o nouă provocare pentru sistemele de stocare cloud existente: creșterea securității datelor. De asemenea acest aspect crează și oportunități noi pentru sistemele la început de drum, le oferă o piață de defacere enormă și un număr ridicat de clienți care sunt gata să plătească preturi relativ ridicate pentru securitatea datelor sale.

Volumul de date crește, însă pe lângă conceptul celor *trei V*: volum, varietate și viteză, se mai adaugă și al 4-lea V: valoare. Pentru oferirea securității datelor, anual se cheltuie în jur de \$100 miliarde, în 2019 această sumă a ajuns la \$124 miliarde.

Atenția industriei IT ar trebui să se concentreze în jurul securității cibernetice, cu referire mai mult la valoarea datelor, nu la volumul sau sursa lor. Soluțiile sunt și trebuie să fie conduse de viziunea și gândirea oamenilor, dar validarea soluțiilor ar trebui să fie validate de inteligența datelor.

Aspectele enumerate sunt un avantaj competitiv pentru industria IT și sunt o provocare pentru construirea unei culturi sănătoase a datelor.

Capitolul 2

Obiectivele Proiectului

În acest capitol este prezentată tema proiectului, obiectivele și cerințele funcționale esențiale ale proiectului.

2.1 Formularea temei

Prin acest proiect, se urmărește implementarea unei platforme de *Cloud Storage*, aceasta trebuie să ofere o bună protecție a datelor, aceste date să nu fie accesibile dezvoltatorilor sistemului, atacatorilor sau oricărei alte instituții care colectează date. Sistemul este menit pentru stocarea datelor sensibile, sub răspunderea directă a utilizatorilor, de asemenea sistemul poate fi privit ca o platformă fără cunoștințe despre datele stocate, nimeni altul decât proprietarul fișierelor nu poate decripta datele.

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

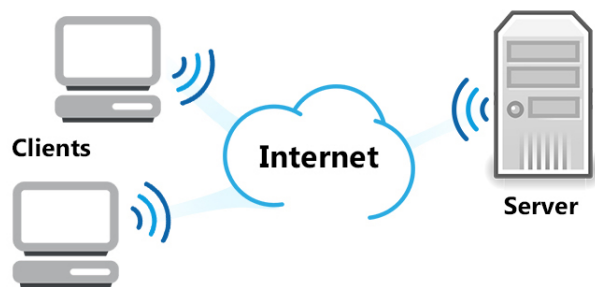


Figura 2.1: Arhitectura Client-Server

Serverul la rândul său având o **arhitectură bazată pe microservicii**, fiecare microserviciu fiind organizat sub formă de *layer*. La nivel de aplicație, responsabilitățile

sunt partiționate între server și client. La nivelul serverului, funcționalitățile esențiale sunt repartizate între servicii, astfel fiind respectat principiul *Single-responsability* din SOLID[5]. La nivel de microserviciu funcționalitățile sunt divizate utilizând **arhitectură layer**.

Serverul se ocupă de prelucrarea și stocarea datelor. Funcționalitățile cheie sunt: **compresie, criptare, conversie, steganografie, monitorizare și stocare**. Partea de stocare a datelor este împărțită în 2: fișiere și date personale. Datele utilizator sunt stocate în MongoDB¹, iar cele despre fișiere în Couchbase², este imposibil să asociezi un fișier cu identitatea unui utilizator fără accesul la ambele baze de date. Mai există o a 3-a bază de date, pentru care se utilizează Redis³, în care sunt stocate logurile din sistem.

Aplicația 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. Aplicația va trebui să fie rezistentă la un nivel mare de utilizare și să se autoscaleze prin crearea unor replici ale aceluiași cod, doar pentru microserviciile utilizate intens.

Sistemul va oferi utilizatorilor funcționalitățile de bază a unui sistem de stocare în *cloud*: încărcare fișier, descărcare fișier, creare fișier nou, grupare fișiere și management fișiere. Pe lângă funcționalitățile enumerate, sistemul va oferi un nivel de securitate înalt prin criptarea tuturor datelor și eficiență de utilizare a spațiului prin compresia datelor. Cheia de criptare/decriptare nu va fi stocată nicăieri în server sau client, utilizatorul va avea în totalitate responsabilitatea de a își păstra fișierele sigure și de a putea recupera datele stocate în sistem, din păcate este imposibil să i se ofere asisitență în cazul pierderii parolei, regenerarea ei prin forță brută ar putea dura între câteva ore și câțiva ani.

De asemenea la descărcarea unui fișier, utilizatorul va avea posibilitatea de a ascunde mesaje în imagini, sau de a aplica marcaje vizibile pe imagini. Această funcționalitate poate fi folosită atunci când se partajează un fișier, pentru a determina dacă datele au fost expuse din sistem și identitatea celui care le-a expus. Un exemplu de steganografie pe imagini este prezentat în figura 2.2.

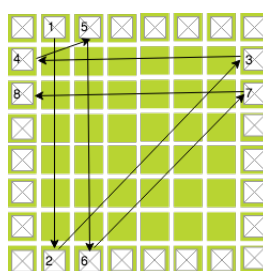


Figura 2.2: Exemplu de steganografie pe imagini

¹<https://docs.mongodb.com/>

²<https://www.couchbase.com/>

³<https://redis.io/>

După cum se poate observa, manipularea biților nu este vizibilă pentru ochiul uman, deoarece alterarea celui mai puțin semnificativ bit nu cauzează schimbări semnificative în percepție, însă pentru calculator poate conține informații valoroase.

Pentru client, se urmărește implementarea unei aplicații web. Aceasta va trebui să interacționeze cu utilizatorii prin interfața web și cu serverul prin apeluri de tip gRPC⁴, pentru a asigura o viteză mai bună de transmisie și răspuns. Clientului nu îi vor fi cunoscute decât 3 dintre serviciile existente: serviciul de autentificare, serviciul de management utilizatori și serviciul de management fișiere.

2.2 Obiectivele proiectului

- Sistemul va oferi utilizatorilor experiența deplină a unui sistem de stocare în *Cloud*: încărcare fișier, descărcare fișier, modificare fișier, grupare fișiere.
- Sistemul va oferi utilizatorilor un nivel înalt de securitate a datelor personale prin folosirea unor algoritmi de criptare eficienți. Cheile nu vor fi stocate în sistem, iar datele vor fi aproape imposibil de decriptat de către atacatori.
- Conexiunea între client și 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*.
- Folosirea spațiului de stocare va fi eficientizată prin folosirea unor algoritmi de compresie eficienți, care permit reducerea volumului de date de mai mult de 2 ori în cazul unor date uzuale.

2.3 Cerințe

În această secțiune sunt descrise și enumerate cerințele principale ale sistemului. Acestea se referă atât la funcționalitățile propriu-zise, cât și la experiența utilizatorului.

2.3.1 Cerințe funcționale

În ingineria software, cerințele funcționale definesc funcțiile sistemului sau comportamentul său, unde funcțiile sunt descrise ca o specificație sau set de acțiuni dintre intrare și ieșire.

Aplicația va oferi funcționalități diferite, în funcție de rolul utilizatorului. Se poate face distincția între 2 tipuri de utilizatori: utilizator primar și administrator. Diagrama rolurilor sistemului este prezentată în Figura 2.3.

⁴<https://grpc.io/>

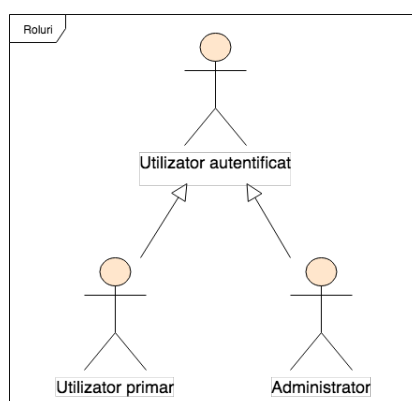


Figura 2.3: Tipurile de utilizatori ale sistemului

Pornind de la obiectivele proiectului, pot fi determinate următoarele cerințe funcționale:

- 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 și parola, care au fost indicate la momentul înregistrării.
- CF3 Ca Utilizator, doresc ca sesiunea mea să fie păstrată și după închiderea *browser*-ului.
- CF4 Ca utilizator, doresc să îmi opresc sesiunea curentă prin deautentificare.
- CF5 Ca utilizator, doresc să am posibilitatea de a îmi bloca temporar contul, fără a pierde accesul la date.
- CF6 Ca utilizator, doresc să am posibilitatea de a îmi debloca contul.
- CF7 Ca utilizator, doresc să am posibilitatea de a îmi șterge contul și 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 fișierele mele din *Drive* și să pot naviga prin ierarhia de directoare.
- CF12 Ca utilizator, doresc să pot crea un fișier nou, care să aibă sau nu conținut.
- CF13 Ca utilizator, doresc să pot crea un director nou.

- CF14 Ca utilizator, doresc să pot încarca un fișier în sistem.
- CF15 Ca utilizator, doresc să pot modifica un fișier care se află în spațiul meu de stocare.
- CF16 Ca utilizator, doresc să pot șterge un fișier din sistemul meu de fișiere.
- CF17 Ca utilizator, doresc să pot schimba directorul în care se află un fișier.
- CF18 Ca utilizator, doresc să pot crea o copie a unui fișier într-un alt director, fără a îmi crește spațiul de stocare utilizat.
- CF19 Ca utilizator, doresc să pot descărca un fișier care se află în spațiul meu de stocare.
- CF20 Ca utilizator, doresc să pot codifica mesaje în fișierul descărcat.
- CF21 Ca utilizator, doresc să pot vedea dacă în fișierul pe care l-am încărcat sunt codificate mesaje.
- CF22 Ca utilizator, doresc să mi se ofere date privind volumul de stocare economisit datorită funcționalităților 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.

2.3.2 Cerințe non-funcționale

Cerințele non-funcționale se referă mai mult la calitatea și experiența de utilizarea a sistemului, decât la funcționalitățile și capacitățile specifice. Aceste cerințe descriu cum ar trebui să fie sistemul, nu ce ar trebui să facă acesta.

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

- (1) Codul trebuie să respecte principiile SOLID.
- (2) Dependențele 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 soluții copiate de pe fudumuri de programare, acestea ar putea conține vulnerabilități.
- (5) Codul trebuie să fie bine documentat și ușor de citit și 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 și vor fi decriptate doar la cererea proprietarului legitim.
- (2) Fișierele vor fi criptate utilizând algoritmi complecși, iar datele despre cheile de criptare nu vor fi stocate în sistem.
- (3) Nu se va face o asociere directă între identitatea utilizatorului și datele stocate în sistemul de fișiere.

CNF3 Sistemul trebuie să utilizeze eficient spațiul de stocare.

- (1) Datele vor fi compimate 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 interfață utilizator inteligibilă, ușor de utilizat și care nu creează ambiguitate în modul de utilizare.

CNF5 Sistemul va fi rezistent la căderile unor anumite servicii și își va putea reveni ulterior, fără ca utilizatorul să știe acest lucru.

CNF6 Clientul trebuie să fie suportat de mai multe browsere și să ofere aceeași interfață web.

Capitolul 3

Studiu Bibliografic

3.1 Caracteristicile arhitecturii monolitice

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

- Autorizarea - responsabilă pentru autorizarea utilizatorului.
- Prezentarea - responsabilă pentru tratarea apelurilor HTTP și 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 aplicațiilor (sau un anti-pattern, dacă privim din perspectiva dezavantajelor). Stilurile și pattern-urile sunt de obicei grupate în categorii sau seturi, pentru a fi mai ușor de asociat. Categoriile de bază pentru arhitectura monolitică sunt:

- Modul - unitățile de cod sunt separate în module și sunt compilate împreună producând un singur artefact.
- Alocare - Toate componentele sistemului sunt compilate, livrate și configurate în același timp, ca un singur artefact, toate având aceeași versiune, indiferent de câte ori au fost modificate. Numărul versiunii este egal cu numărul de livrări al artefactului
- Runtime - Există o singură instanță aplicației care execută toate sarcinile.

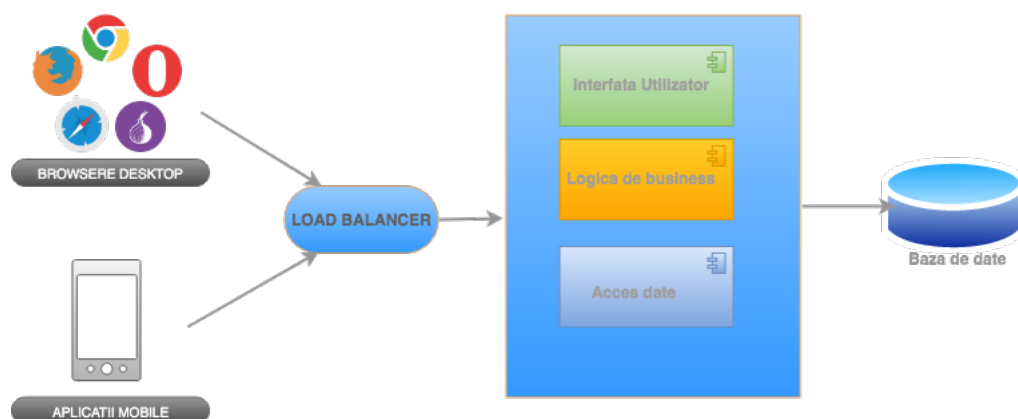


Figura 3.1: Arhitectura monolitică

Avantajele arhitecturii monolitice sunt:

- Ușor de dezvoltat - la începutul unui proiect este mult mai ușor să dezvolti o arhitectură monolitică.
- Ușor de testat. De exemplu, se pot implementa teste end-to-end prin simpla rulare a aplicației și testarea ei cu un tool specializat.
- Ușor de pus în funcțiune, este necesară doar copierea pe un server și rularea programului.
- Scalabilă pe orizontală prin rularea mai multor instanțe.

