**MINISTERUL EDUCAŢIEI ȘI CERCETĂRII**

**al REPUBLICII MOLDOVA**

**UNIVERSITATEA TEHNICĂ A MOLDOVEI**

**FACULTATEA CALCULATOARE, INFORMATICǍ**

**ȘI MICROELECTRONICǍ**

**Departamentul Informatică și Ingineria Sistemelor**

**RAPORT**

**Aplicatii Client-Server**

**Tema: Elaborarea unei aplicatii Client-Server: …………**

**Executant: / Nume Prenume, gtupa /**

**A verificat: / conducătorul /**

**Chișinău 2025**

CUPRINS

[INTRODUCERE 2](#_Toc184256106)

[1. ANALIZA PIETII ÎN DOMENIU 3](#_Toc184256107)

[1.1. Analiza proiectelor de pe piata în domeniu 3](#_Toc184256108)

[1.1.1.Analiza Aplicației Google Drive 3](#_Toc184256109)

[1.1.2.Analiza Aplicației Dropbox 5](#_Toc184256110)

[1.1.3.Analiza Aplicației MEGA 6](#_Toc184256111)

[1.2. Caracteristicile aplicatiilor 7](#_Toc184256112)

[1.3. Justificarea alegerii caracteristicile de baza a aplicatiei 8](#_Toc184256113)

[**1.4. Caracteristicile Aplicației Manager de Fișiere** 9](#_Toc184256114)

[CAPITOLUL 2 : ANALIZA TEHNOLOGIILOR UTILIZATE ÎN APLICATIE. 12](#_Toc184256115)

[2.1. Limbajul de programare Java 12](#_Toc184256116)

[2.2. Framework Spring Boot 13](#_Toc184256117)

[2.3. Protocolul HTTP (HyperText Transfer Protocol) 14](#_Toc184256118)

[2.4. Limbajele HTML, CSS și JavaScript 15](#_Toc184256119)

[2.5. Sistemul de gestionare a bazelor de date PostgreSQL 17](#_Toc184256120)

[2.6. Sistemul de gestionare a datelor Cloud 18](#_Toc184256121)

[CAPITOLUL 3 : CREAREA ALGORITMULUI SI PROGRAMULUI 19](#_Toc184256122)

[3.1. Descrierea schemei de functionare 19](#_Toc184256123)

[3.2. Crearea algoritmului programului 21](#_Toc184256124)

[3.3. Descrierea programului 26](#_Toc184256125)

[CONCLUZIE: 31](#_Toc184256126)

BIBLIOGRAFIE[: 32](#_Toc184256127)

# INTRODUCERE

În era digitală modernă, tehnologiile bazate pe arhitectura client-server joacă un rol crucial în dezvoltarea aplicațiilor distribuite. Acestea oferă o soluție eficientă pentru gestionarea resurselor și datelor utilizatorilor, promovând totodată accesibilitatea și securitatea informațiilor. Prin separarea clară a responsabilităților între client (interfața de utilizare și logica front-end) și server (logica de business și gestionarea datelor), acest model arhitectural asigură scalabilitatea și performanța necesare pentru a răspunde cerințelor unei lumi tot mai digitalizate.

În contextul actual, în care aproape toate informațiile sunt păstrate în format electronic, nevoia de spațiu de stocare devine indispensabilă pentru utilizatori. De la documente personale și fișiere multimedia până la date sensibile utilizate în mediul profesional, spațiul de stocare electronic este fundamentul funcționării noastre zilnice. Mai mult, gestionarea acestor fișiere într-un mod organizat și sigur este esențială, motiv pentru care un manager de fișiere bine conceput reprezintă o necesitate pentru toți utilizatorii, indiferent de domeniul de activitate.

Prezenta lucrare se axează pe proiectarea și implementarea unei aplicații client-server dedicate gestionării fișierelor, având ca obiectiv principal crearea unui mediu securizat, intuitiv și accesibil pentru utilizatorii care doresc să stocheze, partajeze și gestioneze documentele personale. Sistemul oferă posibilitatea fiecărui utilizator să își creeze un cont unic, unde fișierele pot fi organizate eficient și clasificate în funcție de preferințe. Aplicația permite încărcarea și descărcarea fișierelor și oferă utilizatorilor control detaliat asupra accesului: fișierele publice pot fi descărcate de oricine, în timp ce cele private sunt protejate și pot fi accesate doar cu permisiunea explicită a proprietarului.