Arhitectura monolitică este abordarea tradițională care este folosită în multe sisteme, care sunt construite ca o aplicație autonomă. Chiar dacă este prezentă în multe aplicații existente și încă este folosită pentru dezvoltarea aplicațiilor noi, limitările și problemele existente în acest mod de abordare duc la creșterea popularității arhitecturii bazate pe microservicii. **Dezavantajele** arhitecturii monolitice sunt:

- Menținanța - dacă o aplicație este prea mare este foarte greu să faci schimbări rapide și să nu afectezi funcționarea corectă a altor componente.
- Dimensiunea aplicației duce creșterea timpul de start-up.
- Toată aplicația va trebui restartată atunci când se face o schimbare de cod.
- Este foarte complicat de scalat.
- O problemă în una dintre componentele poate afecta întreaga aplicație.
- Este complicat să adopte tehnologii și framework-uri noi, deoarece prea multe lucruri trebuie schimbate în același timp.

- Spre finalul ciclului de dezvoltare complexitatea de a scrie cod devine mai mare, iar raportul timp-eficiență devine tot mai mare.

Această arhitectură are și plusuri și minusuri, dar, dat fiind faptul că de fiecare dată când se rescrie o porțiune 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 niște entități mici, independente, autonome, create să funcționeze împreună, fiecare dintre ele este focusat pe un singur lucru și are scopul de a-l face bine. Arhitectura bazată pe microservicii este un stil architectural care structurează aplicația ca o colecție de servicii independente și modulare, care sunt ușor de testat, de întreținut și de înțeles. Acest tip de abordare duce la creșterea agilității prin îmunătățirea productivității și scăderea timpului de dezvoltare a produsului. Microserviciile au demonstrat că sunt un sistem de nivel superior, în special pentru aplicații mari care sunt dezvoltate de mai multe echipe. Pe lângă beneficiile enumerate mai sus, microserviciile mai oferă următoarele avantaje:

- Sunt mentenabile.
- Sunt scalabile.
- Sunt ușor de testat.
- Au nivel de cuplare joasă.
- Sunt independente.
- Sunt rezistente la căderi.
- Sunt organizate în funcție de capacitățile și funcționalitățile de business.
- Oferă independență dezvoltatorilor.
- Pot fi dezvoltate independent.
- Modificările pot fi aplicate ușor, deoarece nu mai necesită recompilarea întregului produs.

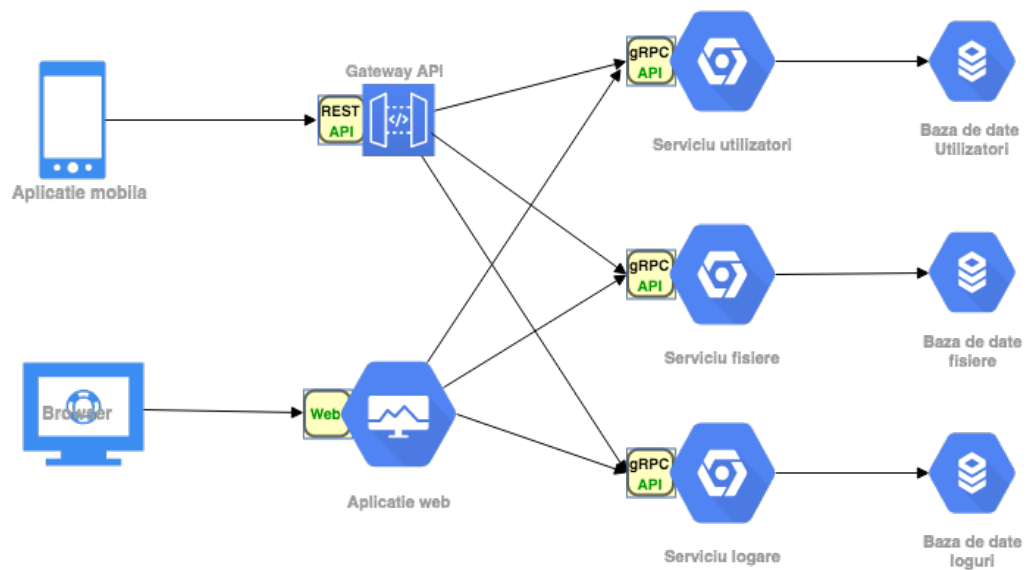


Figura 3.2: Arhitectura bazată pe microservicii

Arhitectura bazată pe microservicii permite integrarea și livrarea continuă a unui produs complex și de volum mare, deasemenea 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 și Amazon sunt cunoscute pentru arhitecturile lor diverse, care au evoluat de la *Monolit* la *Microservicii* cu scopul de a putea face față unor volume imense de date.

Totuși, ca oricare altă soluție, 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 interacțiunii este destul de complexă.
- IDE-urile și tool-urile existente au un număr scăzut de funcționalități care ajută la dezvoltarea aplicațiilor distribuite.
- Un sistem format din microservicii are un nivel mai ridicat al consumului de resurse.

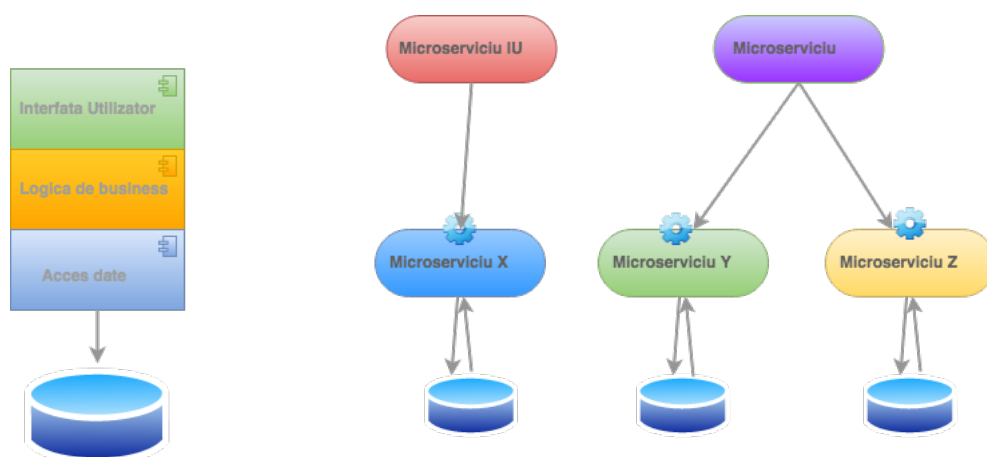


Figura 3.3: Diferența dintre arhitecturi

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

3.3 Microservicii în Cloud

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

Faptul că plătești doar ceea ce utilizezi reduce considerabil prețul experimentării, dar și crește posibilitățile de experimentare. Noi funcționalități pot fi integrate, pot fi oprite și restartate cu noi parametri în cazul unui eșec. Datorită cloudului se pot efectua numeroase experimente fără riscuri, acets lucru constituind cheia de succes a inovației. Acest fapt se potrivește ideal cu conceptul de *microservicii*, oferind posibilitatea de a atinge un nivel înalt de agilitate.

Programabilitatea cloud-ului permite automatizarea proceselor de dezvoltare și livrare. Integrarea contnuă este o parte a ciclului de viață a serviciilor în cloud. Livrarea continuă, pe de altă parte, introduce provocii noi a complexității de administrarea a multiplelor servicii în paralel.

Gândirea, perfecționarea continuă, livrarea continuă, managementul, monitorizarea și întreținerea API-urilor este o responsabilitate complexă și consumă extrem de mult timp. Sistemele Cloud oferă suport pentru acestea și ușurează viața dezvoltatorilor.

Arhitectura bazată pe microservicii este o abordare distribuită, dezvoltată pentru a rezolva limitările arhitecturii monolitice clasice. Microserviciile facilitează scalarea aplicațiilor sau a anumitor module ale aplicației. Totuși sunt o provocare din punctul de vedere a complexității arhitecturale și a operării sistemului. Serviciile cloyd contribuie la reducerea acestei complexități prin oferirea unor funcționalități preimplementate de management al serviciilor.

3.4 Securitatea in sistemele informatice

Securitatea aplicațiilor cuprinde măsurile luate pentru a asigura și a îmbunătăți securitatea sistemului. Deseori aceasta este asigurată prin găsirea, înlăturarea și prevenirea vulnerabilităților de securitate. Sunt utilizate mai multe tehnici pentru a obține acest lucru, acestea pot fi aplicate la diferite etape ale ciclului de dezvoltare, cum ar fi: *design*, *dezvoltare*, *livrare*, *perfecționare* sau *mentenanță*.

Sunt utilizate diferite abordări pentru a găsi diferite subseturi ale vulnerabilităților de securitate, acestea au un impact diferit prin cost, timp, efort și procentul de vulnerabilități ce pot fi detectate. Tehnicile esențiale sunt:

- *Whitebox* - se referă la analiza securității sau a codului. Această abordare poate fi adoptată doar de un inginer care înțelege deplin codul, și poate observa problemele prin analiza manuală și vizuală a codului sursă.
- *Blackbox* - reprezintă auditul în securitate. Operația 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 vulnerabilitățile arhitecturale și ale dependențelor 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 vulnerabilități coordonate* - sunt aplicații care oferă recompense hackerilor experimentați pentru găsirea de probleme de securitate.

Utilizarea acestor tehnici în mod adecvat îmbunătățește calitatea sistemului prin înlăturarea vulnerabilităților. Acest proces este în totalitate responsabilitatea echipei de dezvoltare.

În timp ce stocarea în sistemele cloud este convenabilă și oferă posibilitatea de a accesa datele indiferent de locație și oră, de pe aproximativ orice dispozitiv cu conexiune la internet, securitatea sistemelor de stocare este o problemă prioritară a organizațiilor IT și a departamentelor de securitate. Beneficiul principal al adoptării stocării în cloud este asigurarea securității și integrității datelor cu caracter senzitiv.

Dezvoltatorilor sistemelor de cloud le aparține în totalitate responsabilitatea pentru securitatea aplicațiilor lor. Aceștia implementează în sistemele lor toate funcționalitățile esențiale pentru securitate, acestea fiind: *autentificarea, autorizarea, controlul accesului și criptarea*. De aici încolo, fiecare companie are responsabilitatea de a adăuga noi nivele de protecție pentru date și de a restricționa cât mai mult accesul la datele sensibile.

3.4.1 Amenințări

Administratorii de sistem și dezvoltatorii de servicii software sunt mereu la straja securității aplicațiilor. Însă sunt numeroase probleme care par netriviale, însă pot compromite datele aplicației. Principalele amenințări pentru securitatea aplicațiilor sunt:

- *Utilizatorii* - utilizatorii aplicațiilor noastre sunt cea mai mare amenințare pentru propria lor integritate și pentru datele lor. Deseori aceștia nu realizează cât de important e să accesezi 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 eșua în acest caz.
- *Greșeli elementare de structurare sau scriere a codului* - în ciuda multiplelor avertismente și a anilor de educație încă există cod cu greșeli elementare de securitate. Problemele triviale sunt: *SQL-injection* și *Cross-site scripting*. O recomandare în această direcție este utilizarea unor librării specializate pentru SQL și pentru randare a datelor provenite de la utilizatori, acestea tratează aceste cazuri și aplicația nu poate fi atacată în acest mod.
- *Utilizarea unor librării învechite* - este recomandat să se utilizeze cele mai noi versiuni ale unor librării și aplicații, aceste nu conțin de obicei erorile și problemele de securitate cunoscute.

- *Setarea unor permisiuni de acces greșite* - prin setarea permisiunilor de acces greșite, utilizatorul poate obține prea multă libertate și control asupra aplicației. Este recomandat ca permisiunile să fie la nivelul minim necesar pentru utilizarea aplicației conform modelului de cazuri de utilizare.
- *Hackerii* - persoanele care doresc să obțină profit sau date prețioase vor încerca mereu să strice aplicația, problema nu poate fi înlăturat omplet, 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 crează o facilitate pentru hackerii, aceștia nu mai au nevoie de muncă suplimentară pentru obținerea datelor valoroase odată ce au obținut control asupra sistemului.

Măsurile esențiale pentru securitate vor fi discutate în secțiunile ce urmează.

3.4.2 Criptografia

Criptografia - este utilizată pentru ascunderea mesajelor. Există numeroase metode de criptare a datelor începând cu Cifrul lui Caesar, una dintre cele mai primitive metode de criptare existente, terminând cu AES și RSA, care sunt metodele de criptare standardizate, considerate aproape invincibile în momentul de față. Totuși criptografia nu este o soluție generală pentru securitate, aceasta este privită mai mult ca un *tool*. Adversarul principal al criptografiei este - criptanaliza, știința destinată descifrării mesajelor criptate prin analiza datelor de intrare și ieșire ale unui algoritm.

Pentru criptarea și decriptarea fișierelor, a fost selectat algoritmul *twofish*[10]. Acesta este un cifru cu cheie simetrică, blocurile de date au o lungime de până în 128 biți, iar cheile utilizate pot avea lungimi de până la 256 biți. Algoritmul a participat în concursul AES, a fost printre ultimii finaliști, dar nu a fost selectat pentru standardizare. Eficiența și siguranța acestui algoritm este apropiată de cea a actualului AES, acesta fiind mai încet pentru chei de 128 de biți, dar mai rapid pentru chei de 256 de biți. Algoritmul se bazează pe *rețele Feistel*, operația fundamentală a algoritmului este maparea datelor de intrare în funcție de cheie în date de ieșire, funcția fiind non lineară și non-surjectivă. Un alt element cheie al algoritmului este *S-boxes* - acestea sunt substituții non lineare folosite în majoritatea cifrurilor pe blocuri. Alte operații folosite sunt mapările cu ajutorul matricilor MDS, transformările Pseudo-Hadamard și operația fundamentală XOR. În Figura 3.4 este reprezentată o rundă din algoritm.

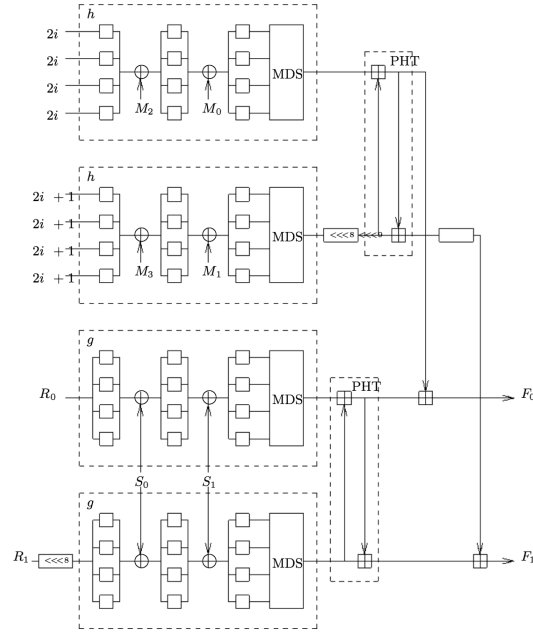


Figura 3.4: Diferența dintre arhitecturi

Avantajele principale ale algoritmului sunt: *performanța*, *flexibilitate*, *simplicitate* și *eficiență*.

Algoritmul selectat pentru criptarea parolilor este **Argon2**. Acest algoritm a fost ales câștigătorul competiției *Password Hashing Competition*¹ în Iulie 2015. Există 3 versiuni ale acestui algoritmului:

- *Argon2d* maximizează rezistența la atacurile care folosesc GPU. Acesta accesează memoria într-o ordine care depinde de caracterele parolei, și reduce riscul atacurilor cauzate de dependențele memorie-timp
- *Argon2i* reduce riscul atacurilor bazate pe canale auxiliare.
- *Argon2id* versiune hibridă a celor 2 menționate anterior, are avantajele ambelor versiuni și este recomandat să fie utilizat.

Toate 3 versiuni permit configurarea a 3 parametri de control: timp execuție, memorie cerută și gradul paralelismului.

Nu există date oficiale sau publice cu referire la atacuri soldate cu succes asupra Argon2d sau Argon2i. Spre deosebire de algoritmii similari, cum ar fi bcrypt sau PBKDF2, argon2 este foarte eficient, foarte sigur și simplu de configurat.

¹<https://password-hashing.net>

3.4.3 Compresia

În procesarea de date, compresia[11] este un proces care presupune encodarea datelor utilizând mai puțini biți decât în setul de date inițial. Compresia poate fi de 2 tipuri: *cu pierderi* sau *fără pierderi*. Compresia fără pierderi reduce numărul de biți prin identificarea și eliminarea redundanței statistice. Compresia cu pierderi se bazează pe reducerea dimensiunii datelor prin eliminarea informației redundante sau ne semnificative, la efectuarea procesului invers datele nu vor fi egale cu cele inițiale.

Avantajele compresiei sunt:

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

Unul dintre cei mai eficienți algoritmi de compresie este *zlib*². Acesta este un algoritm open-source, nepatentat, disponibil pe toate platformele și pentru diverse tipuri de date, de la text la video.

Compresia a forst și rămâne a fi centrul unei schimbări masive în ingineria calculatoarelor. Motivul fiind o simplă teorie economică: fișierele compresate sunt fișiere mai mici. Datorită acestui fapt: sunt mai ușor de transmis, mai ușor de prelucrat, mai ușor de stocat și, desigur, mai ieftine. Distribuitorii plătesc mai puțin să le distribuie, iar clienții plătesc mai puțin să le consume. În lumea în care calculele sunt echivalente cu banii, compresia reprezintă cea mai viabilă soluție economică.

3.4.4 Steganografia

Steganografia[12] este o tehnică de protecție a datelor, aceasta oferă soluții 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 ne semnificativ bit este o metodă simplă și eficientă de encodare a informațiilor ascunse. Vizual nu se observă diferența dintre imaginea inițială și imaginea care conține date encodate în cel ai ne semnificativ bit. La o analiză statistică se pot decoda datele și algoritmul nu mai este fezabil. Pentru a obține o securitate mai ridicată, a se utilizează o metodă în care nu se ascund datele în întreaga imagine, ci doar în pixelii de pe margine. Pentru complicarea analizei statistice se adaugă encodarea datelor în zig-zag.

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

²<https://zlib.net>

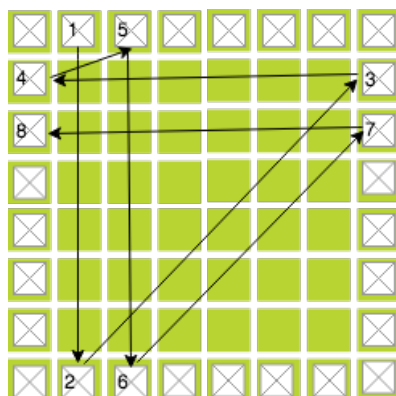


Figura 3.5: Exemplu de encodare a unui byte

După cum se poate observa din Figura 2.2, pentru encodarea unui octet se execută 8 pași, biții sunt encodați alternativ în marginile imaginii. Această metodă nu poate fi considerată 100% sigură, însă reduce riscul descoperirii datelor în cazul unei analize statistice clasice, datele se pot afla doar știind ordinea exactă de codificare.

3.5 Sisteme similare

Acest capitol reprezintă clasificarea și analiza sistemelor similare existente, bazată pe etapa de cercetare a proiectului. Sistemele au scop și funcționalități similare cu proiectul propus. Sistemele alese pentru comparație sunt:

- CloudMe
- Dropbox
- CrashPlan
- iCloud
- Google Drive
- OneDrive
- pCloud
- sync.com

Dropbox, Google Drive, iCloud și OneDrive au fost incluse în acest studiu deoarece sunt în top 10 cele mai populare servicii de cloud 2019 [13]. Acestea reprezintă un model pentru cum este văzut un cloud storage modern: simplu de configurat, simplu de utilizat

și disponibil la un preț avantajos. pCloud și sync.com sunt în topul sistemelor cu cea mai înaltă recuritate de pe piață. pCloud este categorizat ca un sistem înfraudabil și nu a avut nici o expunere a datelor utilizatorilor. Însă aceste sisteme vin și cu prețuri de 3-4 ori mai mari decât sistemele clasice.

3.5.1 Metodologia de analiză

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

Sunt sumarizate prețurile de utilizare a sistemului și a diferitor opțiuni. Sunt analizate detaliile capabilităților tehnice și organizaționale ale părții client și server a sistemului. Informațiile colectate sunt bazate pe secțiunile *Terms of Service* și *Privacy Policy* ale documentațiilor oficiale ale sistemelor[14]. Rezultatele analizei comparative au ca scop determinarea cerințelor principale ale unui sistem de stocare cloud și comparația sistemului elaborat cu cele existente. În secțiunile ce urmează se vor analiza următoarele funcționalități:

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 cerințele obligatorii pentru funcționalitatea respectivă au fost îndeplinite și câteva dintre cele opționale.
- ✓ este echivalent pentru *bine*, adică toate cerințele obligatorii pentru funcționalitatea respectivă au fost îndeplinite.
- ± este simbolul pentru *bine cu câteva vulnerabilități*, nu toate cerințele esențiale au fost îndeplinite.
- ✗ este echivalent cu *slab*, cel puțin o cerință obligatorie nu a fost îndeplinită.
- ✗✗ este echivalentul pentru *foarte slab*, adică mai multe dintre cerințele obligatorii nu sunt îndeplinite în funcționalitatea respectivă sau funcționalitatea lipsește.

3.5.2 CloudMe

CloudMe[15] este un sistem de cloud standard, care vine cu un serviciu de sincronizare și backup a fișierelor, după o analiză detaliată ne putem da seama că acest sistem a fost inspirat din arhicunoscutul **Dropbox** care este analizat în secțiunea 3.5.3.

În tabelul 3.2 sunt prezentate funcționalitățile sistemului și o notă a fiecărei funcționalități în concordanță cu evaluările primite pe pagina oficială a sistemului, precum și a vulnerabilităților depistate în ultima perioadă.

Tabelul 3.2: CloudMe Funcționalități

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

Tabelul 3.3 prezintă disponibilitatea **CloudMe** pe diferite platforme și prețul acestuia.

Tabelul 3.3: CloudMe Platforme disponibile și preț

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

Un fapt bun despre **CloudMe** este că acesta pune la dispoziția utilizatorului un spațiu de stocare gratuit de 3GB, iar pentru volume de date mai mari oferă opțiuni la prețul mediu al pieței. Serviciul oferă funcționalitățile de bază, însă nimic deosebit în materie de securitate.

Interfața web a **CloudMe** este foarte simplă și clară, este evident cum să încarci un fișier sau cum să îl schimbi în alt folder.

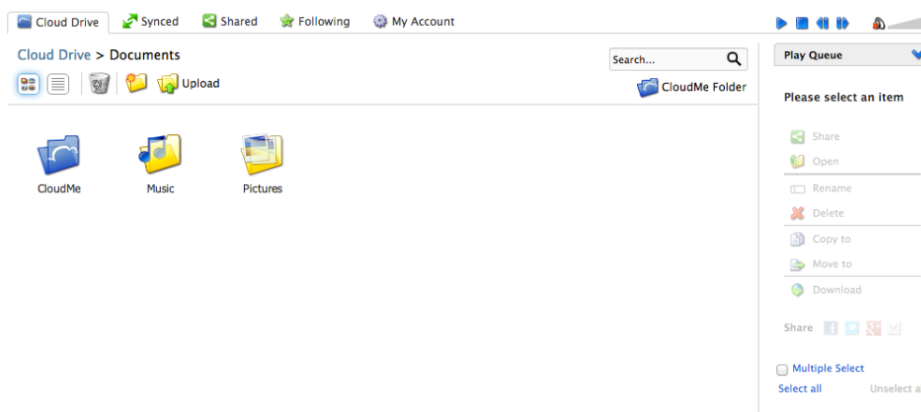


Figura 3.6: CloudMe - interfața web

CloudMe oferă sincronizarea aplicațiilor pentru Windows, Mac, Linux, iOS și Android. Pentru ca sincronizarea să poată avea loc utilizatorul primește un așa numit ”director albastru” oferă utilizatorului sincronizare în timp real. De asemenea, **CloudMe** oferă opțiunea de a alege timpul la care se va întâmpla sincronizarea. Pentru a distribui fișiere sunt disponibile mai multe opțiuni, începând cu distribuire clasică, care permite utilizatorului să ofere acces la fișierele lui prin distribuirea unui link, și ajungând la metode mult mai colaborative care permit altor utilizatori, cărora li se oferă acces, să modifice fișierele din cloud-ul altui utilizator sau să încarce fișiere noi.



Figura 3.7: CloudMe - planuri de preț

CloudMe arhivează versiunile precedente ale fișierelor prin funcționalitatea coșului de gunoi, care păstrează pentru 60 de zile toate fișierele care au fost șterse.

În privința securității **CloudMe** nu este o opțiune prea bună deoarece nu oferă criptarea datelor, acestea fiind vulnerabile pentru atacuri. Este posibil să încarci și să descarci fișiere criptate, însă criptarea și decriptarea acestora rămâne la latitudinea utilizatorului.

CloudMe este o aplicație foarte ușor de utilizat și are o interfață foarte intuitivă. Acesta dispune de funcționalitățile de bază ale unui sistem de stocare în cloud, însă nu este potrivit pentru stocarea fișierelor cu conținut de date senzitiv din cauza lipsei criptării, de asemenea atunci când fișierele sunt distribuite după un link este foarte greu să determini cine a făcut public un fișier cu caracter privat deoarece link-ul de acces poate fi ușor furat.

Un alt defect al acestui sistem este funcționalitatea de *Sync*, motivul pentru care i-am oferit punctaj minim este că în ultimul an a avut multiple vulnerabilități, conform *CVE-2018-6892*³ atacatorii se puteau conecta la clientul de ”CloudMe Sync” prin portul 8888 și trimiterea unor date malițioase puteau cauza ”buffer overflow”, acest lucru le oferea control asupra execuției și posibilitatea de a executa cod malițios.

3.5.3 Dropbox

Dropbox a fost lansat în 2007 și este definit ca unul dintre cele mai bune servicii de cloud pentru uz general[16]. Sistemul are peste 500 de milioane de utilizatori în toată lumea, fiind unul dintre cele mai competitive servicii de pe piață.

³<https://nvd.nist.gov/vuln/detail/CVE-2018-6892>

În tabelul 3.4 este prezentată o evaluare a sistemului pe baza unei evaluări personale, dar și pe baza evaluărilor oferite de *CloudWards*[16] și *On the Security of Cloud Storage Services*[14].

Tabelul 3.4: Dropbox Funcționalități

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

În tabelul 3.5 sunt prezentate opțiunile de clienți disponibili pentru **Dropbox** și oferta de preț.

Tabelul 3.5: Dropbox Platforme disponibile și preț

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

Dropbox este ușor de utilizat atât utilizând aplicația web, cât și cea mobile sau desktop. În figura 3.8 este prezentată interfața web a sistemului.

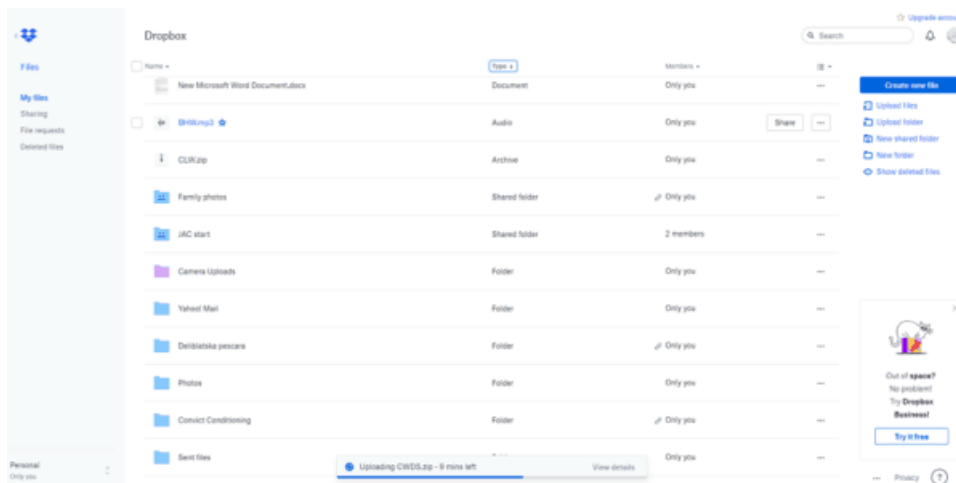


Figura 3.8: Dropbox - interfața web

Dropbox oferă un serviciu de sincronizare "rapid și inteligent", cu diverse posibilități de customizare, însă această opțiune poate fi aplicată doar pe anumite directoare, lucru care restrânge libertatea utilizatorului.

Funcționalitatea de partajare a fișierelor este în topul celor disponibile pe piață deoarece oferă partajare atât de fișiere cât și de directoare, cu parolă sau fără, cu customizare de permisiune și termen de expirare. Se pot trimite link-uri pe mail sau prin copierea directă, link-urile pot fi șterse și fișierul nu mai este accesibil prin acel link. Această funcționalitate este disponibilă atât din orice aplicație **Dropbox**.