De asemenea, flexibilitatea arhitecturii client-server facilitează integrarea unor tehnologii și instrumente moderne pentru dezvoltarea aplicațiilor. Tehnologiile utilizate în cadrul proiectului, precum Java pentru logica serverului, Spring Boot pentru optimizarea proceselor de configurare și MySQL pentru gestionarea eficientă a bazelor de date, exemplifică acest lucru.

Nu în ultimul rând, modelul client-server permite extinderea aplicației pentru a include funcționalități suplimentare, precum integrarea cu alte sisteme sau servicii cloud, automatizarea proceselor și analiza datelor. Astfel, aplicația propusă în această lucrare are potențialul de a evolua în timp, pentru a răspunde unor cerințe diverse și în continuă schimbare. Aceasta vine în întâmpinarea unei nevoi universale: aceea de a avea un spațiu de stocare fiabil și o metodă eficientă de a organiza și accesa datele în era digitală.

# 1. ANALIZA PIETII ÎN DOMENIU

## 1.1. Analiza proiectelor de pe piata în domeniu

Într-o piață dinamică, aplicațiile de stocare în cloud devin tot mai populare, fiind esențiale pentru utilizatorii care doresc să-și acceseze, să-și partajeze și să-și gestioneze fișierele online. Principalii competitori din acest domeniu oferă o gamă diversificată de funcționalități pentru a satisface nevoile de stocare, securitate și colaborare ale utilizatorilor.

### 1.1.1.Analiza Aplicației Google Drive



Figura 1.1 Logo-ul Google Drive

Google Drive a carui logo este prezentat in Figura 1.1 este unul dintre cele mai populare servicii de stocare în cloud, oferit de gigantul tehnologic Google. Acesta a devenit o platformă indispensabilă pentru utilizatorii care doresc să își păstreze fișierele accesibile oriunde și oricând, cu posibilitatea de a lucra în colaborare pe documente. Cu o integrare perfectă în ecosistemul Google și o gamă variată de funcționalități, Google Drive transformă modul în care utilizatorii își gestionează fișierele online. Mai jos este prezentata descrierea a cateva din caracteristicile de baza a aplicatiei:

**Interfața utilizatorului (UI)**

Google Drive are o interfață intuitivă și ușor de utilizat, optimizată pentru desktop și mobil. Utilizatorii pot organiza fișierele în foldere, pot folosi bara de căutare pentru a găsi rapid fișierele și pot vizualiza fișierele în formate variate (PDF, imagini, documente). Interfața permite previzualizarea fișierelor fără a le descărca.