O funcționalitate a cărui autor este **Dropbox** este deduplicarea la nivel de bloc, prin împărțirea fișierului la încărcare în multiple porțiuni de dimensiune fixă și prin scanarea dacă porțiunea există, acest lucru oferă o viteză de încărcare mai mare, dar prezintă un risc deoarece un atacator poate determina conținutul unui fișier de anumit format prin încărcări repetate de conținut diferit a anumitor porțiuni, de exemplu un fișier cu analize medicale.

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

Dropbox a avut o evoluție specaculoasă în ultimii 6 ani, adaugând numeroase noi funcționalități și devenind mult mai ușor de utilizat. Însă **Dropbox** rămâne o soluție pentru utilizatorii care nu au date sensitive stocate în acest sistem. Există mai multe variante de compromitere a datelor, una dintre acestea este cauzată de implementarea FTS, prin pastrarea metadatelor pentru căutare în format necriptat. Alta vulnerabilitate este cauzată de deduplicarea la nivel de bloc ce se execută la nivel de întreg sistem în loc de nivel fișiere utilizator.

O vulnerabilitate de securitate a fost descoperită în 2018 *CVE-2018-12271*⁴ atunci când un atacator se putea loga pe un cont **Dropbox** cu orice amprentă arbitrară și avea access la toate fișierele 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 fișierelor și backup care a apărut pe piață în 2007. În prezent, sistemul a devenit unul foarte popular, zilnic acesta procesează peste 100 de miliarde de fișiere.

În tabelul 3.6 sunt prezentate funcționalitățile sistemului și o evaluare efectivă a acestor capabilități.

⁴<https://www.cvedetails.com/cve/CVE-2018-12271/>

Tabelul 3.6: CrashPlan Funcționalități

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

Din evaluarea de mai sus se observă că sistemul nu oferă partajare de fișiere, în realitate această funcționalitate a fost înlăturată recent, cauza fiind riscul de compromitere a datelor. De asemenea partajarea fișierelor și menținerea 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 prețul este mai mic comparabil cu adversarii analizați în secțiunile precedente, acest fapt se poate datora lipsei unor anumite funcționalități.

Tabelul 3.7: CrashPlan Platforme disponibile și prețuri

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[17] datorită factorului că nu are dimensiune maximă a fișierelor pentru backup. Crashplan nu necesită implicare din partea utilizatorului pentru a executa copierea regulată a fișierelor. De asemenea, CrashPlan permite executarea operațiunii de backup în același cont de utilizator a până la 10 calculatoare.

CrashPlan oferă mai multe nivele de securitate pentru fișiere, toate datele sunt criptate de la client până la server. Criptarea se execută utilizând o cheie de 448 biți pentru utilizatorii unui plan plătit, pentru utilizatorii opțiunii gratuite se utilizează o cheie de 128 biți. Cheile de criptare sunt generate utilizând un sistem eficient de numere aliate. De asemenea există opțiunea de a selecta o cheie privată de criptare, acesta nu va fi salvată niciodată sub formă de text și nu va fi accesibilă nimănui.

Cel mai important lucru care face **CrashPlan** un sistem extrem de bun este performanța și eficiența criptării care oferă o securitate excepțională a datelor, acestea nu pot fi decriptate fără cunoștința și acordul utilizatorului. Datele utilizatorilor au fost compromise o singură dată, din cauza unui vulnerabilități ce permitea executarea codului la distanță, acest lucru a devenit posibil din cauza unei vulnerabilități din clasa Java *DateRMI*, descrierea vulnerabilității poate fi găsită în *CVE-2017-9830*⁵.

3.5.5 iCloud

iCloud este unul dintre cele mai populare și utilizate sisteme de cloud conform CloudWards [18], acest fapt nu este datorat doar poziției monopolistice pe care o ocupă

⁵<https://www.cvedetails.com/cve/CVE-2017-9830/>

pe piață, ci și funcționalităților 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. Totuși, conform analizei funcționalităților, care este prezentată în tabelul 3.8 se poate observa că funcționalitățile acestuia se ridică la un nivel destul de înalt.

Tabelul 3.8: iCloud Funcționalități

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

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

Tabelul 3.9: iCloud Platforme disponibile și prețuri

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

Însă acesta are și câteva probleme legate de clientul Desktop, care are funcționalități limitate, și limitarea funcționalității de partajare de fișiere. **iCloud** oferă funcționalitatea de sincronizare cu Apple Photos. Iar funcționalitatea de share poate fi accesată din orice fișier care se află într-un director sincronizat.

În 2014 a avut loc un furt de date de dimensiuni foarte mari, acesta a compromis reputația iCloud, însă atacul a fost făcut prin forță brută și "pescuirea datelor" de la viitoarele victime. În realitate apple oferă câteva funcționalități de securitate care îl fac un sistem cu nivel de securitate peste media de pe piață. Însă Apple nu este un sistem cu "cunoștință zero", acesta stochează cheile de criptare în același loc cu fișierele criptate, asta îl 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 și nu va oferi informații private despre clienții săi.

iCloud nu este o soluție genrală pentru stocarea fișierelor, acesta este potrivit pentru utilizatorii care nu au nevoie să păstreze date extrem de sensibile din cauza posibilității de decriptare prin brute force. Acesta nu este potrivit nici pentru utilizatorii care doresc o viteză ridicată de încărcare și decărcare a fișierelor, însă Apple continuă să se perfecționeze și să crească viteza operațiilor de rețea.

De asemenea iCloud este extrem de vulnerabil pentru executare de cod la distanță conform datelor oferite de CVEDetails.com, vulnerabilitățile recent descoperite sunt *CVE-2018-20506*⁶ și *CVE-2018-4464*⁷, acest fapt este datorat popularității sistemului care îl face

⁶<https://www.cvedetails.com/cve/CVE-2018-20506/>

⁷<https://www.cvedetails.com/cve/CVE-2018-4464/>

o țintă 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 piață, această popularitate nu este datorată doar faptului că este preinstalat pe telefoanele Android, dar și capabilităților de partajare și vitezei înalte de decărcare și încărcare a fișierelor.

În tabelul 3.10 sunt prezentate evaluări ale funcționalităților sistemului.

Tabelul 3.10: Google Drive Funcționalități

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

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

În tabelul 3.11 sunt prezentate opțiunile de client disponibile pentru **Google Drive**, de asemenea fiecare utilizator primește inițial 10 GB de stocare gratuită.

Tabelul 3.11: Google Drive Platforme Disponibile și Prețuri

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 secțiunea 3.5.3. Punctul forte al colaborării oferite de Google Cloud este faptul că Office Suite este integrat în acest sistem.

Funcționalitatea de sincronizare este extrem de performantă, dar nu oferă sincronizare la nivel de bloc de fișier. Partajarea de fișiere este una dintre cele mai folosite funcționalități ale sistemului, însă acesta vine cu câteva vulnerabilități cauzate de lipsa criptării la partajare sau protejarea cu parolă. Partajarea este ușor de executat link-ul poate fi partajat prin email, Facebook, Twitter sau copiat și salvat în destinația dorită.

Datele sunt criptate utilizând AES-128 atunci când sunt salvate pe disc și prin TLS atunci când sunt transportate între client și server. Oricum sistemul nu oferă "zero-knowledge" și oricine deține are control asupra sistemului poate să citească datele (exemplu: programatorii sistemului). Totuși pentru o protecție mai bună Google oferă posibilitatea de autentificare în 2 pași.

Google Drive vine cu foarte multe aplicații integrate și posibilități de colaborare integrate în sistem. Autentificare în 2 pași, criptare a datelor și viteză ridicată de încărcare și descărcare, dar lipsa posibilității de criptare privată și vulnerabilitățile cauzate de lipsa unei criptări destul de sigure pentru funcționalitatea de partajare de fișiere fac **Google Cloud** să nu fie cea mai bună soluție pentru securitatea și 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 privința securității datelor, deși și-a perfecționat securitatea, nu poate fi încadrat în topul celor mai sigure sisteme de securitate[19], însă tinde spre acesta.

Analiza funcționalităților sistemului este prezentată în tabelul 3.12.

Tabelul 3.12: OneDrive Funcționalități

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

Funcționalitățile nu au primit o notă maximă din cauza ambiguității de utilizare a sistemului, în dependență de tipul fișierului selectat, meniurile arată diferit, uneori poate fi o problemă pentru utilizatori. De asemenea, sistemul nu este unul cu cunoștințe zero, atacatorul poate obține datele unui utilizator, odată ce s-a infiltrat în sistem, deoarece 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 și prețul unui abonament cu spațiu de stocare de 500GB.

Tabelul 3.13: OneDrive Platforme Disponibile și Prețuri

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

OneDrive urmează niște principii definite de Dropbox cu privire la standardele de sincronizare a fișierelor. OneDrive obișnuia să aibă probleme serioase cu privire la securitate, neavând nici criptare la nivelul nivelului de stocare. În prezent, însă sistemul oferă criptare la transmitere și la stocare. Criptarea nivelului de stocare include 2 componente esențiale: *BitLocker*, criptare la nivel de disc, și un sistem de criptare per fișier a conținutului. Fiecare fișier este securizat prin utilizarea unei chei AES unice de 256 de biți, de asemenea se folosește protocolul TLS pentru a preveni atacurile de tipul *man-in-the-middle*. Dar minusul acestui sistem este că cheile sunt stocate pe același 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 funcționalitate de compresie, ceea ce crește costul de stocare a datelor, datele pot fi compresate de utilizator în prealabil, iar după descărcare acestea pot fi decompresate, operația însă este anevoioasă și prezintă o problemă în cazul partajării fișierelor. Sistemul nu este potrivit pentru stocarea unor date sensibile și este recomandat doar pentru stocarea unor date netriviale.

3.5.8 pCloud

pCloud a fost fondat în 2013 și în doar 3 ani a ajuns la peste 3 milioane de utilizatori, competitorii cei mai importanți sunt Dropbox și Copy. Chiar dacă este nou pe piață, spre deosebire de competitorii săi mari, acesta oferă funcționalități de top și este inclus în topul *Most Secure Cloud Storage 2019: Safety First*[19].

pCloud oferă funcționalitățile de sincronizare, backup (se poate aplica și pe datele de pe Instagram sau Facebook), colaborare și criptare avansată a datelor. O evaluare a acestor capabilități este oferită în tabelul 3.14.

Tabelul 3.14: Funcționalități

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

Sistemul este prezent pe toate tipurile de platforme, tabelul 3.15 și are un preț 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 și să ofere funcționalități care nu sunt prezente la competitori. Sistemul folosește chei de 256 biti și TLS pentru transmisia de date. Securitatea însă vine și cu un preț, fișierele nu pot fi modificate în sistem, deoarece operațiile de criptare și decriptare ar trebui efectuate repetat, provocând costuri de prelucrare ridicate. De asemenea, sediul central al companiei se află în Elveția, unde sunt cele mai stricte reguli de protecție a datelor, astfel utilizatorul poate fi liniștit cu privire la integritatea datelor sale.

Datele încărcate sunt repartizate în funcție de tipul de date: imagini, documente, muzica și video.

Un dezavantaj al sistemului este că criptarea este o funcționalitate care nu este oferită în pachetul de bază și are un preț 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, și Darius Antia. Acesta a devenit deja un competitor serios pentru Google Drive și Dropbox. Caracteristicile pentru care a ajuns atât de popular sunt analizate în tabelul 3.16.

Tabelul 3.16: Funcționalități

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

Din păcate sistemul nu oferă compresia sau deduplicarea datelor, sistemul stocând volumul real de date pe care îl primește de la utilizator, plus câteva metadate create de algoritmi 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 și vizualizare a fișierelor, doar decărcarea și manipularea lor ulterioară cu alte aplicații. Însă există și opțiunea de a stoca fișierele fără criptare. Caracteristicile cheie ale sync.com sunt :

- Cunoștințe zero despre date
- Criptare privată
- Ușor de utilizat
- Sincronizare
- Partajare de fișiere cu control asupra operației

Sync este unul dintre sistemele care garantează securitate la nivelul cel mai înalt, acest sistem folosește pentru criptare RSA cu chei între 512 și 2048 biți. Sync nu păstrează cheile de criptare în sistem, dacă utilizatorul își uită parola atunci datele lui nu vor putea fi recuperate niciodată. De asemenea se poate activa și opțiunea de autentificare în doi pași, pentru a oferi o protecție și mai bună.

Sistemul oferă funcționalitatea de sincronizare, însă nu orice director poate fi selectat pentru efectuarea operației din cauza faptului că se iau în considerare particularitățile fiecărei aplicații. Sync oferă funcționalitatea de partajare de fișiere care poate fi configurată adăugând parolă sau timp de expirare.

Nu putem nega că sync.com oferă o funcționalitate de criptare complexă și eficientă care oferă o securitate avansată, dar, partea negativă a acestui lucru este că atunci când dorim să vizualizăm un fișier sau să îl descărcăm, operația ar putea dura între câteva secunde și câteva minute din cauza complexității adăugate de criptare.

3.5.10 Concluzii și plasarea sistemului

Volumul de date stocate în cloud a crescut cu un factor de 40 în ultimii 10 ani, creșterea este constantă. Evoluția tehnologică aduce un preț mai mic pentru componentele hardware de stocare, dar și cantități mai mari de date ceea ce împiedică scăderea prețului serviciilor. De asemenea, evoluția tehnologică aduce un impact negativ și asupra securității, un atac de forță brută poate fi executat mult mai ușor pe un sistem mai performant.

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

Tabelul 3.18: Comparatie sisteme similare

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

Majoritatea sistemelor oferă criptarea datelor, unele au metode imposibil de spart, altele au metode mai simple și puțin sigure. Sistemul propus criptează datele utilizatorului utilizând algoritmi auto-calibrabili care evoluează în funcție de sistemul pe care rulează aplicația sau de trecerea timpului. De asemenea sistemul folosește cei mai noi și siguri algoritmi TwoFish și PBKF2, care au fost modificați să folosească chei mai sigure de o lungime de 256 și 512 biți.

Un aport nou pe care îl aduce sistemul și nu a fost observat la niciuna dintre aplicațiile studiate este semnătura pe fișiere, pentru a detecta furtul de date prin extragerea codului - tehnica folosită este steganografia.

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

Capitolul 4

Analiză și Fundamentare Teoretică

4.1 Arhitectura conceptuală a sistemului

Această secțiune reprezintă arhitectura conceptuală a sistemului, prezentată în Figura 4.1.

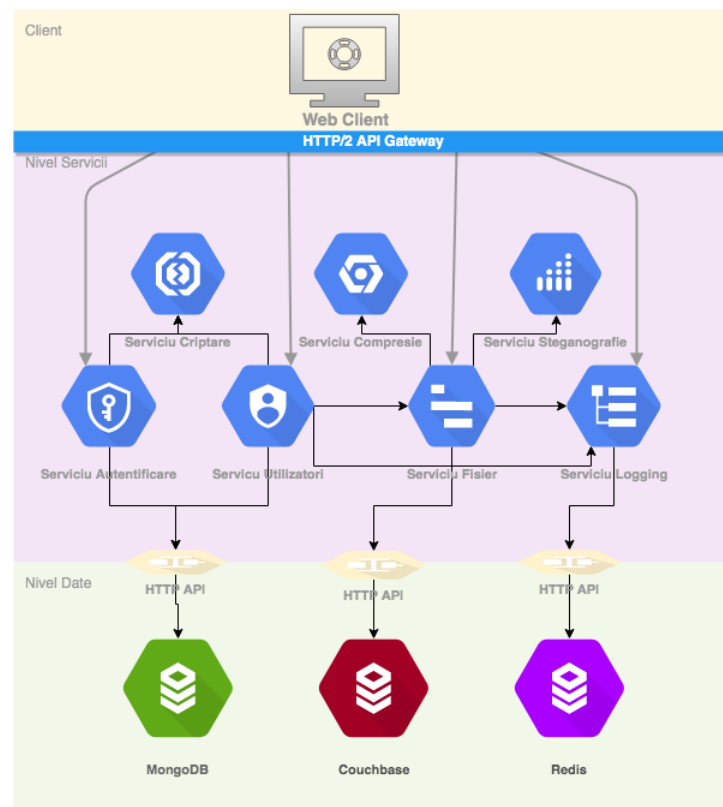


Figura 4.1: Arhitectura sistemului

Sistemul a fost împărțit în microservicii în funcție de principalele funcționalități ale sistemului. Divizarea sistemului în microservicii contribuie la *scăderea nivelului de cuplare* a componentelor și *creșterea coeficientului de coeziune*.

Aplicația client este de asemenea un microserviciu care assemblează serviciile serverului și expune funcționalitățile pentru client. Arhitectura clientului este reprezentată în Figura 5.10.

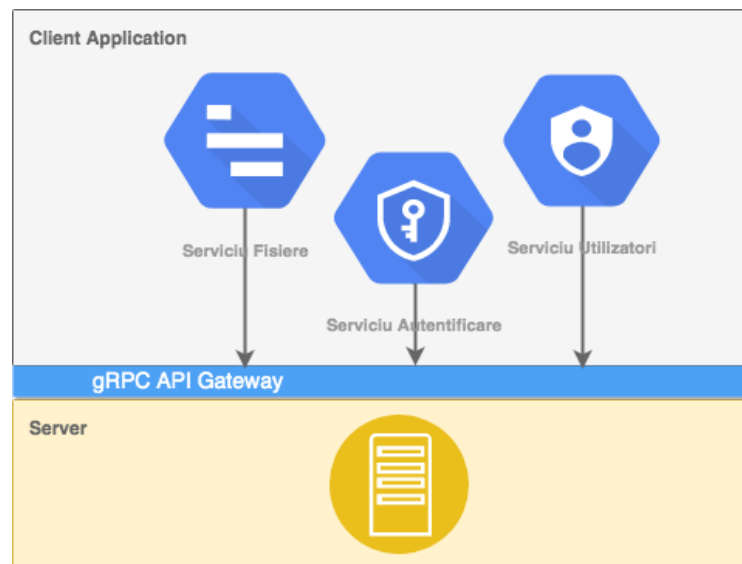


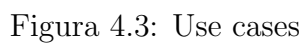
Figura 4.2: Arhitectura clientului

Serviciile din care este format clientul nu interacționează între ele și pot fi considerate entități separate. Acest lucru contribuie la scăderea numărului de erori cauzate de interacțiunea serviciilor.

4.2 Cazuri de utilizare

În această secțiune 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 și decărca fișiere.
- **Administrator** - pe lângă atribuțiile unui utilizator ordinar, are dreptul de a vizualiza toți utilizatorii primari și de a le bloca sau debloca contul.



Din diagramă se pot observa ramificările și compoziția din cadrul cazurilor de utilizare stabilite.

4.2.1 Modele de cazuri de utilizare

4.3 Tehnologii

În această secțiune sunt descrise tehnologiile folosite pentru implementarea proiectului și avantajele pentru care au fost selectate.

4.3.1 Golang

Golang este un limbaj de programare care a apărut în 2007, fiind și un proiect *open source*. Este un limbaj care oferă analiză statică a codului: *gofmt* - formatarea statică a codului, *golint* - stilizarea codului, *godoc* - documentarea codului, acestea oferă o siguranță asupra codului scris. În figura 4.4 este reprezentată emblema oficială a Golangului.

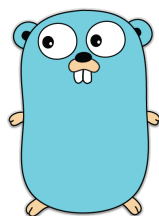


Figura 4.4: Golang

Go oferă un tool pentru testare integrat în limbaj, acesta a fost elaborat pentru simplitate și eficiență. Avantajul cheie este concurența, 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 eficiența managementului de memorie și 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 limbaj și 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 documentației și detecția unor eventuale erori, cum ar fi: erori netratate, variabile neutilizate, metode neapelate și lipsa verificărilor de tip înate de efectuarea unei conversii sau a unui *cast*.

Printre marile companii care utilizează Golang se numără: Google, YouTube, Facebook, Apple, Docker, Dropbox, Twitter, Netflix, Ethereum, Kubernetes, Uber, Couchbase, IBM și altele. Dropbox, sistem descris în secțiunea 3.5.3, a migrat câteva dintre funcționalitățile sale criptice spre Go în 2013. Sistemul a fost scris inițial în Python, scopul trecerii a fost creșterea performanței, creșterea numărului de clienți și viteza de dezvoltare software.

Datorită performanței, vitezei de procesare și răspuns, Golang a fost selectat ca limbaj primar pentru acest proiect. Un alt motiv ar fi și faptul că mai multe sisteme care efectuează operații cu volume de date destul de mari au optat pentru acest limbaj, lucru care denotă faptul că este potrivit pentru procesare de fișiere mari.

4.3.2 gRPC

gRPC este un *framework* RPC foarte performant, care poate rula pe orice platformă și în orice mediu. Acesta poate conecta într-un mod eficient serviciile din diferite centre de date, acesta suportă balansarea, urmărirea, verificarea funcționării corecte și autentificarea. Acesta este de asemenea aplicabil în cazul sistemelor distribuite pentru conectarea dispozitivelor, aplicațiilor mobile și 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 și scalat** - instalarea mediului de lucru se poate face în câțiva pași simpli și se poate scala la milioane de apeluri RPC pe secundă.
- **Compatibil cu multiple limbaje și platforme** - se poate genera în mod automat codul pentru server și client gRPC în numeroase limbaje, cum ar fi: Java, Go, C#, C++, JavaScript, Python.
- **Streaming bi-directional și autentificare integrată** - se poate face streaming bi-direcțional și dispune de autentificare integrată și configurabilă cu transport prin HTTP/2.

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

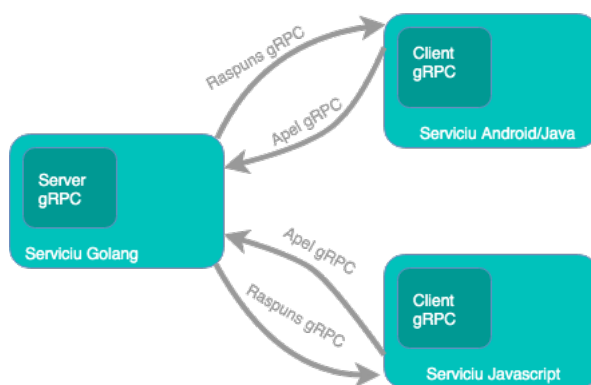


Figura 4.5: Exemplu de interacțiune gRPC

În gRPC o aplicație client poate apela metodele aplicației server care rulează pe alt dispozitiv de parcă ar fi metode locale, modelul de interacțiune a clientului și serverului este prezentat în Figura 4.5. Principiul principal al gRPC este definirea serviciilor, metodelor și parametrilor acestora prin *Protocol Buffers*, un exemplu de definiție a unui serviciu este prezent în Anexa A.

Porning de la definiția fișierului *.proto*, gRPC oferă compilatorul pentru fișierele protobuf care generează codul client și sever. Utilizatorii gRPC apelează API-urile de pe partea de client și 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 entități ce utilizează această metodă. API-urile create cu ajutorul protocolului gRPC sunt mult mai performante decât cele create cu ajutorul metodei clasice REST, timpul de răspuns micșorându-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 interfețe utilizator și aplicații web. Vue utilizează template-uri cu sintaxă bazată pe HTML care permit randarea paginilor utilizând asocierea elementelor cu datele de instanță ale componentei Vue a aplicației.

Vue folosește un sistem reactiv bazat pe Javascript pur care permite optimizarea re-randării. Acest sistem folosește urmărirea dependențelor, pentru a ști când trebuie re-randate componentele și care dintre ele trebuie randate. La fel ca majoritatea framework-urilor de JS, Vue oferă suport nativ pentru TypeScript, fapt care în oferă și 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 aplicațiilor web.
- **Flexibilitate** - aplicațiile Vue pot fi rulate ușor din browser, fiecare schimbare în cod este detectată și aplicația este recompilată automat. Această abordare este deosebit de utilă în timpul testării aplicațiilor.
- **Versatilitate în privința dimensiunii aplicațiilor** - Vuex este un feature al Vue.js care este folosit pentru managementul bazat pe stare și funcții de rutare. Acesta poate fi folosit pentru construirea unor funcții mai largi și mai complexe.

- **Integrare simplă** - framework-ul are niște capacități de integrare uimitoare cu aplicațiile existente, fapt datorită căruia i-a crescut popularitate în comitățile de developeri.

Avantajele framework-ului l-au făcut să câștige popularitate încă de la apariția acestuia, printre utilizatorii Vue.JS se numără Adobe, Gitlab, Netflix, Xiaomi, Codeship și alții. Acesta este foarte flexibil și versatil, dar fiind foarte nou, are o comunitate destul de restrânsă de programatori, unele funcționalități și 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 performanța, nivelul ridicat de disponibilitate și scalabilitate. Baza de date este container-ul fizic pentru colecții. Colecțiile la rândul lor sunt un grup de documente. Documentele sunt un set de perechi cheie valoare. Documentele au o schemă dinamică și nu trebuie să aibă aceeași 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 și grupa datele în orice mod. Structura unui obiect este foarte simplă și clară.
- **Flexibilitate query-urilor** - Se pot executa filtrări complexe, datorită unui limbaj de *query* foarte versatil.
- **Este ușor de scalat** - scalarea bazei de date poate fi configurată foarte ușor.
- **Suport pentru diferite tipuri de date** - datorită structurii datelor, nu mai este necesară conversia sau maparea datelor aplicației la un tip specific, acestea sunt convertite automat în BSON.
- **Este bine documentat** - fiind foarte popular, MongoDB dispune de o documentație foarte bine definită.
- **Indexarea** - se pot crea indecși, aceștia contribuie la creșterea vitezei de căutare.

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

4.3.5 Couchbase

Bazele de date NoSQL au schimbat procesul de dezvoltare a aplicațiilor software din punctul de vedere a adaptabilității schemei dinamice și a scalabilității. Comparând între ele sistemele de baze de date NoSQL existente, Couchbase[20] este una dintre cele mai rapide. Este ușor de configurat și are niște funcționalități puternice pentru stocarea datelor cu diverse structuri. Pentru extragerea datelor utilizează 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-un singur pas în cazul în care se cunoaște cheia obiectului, în cazul unor operații mai complexe, timpul de procesare crește. Prin adăugarea indecșilor se poate eficientiza aproape orice operație.

Echivalentul termenului bază de date din terminologiile relaționale, în termenii Couchbase, este numit *bucket*, acesta este un container logic pentru documentele grupate logic. Couchbase oferă câteva mecanisme de configurare, interacțiune și monitorizare a clusterelor. Acestea sunt: interfață grafică web pentru administrator, administrarea utilizând REST API și interfață de control din linia de comandă. Datorită acestor funcționalități, administratorul, poate configura, edita și expanda baza de date într-un mod simplu și eficient.

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

Viteza, nivelul înalt de securitate, viteza de operare a datelor și configurabilitatea fac Couchbase să fie cel mai potrivit sistem pentru stocarea fișierelor din aplicație. Un avantaj semnificativ sunt funcționalitățile 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 definește structura și identitatea conținutului 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 afișate pe ecran, hârtie sau alt tip de media. Se poate configura modul de afișare, poziționarea și efectele elementelor.

Bootstrap este un framework de front-end care crează condiții pentru o dezvoltare mai rapidă a paginilor web. Acesta include template-uri HTML și CSS pentru design-ul elementelor tipografice, butoanelor, formurilor, tabelelor, modalurilor, imaginilor și multor

¹<https://developer.mozilla.org/en-US/docs/Web/HTML>

²<https://developer.mozilla.org/en-US/docs/Web/CSS>

altor elemente. Datorită acestui framework se pot crea design-uri customizable și foarte responsabile.

CSS, împreună cu HTML și 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, doar cele disponibile în HTML: listă, tabel, div-uri.

4.3.7 JSON Web Token(JWT)

JWT³ este un standard(RFC 7519⁴) care definește transmisia compactă și securizată a informațiilor între 2 sisteme sub formă de obiect JSON. Această informație poate fi verificată și considerată de încredere, deoarece are o semnătură digitală. JWT-urile pot fi semnate folosind un secret, prin metoda HMAC sau printr-un algoritm de cheie publică/privată folosind RSA.