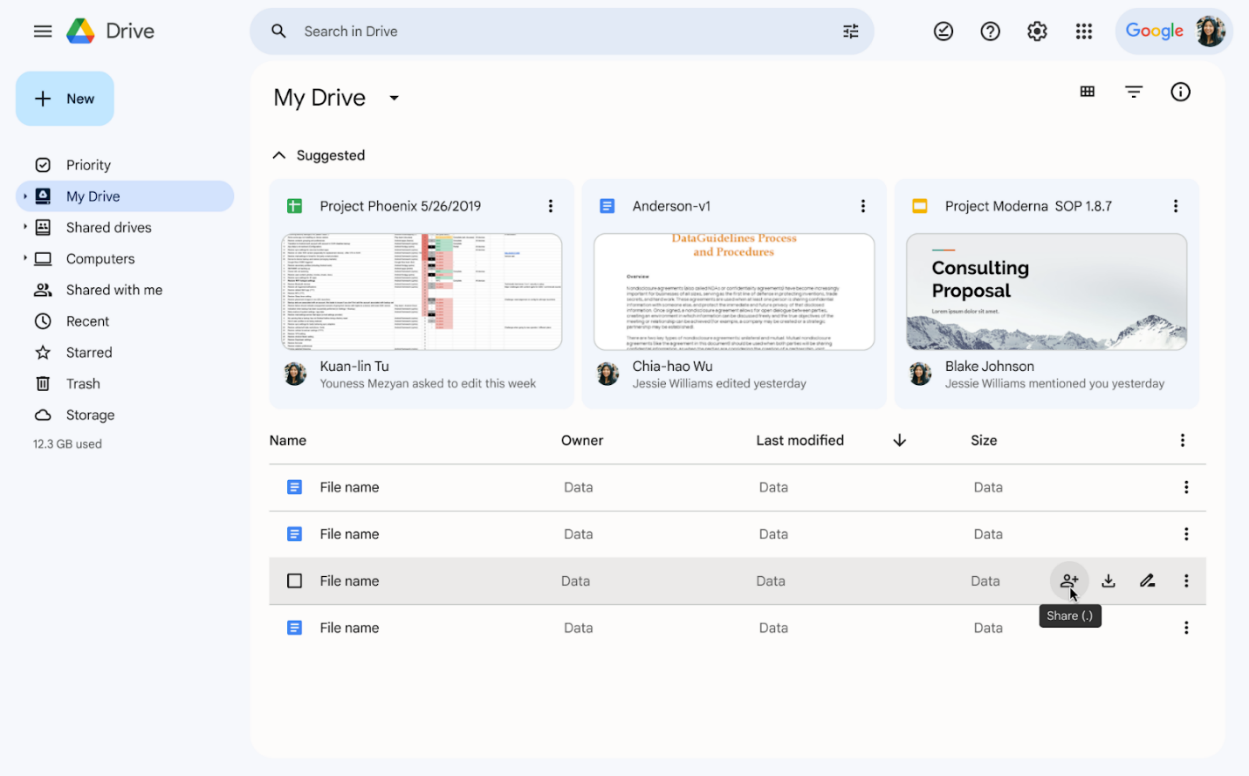


Figura 1.2 Interfața Google Drive

**Partea client:**

Partea client este disponibilă prin aplicații web și mobile, precum și printr-un program desktop prezentata in Figura 1.2 pentru sincronizarea fișierelor. Utilizatorii pot încărca, descărca și partaja fișiere direct din aplicație. Sincronizarea automată permite actualizarea în timp real a fișierelor de pe toate dispozitivele conectate.

**Partea server:**

Google Drive utilizează o infrastructură distribuită, găzduită pe serverele Google. Acest lucru asigură viteze mari de acces și disponibilitate constantă. Serverele sunt responsabile pentru gestionarea datelor, autentificarea utilizatorilor și oferirea unor funcții avansate, cum ar fi backup-ul și controlul versiunilor fișierelor.

**Logare și autentificare:**

Google Drive folosește sistemul de conturi Google pentru autentificare. Acesta oferă autentificare în doi pași (2FA) pentru securitate suplimentară. Logarea unică (SSO) simplifică accesul la toate aplicațiile Google.

**Securitate:**

Fișierele sunt criptate atât în tranzit (TLS), cât și în repaus (AES-256). Google Drive include și opțiuni de partajare securizată prin linkuri, unde utilizatorii pot seta permisiuni pentru vizualizare, comentare sau editare.[[1]](https://www.techradar.com/reviews/google-drive)

### 1.1.2.Analiza Aplicației Dropbox



Figura 1.3 Logo-ul Dropbox

*Dropbox* a carui figura este prezentata in Figura 1.3 este unul dintre pionierii stocării în cloud și continuă să fie o soluție preferată pentru partajarea și sincronizarea fișierelor. Recunoscut pentru simplitatea utilizării și fiabilitatea sa, Dropbox permite utilizatorilor să acceseze și să gestioneze fișiere de pe mai multe dispozitive fără probleme. Cu accent pe productivitate și colaborare, Dropbox rămâne o alegere populară atât pentru utilizatori individuali, cât și pentru echipe. Mai jos este prezentata descrierea a cateva din caracteristicile de baza a aplicatiei:

**Interfața utilizatorului (UI):** Dropbox are o interfață simplă, minimalistă, concepută

pentru utilizare rapidă. Bara de căutare este eficientă, iar fișierele pot fi organizate ușor în foldere. Aplicația oferă previzualizări pentru majoritatea tipurilor de fișiere și un panou lateral care afișează activitățile recente.