Contextele în care sunt folosite JWT sunt:

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

Docker⁵ este singura platformă compilare end-to-end, partajare și rulare a aplicațiilor bazate pe containere. Acesta poate efectua managementul unei întregi aplicații din momentul compilării pe calculatorul programatorului până la pasul în care ajunge pe cloud. Docker este cea mai rapidă soluție pentru deplomentul unei aplicații spre producție. Acesta oferă suport securizat pentru livrarea aplicațiilor pe orice tip de cloud - de la cloud hibrid până la muchie.

Kubernetes⁶ este un sistem de orchestrare open-source, folosit pentru management, plasarea, scalarea și rutarea containerelor. Platforma Docker oferă un mediu Kubernetes securizat și foarte performant, acesta este potrivit pentru atingerea unor scopuri comune în dezvoltarea software și oferă maximă flexibilitate.

³<https://jwt.io>

⁴<https://tools.ietf.org/html/rfc7519>

⁵<https://www.docker.com>

⁶<https://www.docker.com/products/kubernetes>

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

- Poduri - un grup de containere care conțin diverse microservicii ce pot fi conectate ca un întreg.
- Kubernetes oferă load-balancing, identificarea și descoperirea microserviciilor pentru a putea fi izolate.
- Se poate configura nivelul de interacțiune între servicii.

Se poate efectua decuplarea aplicației de mașina fizică, astfel mai multe microservicii pot conviețui pe același 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 aplicațiile sunt construite și consumate în cloud. Datorită combinației dintre Docker și Kubernetes sistemul poate avea un grad de *velocitate*, *eficiență* și *agilitate* mult mai înalt. Folosirea Docker și Kubernetes crește nivelul de eficiență și securitate a unei aplicații, acestea sunt indispensabile pentru construirea unui mediu securizat de stocare și lucru.

4.3.9 Google Cloud

Costul resurselor hardware este foarte mare, iar sistemele sunt foarte greu de scalat. Sistemele cloud sunt soluția cea mai bună pentru problema respectivă. Google Cloud⁷ este o soluție foarte populară pentru servicii în cloud. Principalele avantaje sunt: nivelul înalt de securitate, criptarea, protecția identității și multe servicii auxiliare pe care le oferă.

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

- **Preț mai scăzut decât competitorii** - pentru o aplicație scalabilă, la început de drum, este o soluție fezabilă și eficientă.
- **Monitorizarea și migrarea în timp real a mașinilor virtuale** - nici unul dintre marii competitori nu oferă această funcționalitate. Avantajul oferit constă în faptul că reparația, scalarea și actualizarea părților hardware sau software se poate efectua foarte rapid.
- **Performanță sporită** - poate suporta până la 60000 de apeluri concurente
- **Securitatea** - toate datele sunt criptate, toate accesările la componente pot fi efectuate doar după autorizare.

Utilizarea platformei Google Cloud oferă numeroase avantaje. Nivelul de securitate oferit, precum și performanța sporită îl transformă într-o soluție 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 și modifica diverse schimbări.

Git⁸ 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 și siguranța. *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 și un număr nelimitat de proiecte hostate în mod gratuit.

⁸<https://git-scm.com/about>

⁹www.github.com

Capitolul 5

Proiectare de Detaliu și Implementare

Acest capitol prezintă deciziile și pașii de implementare parcurși în ciclul de dezvoltare al proiectului. Sistemul propus este format din 3 subsisteme: aplicația web; serverul, care este format din alte subsisteme, și bazele de date. Capitolul va oferi o descriere succintă a tuturor componentelor, incluzând șabloanele arhitecturale, șabloanele de design și algoritmi folosiți în dezvoltarea proiectului.

5.1 Arhitectura serverului

5.1.1 Descriere generală

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

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

Arhitectura sistemului este prezentată în figura 5.1

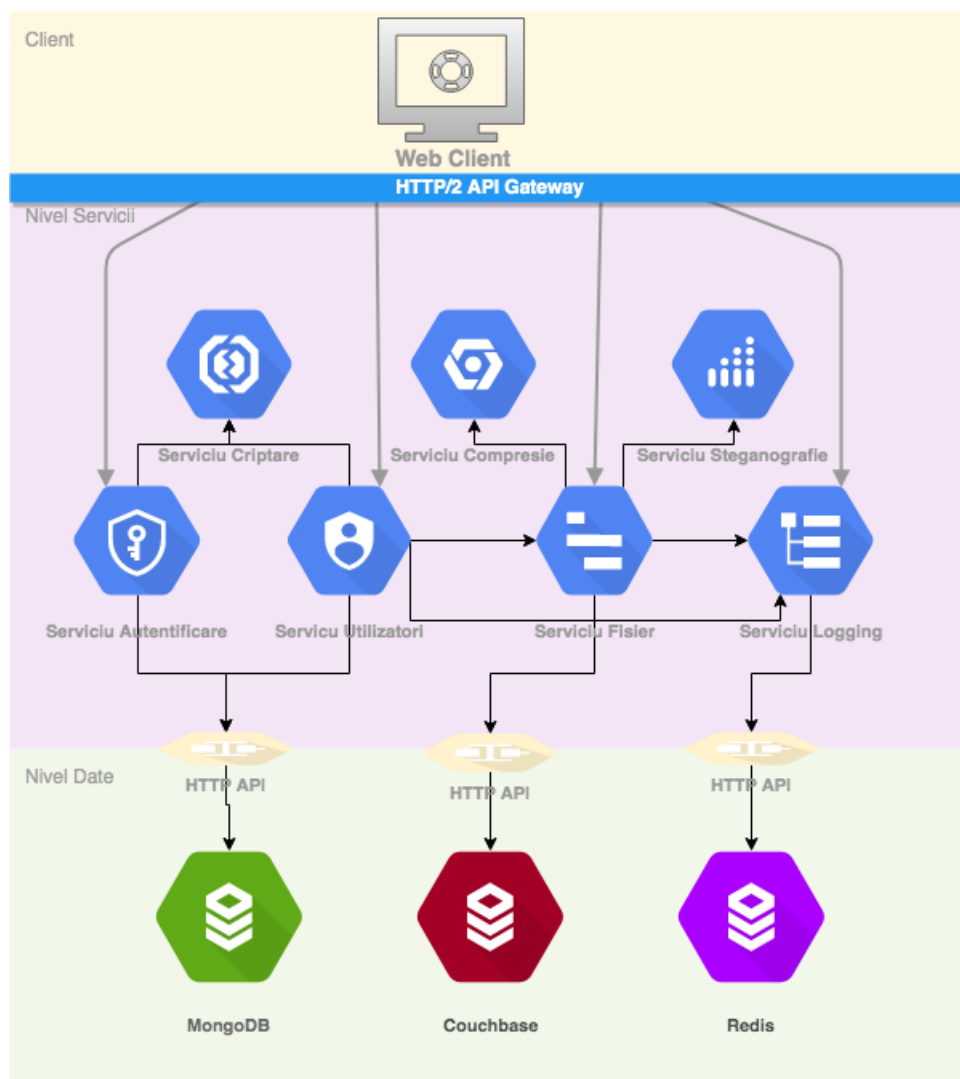


Figura 5.1: Arhitectura sistemului

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

5.1.2 Orchestrarea microserviciilor

Microserviciile sunt o modalitate de a despărți funcționalitățile dintr-un sistem. Acestea ne oferă flexibilitatea de a scala funcționalități specifice și de a fi agili în livrarea produselor. După ce funcționalitățile au fost separate în subsisteme dedicate, întrebarea

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

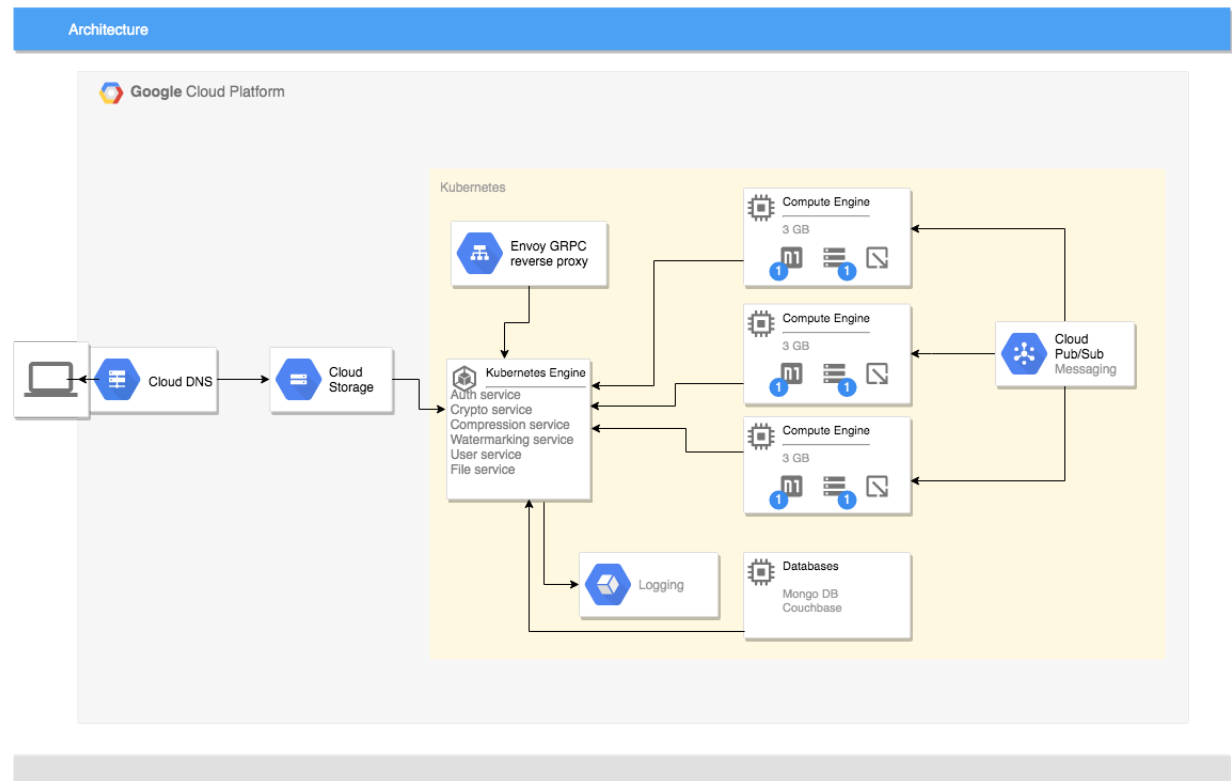


Figura 5.2: Diagrama orchestrării în Cloud

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 necalitativă, dificultate crescută de înțelegere și costuri crescute de consum.

5.1.3 Microserviciul de autentificare

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



Figura 5.3: Arhitectura serviciului de autentificare

Nivelul de acces date efectuează operații pentru identificarea și validarea utilizatorilor în funcție de datele de autentificare pe care le-a oferit utilizatorul. Baza de date folosită este *MongoDB*, serviciul accesează aceeași colecție ca și serviciul de utilizatori. Pentru operația de generare a unui Token de sesiune se caută utilizatorul cu credențialele care au fost primite de la utilizator, dacă sunt valide se returnează un Token JWT în care sunt encode datele utilizatorului. Pentru validarea tokenului, la acest nivel, se execută operația de verificare dacă acesta este existent în baza de date și 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 niște reguli stricte de format și securitate.

La nivelul *Logicii de Business* se execută operații de encodare și decodare a token-urilor, se gestionează accesul la resurse. De asemenea, 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 conține implementarea interfeței de comunicare gRPC, fiecare metodă definită în protobuf este implementată și poate fi apelată de un client gRPC.

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

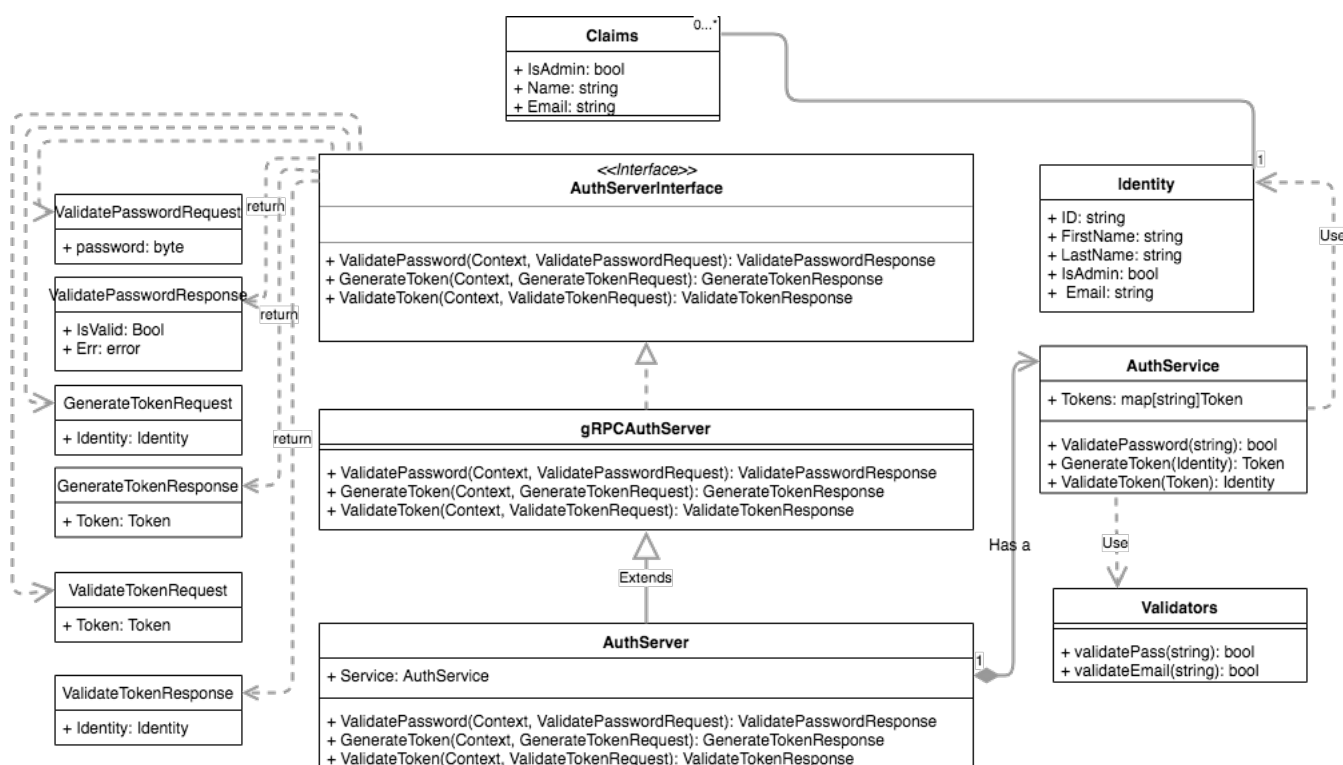


Figura 5.4: Diagrama de clase a serviciului de autentificare

Cea mai importantă clasă este `AuthServer`, acesta extinde `gRPCAuthServer`, care este o implementare a interfeței definite în fișierul `service.proto`.

5.1.4 Microserviciul de criptare

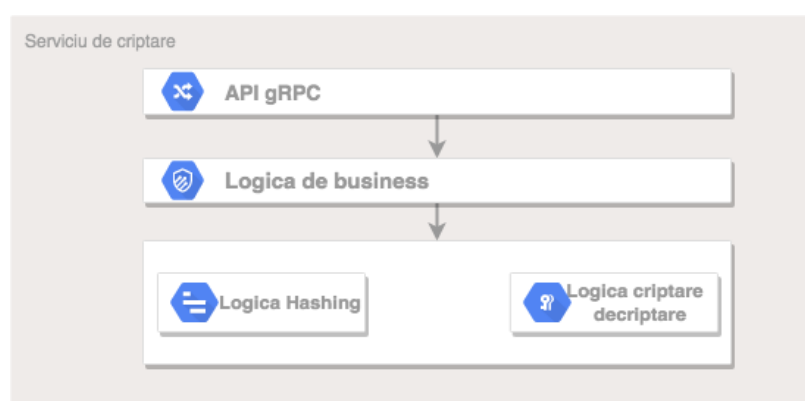


Figura 5.5: Arhitectura serviciului de criptare

5.1.5 Microserviciul de steganografie și marcare

Steganografia este știința ascunderii mesajelor, numele provenind de la cuvintele gresești *steganos*, care înseamnă protejat, ascuns, și *graphein*, care înseamnă a scrie.



Figura 5.6: Arhitectura serviciului de marcare

5.1.6 Microserviciul de compresie

Microserviciul de compresie oferă 3 funcționalități: compresie text, compresie PNG și compresie JPEG. Arhitectura acestuia este prezentată în Figura 5.7.

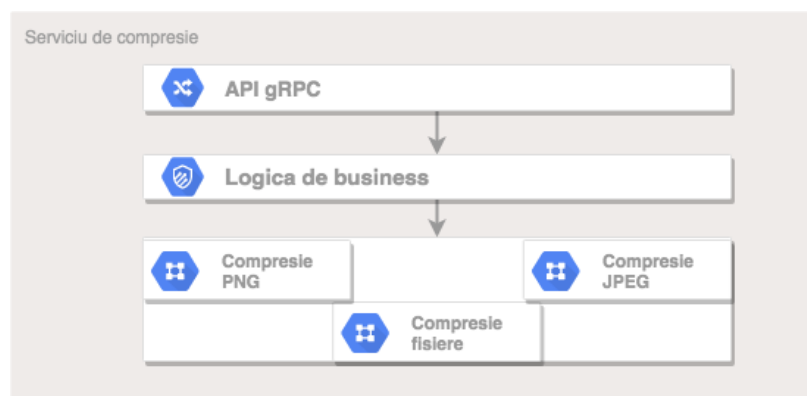


Figura 5.7: Arhitectura serviciului de compresie

Compresia de text este bazată pe zlib care folosește algoritmul DEFLATE, acesta permite utilizarea unui număr minim de resurse pentru a obține un rezultat cât mai bun. Algoritmul este unul fără pierderi. Rezultatul bun este obținut datorită folosirii arborilor de codificare Huffman, deoarece se crează un arbore Huffman optimizat pentru fiecare block de date. Totuși folosirea Huffman este recomandată mai mult pentru mesaje mici, deoarece aceasta introduce pentru fiecare block niște instrucțiuni de decompresie. Compresia este obținută prin 2 pași:

- Potrivirea și înlocuirea datelor duplicate
- Înlocuirea simbolurilor cu altele noi bazate pe frecvența de apariție

Dimensiune fișier text (octeți)	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 5.1: Rata de compresie fișiere text

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

Tabelul 5.2: Rata de compresie PNG entropie mică

În cazul imaginilor cu o entropie mică, de exemplu cele care contin text sau au un procent de fundal foarte mare se obține o rată de compresie foarte bună cu un factor cuprins între 20 și 1000. Acest lucru contribuie la scăderea prețului de stocare per octet, însă introduce un cost mediu spre mare de timp pentru imaginile foarte mari.

	Imagine mică	Imagine medie	Imagine mare	Imagine foarte mare
--	--------------	---------------	--------------	---------------------

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

Tabelul 5.3: Rata de compresie PNG entropie mare

În cazul imaginilor cu o entropie mare compresia crește dimensiunea fișierului prin adăugarea de metadate, iar dimensiunea datelor rămâne aceeași. În cazul unor imagini foarte mari de acest tip se poate introduce o întârziere de până la 35 de secunde fără a aduce impact asupra prețului de stocare. Însă aceste imagini se întâlnesc extrem de rar, aplicația ar fi afectată foarte puțin de acest fenomen.

5.1.7 Microserviciul de fișiere



Figura 5.8: Arhitectura serviciului de fișiere

5.1.8 Microserviciul de utilizatori

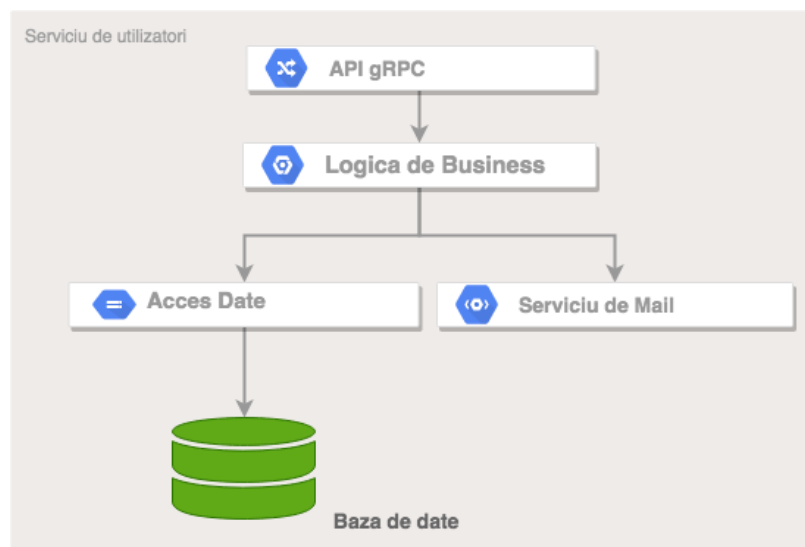


Figura 5.9: Arhitectura serviciului de utilizatori

5.2 Arhitectura aplicatiei web

5.2.1 Descriere generală

Aplicația web a fost scrisă utilizând framework-ul Vue. În figura 5.10 este prezentată arhitectura clientului, acesta a fost structurat ca 3 microservicii separate. Fiecare microserviciu comunică cu serviciile serverului prin gRPC Gateway construită cu ajutorul Envoy.

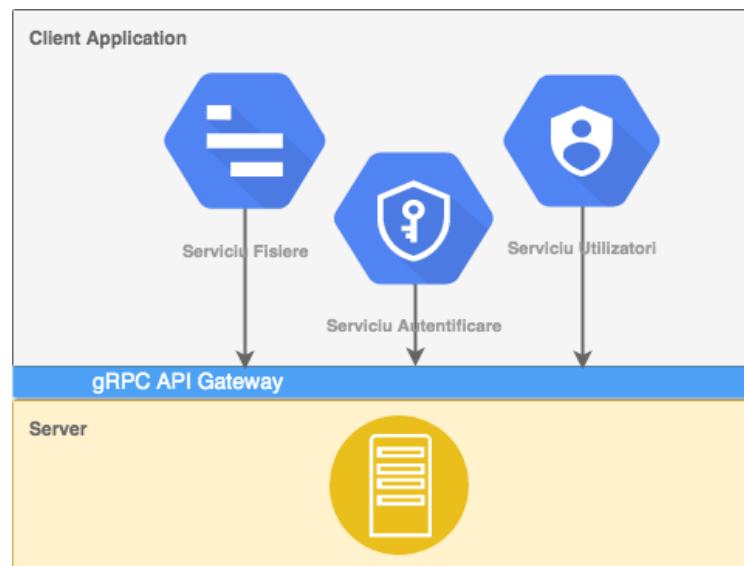


Figura 5.10: Arhitectura clientului

5.2.2 Descrierea componentelor

Capitolul 6

Testare și Validare

Acest capitol conține descrierea metodei de testare și validate a sistemului. Sunt descrise toolurile care au fost folosite pentru testarea serverului și a clientului. Deasemenea este analizat procentul de acoperire a codului prin teste.

6.1 Testarea serverului

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

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

6.1.1 Reguli de testare în Golang

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

Scrierea unor teste bune nu este un lucru trivial, în multe situații sunt foarte multe cazuri de acoperit, ceea ce înseamnă cătrebuie scrise multe teste, **Golang** rezolvă această problemă prin adăugarea de *teste tabelare*, acestea permit crearea testelor imbricate, care pot fi rulate în paralel, astfel fiind mai ușor de detectat "*data race*". Acest tip de testare are un avantaj enorm deoarece permite scrierea unui cod citibil și foarte ușor de înțeles. Fiind integrate în limbaj, acest tip de teste pot fi create înainte de cod, imediat după definirea semnăturilor de metode, ceea ce este ideal pentru o abordare *Test Driven Development*.

De asemenea, **Golang** permite adăgarea de funcții ajutătoare, a căror ordine și frecvență de rulare poate fi condiționată, acestea pot fi precondiții/postcondiții globale sau locale.

6.1.2 Testarea serviciilor

Pentru fiecare serviciu au fost generate un număr 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 secțiunea de mai jos este prezentat un exemplu de test pentru inițierea 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 numește *Go Cover*. În tabelul 6.1 este reprezentată rata de acoperire

cu teste a fiecărui serviciu în parte, unele părți nu au putut fi acoperite, acestea conținând tratări de erori care vin din alte librării folosite, o situație de apariție 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 fișiere	90%	80%	99%
Serviciu steganografie	81%	77%	86%
Serviciu utilizatori	85%	83%	60%

Pentru fiecare metodă și funcționalitate a serverului au fost generate toate combinațiile posibile de date, valide și invalide, pentru a detecta cât mai multe probleme și a le fixa, sau pentru a avea siguranța că totul funcționează cum este de așteptat.

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

6.2 Testarea clientului

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

Cu ajutorul Unit testelor s-a testat funcționarea corectă a modulelor aplicației și a logicii de prelucrare a datelor. Pentru fiecare funcționalitate au fost scrise teste care ar trebui să urmeze calea de succes și teste care conțin date eronate și ar trebui să producă eroare, fără a modifica starea aplicației, și fără a cauza erori în funcționare.

Pentru partea de teste e2e a aplicației client s-a folosit framework-ul *Nightwatch*, acesta este un *framework* scris în *Node.js* și este o soluție performantă pentru testarea aplicațiilor web. *Nightwatch* este potrivit pentru proiectele care adoptă calea livrării continue. Cu ajutorul acestui framework am testat interacțiunea clientului cu serverul și că toate acțiunile și toate datele se propagă corect între module și servicii.

Prin testare manuală s-a testat modul de interacțiune a aplicației cu utilizatorul. Acest lucru s-a efectuat prin parcurgerea căilor de evenimente posibile, erorile s-au difunționate detectate devenind noi sarcini de îndeplinit în procesul de dezvoltare a proiectului.

Capitolul 7

Manual de Instalare și Utilizare

În acest capitol sunt prezentate cerințele și pașii de configurare a aplicației din perspectiva unui dezvoltator software și pașii de utilizare din perspectiva unui client al sistemului.

7.1 Cerințe sistem

Pentru a obține un mediu optim pentru rularea aplicației, calculatorul dezvoltatorului software trebuie să îndeplinească următoarele cerințe:

- **CPU:** Intel Core i5 sau mai mult
- **Frecvență:** 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 aplicației web, cerințele se limitează la cerințele browser-ului și cele de RAM.

7.2 Instalare si configurare

Atât aplicație client, cât și cea server sunt hostate pe GitHub, acestea se află în repository separate. Toate sub-proiectele necesare se găsesc reunite în aceeași organizație. Primul pas constă în clonarea fiecărui proiect pe mașina locală, fiecare subproiect conține

un fișier în care sunt descrise cerințele de configurare. În primul rând se vor configura microserviciile pentru back-end, iar apoi se va configura aplicația client.

În continuare vor fi descriși etapele de configurare ale ambelor părți ale proiectului.

Pasul 1 Obținerea unei copii locale a proiectelor

- Se va descărca și instala GIT¹
- Se va accesa **bitstored**, toate repository sunt publice și disponibile în organizația respectivă
- Se vor clona toate proiectele listate în organizație într-un director local. Pentru clonare se pot utiliza clienți de Git sau se poate rula comanda **git clone \$REPO_URL** pentru fiecare subproiect

Pasul 2 Instalarea și configurarea bazelor de date

- Se va instala și configura **Couchbase**², conform pașilor 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 fișierul *server/repository/scripts/create_cluster.sh*
- Serverul de bază de date va fi accesibil la adresa *localhost:8091*, din interfața de administrator se pot gestiona datele, se pot crea noi clustere și se pot vedea metrice.
- Pentru a putea utiliza Couchbase ORM din limbajul Golang, se va instala următoarea librărie

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

- Se va instala și configura **MongoDB v4.x**³, pentru instalare se vor urmări pașii de la adresa <https://docs.mongodb.com/manual/installation/>
- Opțional se va instala Robo3T⁴, acest IDE permite vizualizarea, monitorizarea și editarea înregistrărilor din baza de date Mongo
- Se va porni serverul de MongoDB utilizând comenzile:

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

- O instanță de MongoDB va fi accesibilă la adresa *localhost:27017*

¹<https://git-scm.com/>

²<https://couchbase.com>

³<https://mongodb.com>

⁴<https://robomongo.org/>

- Pentru a fi accesibilă din Golang, se va instala un pachet MongoDB + Driver pentru Go, se va utiliza comanda ce urmează

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

Pasul 3 Configurarea și rularea aplicației server

- Se va instala **Golang**⁵, versiune mai nouă de 1.12
- Se va instala **Visual Studio Code**⁶ și se vor adăuga extensii de Golang, sau se va instala **Vim** și se va configura pentru suportul libajului Golang, sau se va instala editorul dedicat **Goland**⁷
- Se va descărca și instala **Docker**⁸
- Se va instala Vue.JS⁹ urmând pașii din următorul tutorial <https://vuejs.org/v2/guide/installation.html>
- Se va instala **Envoy**¹⁰
- Se va instala compilatorul de fișiere proto numit **protoc**¹¹
- Înainte de rularea proiectului se vor instala, configura și 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 rădăcină a proiectului și 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:

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

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>

Pasul 4 Configurarea și rularea aplicației client

- Se va deschide o sesiune nouă de terminal și se va naviga în directorul aplicației client
- Se va rula aplicația, utilizând comenzile:

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

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

7.3 Instrucțiuni de utilizare

În această secțiune se va face o prezentare generală a aplicației, se va prezenta conținutul web al fiecărei pagini care poate fi accesată de utilizatori, în funcție de rol: utilizator neînregistrat, utilizator autentificat sau administrator. Va fi prezentat arborele de navigare a aplicației, cu scopul de a obține o viziune asupra cazurilor de utilizare.

7.3.1 Arborele de navigare

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

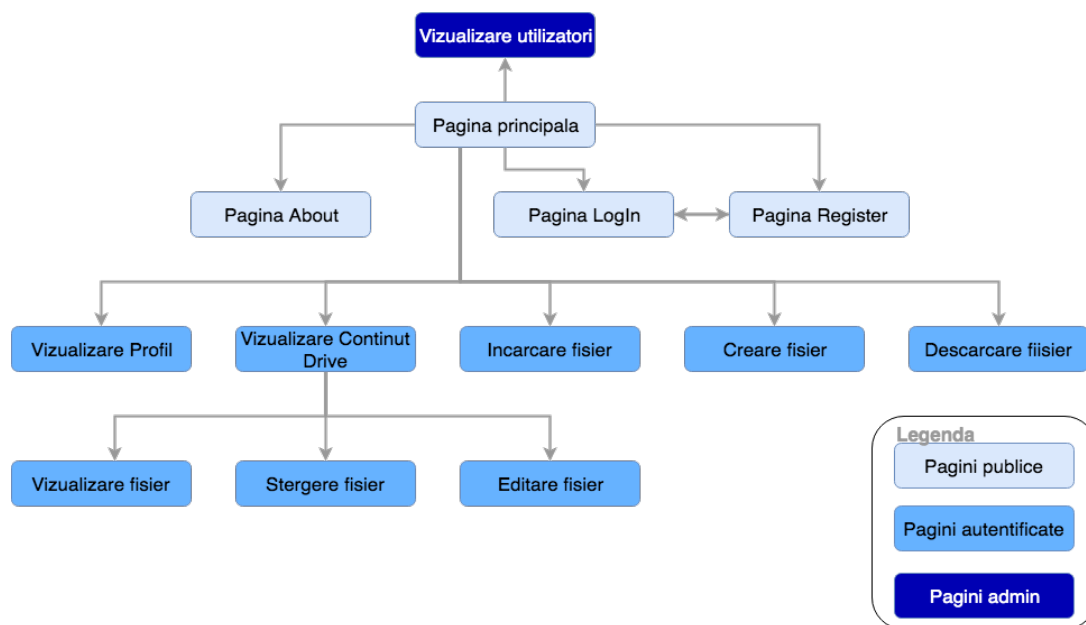


Figura 7.1: Arborele de navigare al aplicației

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

7.4 Navigare

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

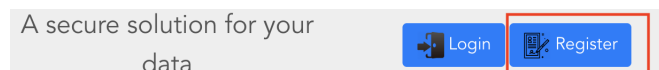


Figura 7.2: Butonul de Register

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

A registration form titled "Register" in bold black text. The form contains several input fields with labels above them: "First Name" (with "John" entered), "Last Name" (with "Doe" entered), "Photo" (with a "Choose file" button and "photo.png" text), "Email" (with "john.doe@email.com" entered), "Username" (with "john.doe" entered), "Password" (with a masked field of dots), "Phone number" (with "0730485910" entered), and "Birthday" (with "28/02/1900" entered). At the bottom right of the form are two buttons: "Register" in blue and "Cancel" in light blue.

Figura 7.3: Exemplu înregistrare

Condiția principală pentru ca o înregistrare să fie posibilă este ca email-ul, numele de utilizator și 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 aplicației spre ce pagină dorește să navigheze. Pentru a efectua operații cu fișiere se va naviga la pagin *My Drive*. În Figura 7.4 este reprezentată pagina principală a Drive-ului.

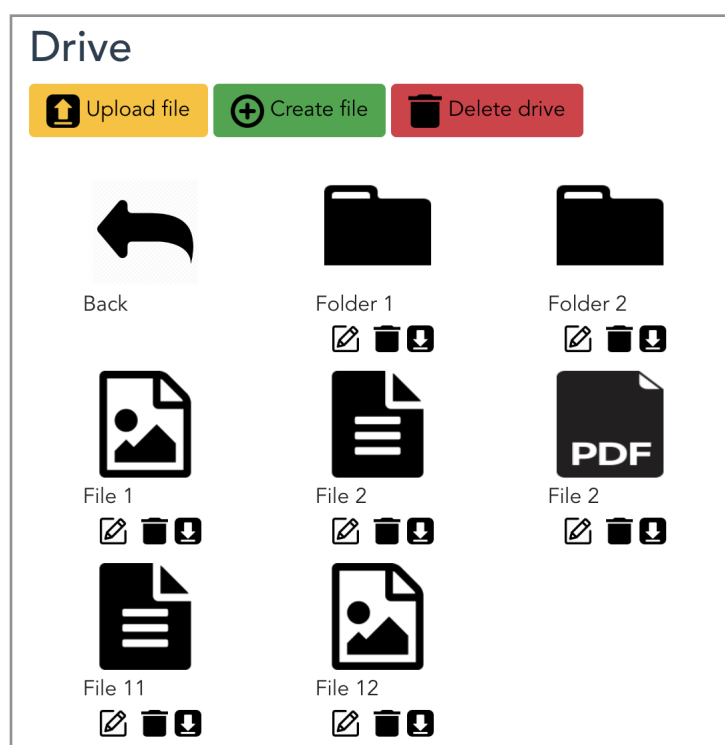


Figura 7.4: Vizualizare drive

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

Capitolul 8

Concluzii

În acest capitol vor fi descrise rezultatele obținute, contribuțiile aduse și deasemenea ideile de dezvoltare ulterioară și îmbunătățire a aplicației.

8.1 Rezultate obținute

Scopul proiectului a fost dezvoltarea unei aplicații securizate de stocare a datelor. Cerințele de bază au fost îndeplinite, aplicația având funcționalitățile cheie ale unui sistem de stocare de fișiere clasic.

Comparativ cu sistemele analizate în Secțiunea 3.5 are un set de funcționalități mai puțin numeros, dar implementarea acestora este bazată pe ideea menținerii unui nivel înalt de securitate a datelor. Sistemul permite compresia eficientă a datelor, prețul de stocare scăzând astfel de 4-5 ori în comparație cu sistemele analizate în Secțiunea 3.5.

8.2 Contribuții

Aplicația obținută oferă funcționalități de încărcare și descărcare a fișierelor, acesta fiind un sistem cu zero cunoștințe 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 fișierelor text și a imaginilor s-a obținut reducerea considerabilă a spațiului de stocare în cazul fișierelor mari. De asemenea adăugarea compresiei a contribuit la creșterea securității aplicației prin creșterea nivelului de complexitate pentru decriptare.

8.3 Dezvoltări ulterioare

Proiectul are funcționalitățile minime care îl califică în rândul aplicațiilor de stocare a fișierelor. Pentru îmbunătățirea modului de funcționare se pot implementa următoarele idei:

- Deoarece operațiile de criptare și compresie, precum și operațiile inverse, sunt foarte costisitoare în cazul unor date mari ca volum și cu rată de repetare mică, o soluție care ar eficientiza procesul ar fi combinarea celor 2 operații în una singură. Acest efect se poate obține prin adăugarea unor permutații pseudoaleatoare în procesul de compresie al datelor[21].
- Se pot eficientiza funcționalitățile de încărcare și descărcare a datelor prin adăugarea operației de despărțire în blocuri, acestea fiind transmise alternativ, operațiile de transfer prin rețea sunt cunoscute a fi cele mai încete operații de manipulare a datelor. O astfel de abordare ar putea reduce extrem de mult operațiile de transmisie, dar și cele de criptare și compresie, datorită faptului că operațiile pe blocuri ar putea fi efectuate în paralel, în loc de serial.
- Deoarece o parte dintre aplicațiile existente sunt vulnerabile la execuție de cod malign sau corupere de date, o funcționalitate care ar putea crește nivelul de securitate a aplicației ar fi adăugarea funcționalității de antivirus. Astfel fiecare încărcare de fișier ar fi validată în prealabil și fișierul ar ajunge în spațiul de stocare doar după ce ar fi marcat ca unul sigur.
- Dat fiind faptul că mulți hackeri utilizează atacurile prin forță brută, aplicația poate fi expusă la acest tip de atacuri, o soluție pentru acest pericol ar fi adăugarea autentificării în 2 pași.
- O funcționalitate de care poate beneficia aplicația este monitorizarea activității utilizatorilor și analiza acestora printr-un algoritm de inteligență artificială, astfel se pot bloca automat utilizatorii care au activități suspicioase sau se pot prezice viitoarele acțiuni sau procesul de creștere 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] 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.
- [11] A. Haecky and C. McAnlis, *Understanding Compression*. OReilly Media, Inc., 2016.
- [12] C.-N. Yang and S. Cimato, *Visual Cryptography and Secret Image Sharing*. CRC Press, 2017, vol. 1.
- [13] “Best cloud storage.’ [Online]. Available: <https://www.techradar.com/news/the-best-cloud-storage>
- [14] 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.

- [15] “Cloudme documentation.’ [Online]. Available: <https://www.cloudme.com/>
- [16] “Cloudwards - best cloud storage.’ [Online]. Available: <https://www.cloudwards.net/best-cloud-storage/>
- [17] “Crashplan review.’ [Online]. Available: <https://www.bestbackups.com/blog/5210/crashplan-review-2/>
- [18] “Icloud drive review.’ [Online]. Available: <https://www.cloudwards.net/review/icloud-drive/>
- [19] “Most secure cloud storage 2019: Safety first.’ [Online]. Available: <https://www.cloudwards.net/most-secure-cloud-storage/>
- [20] H. Potsangbam, *Learning Couchbase*. PacktPub, 2015, vol. 1.
- [21] C.-E. Wang, “Cryptography in data compression,’ *CodeBreakers Journal*, vol. 2, no. 3, pp. 1–20, 2005.

Anexa A

Secțiuni relevante din cod

Exemplu de fișier *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;
}
message CompressImageResponse {
  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

1.1	Creșterea volumului de date 2010-2025	1
1.2	Triada CIA	2
2.1	Arhitectura Client-Server	4
2.2	Exemplu de steganografie pe imagini	5
2.3	Tipurile de utilizatori ale sistemului	7
3.1	Arhitectura monolitică	11
3.2	Arhitectura bazată pe microservicii	13
3.3	Diferența dintre arhitecturi	14
3.4	Diferența dintre arhitecturi	18
3.5	Exemplu de encodare a unui byte	20
3.6	CloudMe - interfața web	22
3.7	CloudMe - planuri de preț	23
3.8	Dropbox - interfața web	24
4.1	Arhitectura sistemului	34
4.2	Arhitectura clientului	35
4.3	Use cases	36
4.4	Golang	37
4.5	Exemplu de interacțiune gRPC	38
5.1	Arhitectura sistemului	46
5.2	Diagrama orchestrării în Cloud	47
5.3	Arhitectura serviciului de autentificare	48
5.4	Diagrama de clase a serviciului de autentificare	49
5.5	Arhitectura serviciului de criptare	49
5.6	Arhitectura serviciului de marcaje	50
5.7	Arhitectura serviciului de compresie	51
5.8	Arhitectura serviciului de fișiere	54
5.9	Arhitectura serviciului de utilizatori	54
5.10	Arhitectura clientului	55

7.1	Arborele de navigare al aplicației	62
7.2	Buttonul de Register	63
7.3	Exemplu înregistrare	63
7.4	Vizualizare drive	64

Lista tabelelor

3.1	Criterii evaluare sisteme similare	21
3.2	CloudMe Funcționalități	22
3.3	CloudMe Platforme disponibile și preț	22
3.4	Dropbox Funcționalități	24
3.5	Dropbox Platforme disponibile și preț	24
3.6	CrashPlan Funcționalități	26
3.7	CrashPlan Platforme disponibile și prețuri	26
3.8	ICloud Funcționalități	27
3.9	ICloud Platforme disponibile și prețuri	27
3.10	Google Drive Funcționalități	28
3.11	Google Drive Platforme Disponibile și Prețuri	28
3.12	OneDrive Funcționalități	29
3.13	OneDrive Platforme Disponibile și Prețuri	29
3.14	Funcționalități	30
3.15	Sisteme de operare	30
3.16	Funcționalități	31
3.17	Sisteme de operare	31
3.18	Comparație sisteme similare	32
5.1	Rata de compresie fișiere text	51
5.2	Rata de compresie PNG entropie mică	52
5.3	Rata de compresie PNG entropie mare	53
6.1	Rata de acoperire a testelor	58

Anexa B

Diagrame UML

Anexa C

Glosar

Termen	Definiție
HTTPS	<i>Hyper Text Transfer Protocol Secure</i> - este o versiune securizată a protocolului de transmisie HTTP.
gRPC	<i>Google Remote Procedure Call</i> - framework pentru comunicare între servicii, utilizând protocolul HTTP/2.
e2e gRPC HTML framework DOM RPC Protocol buffers	