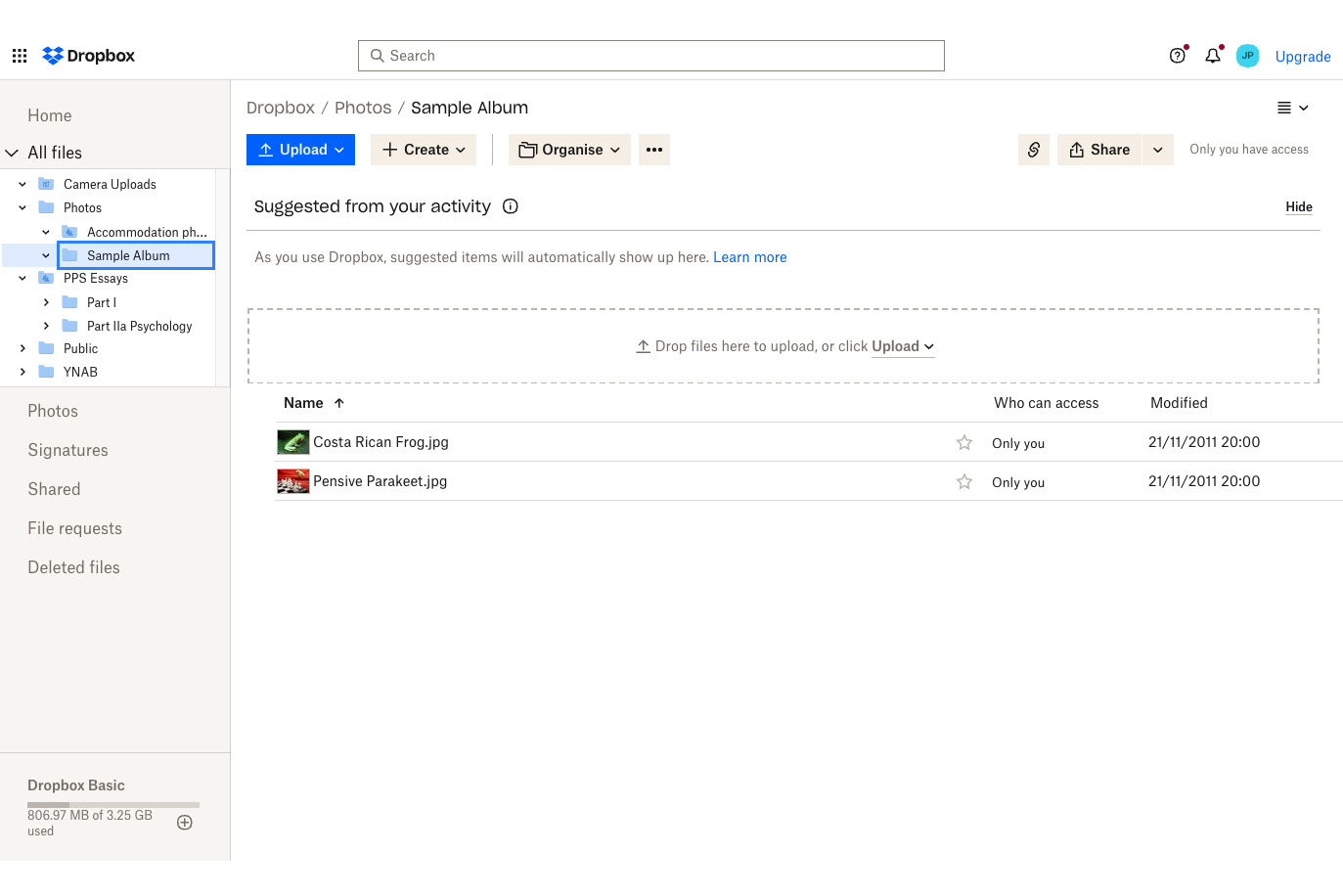


Figura 1.4 Interfața Dropbox

**Partea client:** Dropbox poate fi accesat prin aplicații web, mobile și desktop putem observa in Figura 1.4. Utilizatorii pot sincroniza fișierele automat între dispozitive și pot partaja fișiere rapid prin generarea de linkuri. Aplicația desktop oferă opțiuni de sincronizare selectivă, care economisesc spațiu pe dispozitivele utilizatorilor.

**Partea server:** serverele Dropbox gestionează stocarea și sincronizarea fișierelor utilizatorilor. Acestea folosesc tehnologia de deduplicare, care reduce utilizarea spațiului prin eliminarea fișierelor duplicate. Sistemele lor sunt scalabile, asigurând performanță chiar și pentru utilizatorii cu volume mari de date.

**Logare și autentificare:** Utilizatorii pot accesa Dropbox prin conturi proprii sau prin autentificare cu conturi Google sau Apple. Sistemul suportă autentificarea în doi pași pentru o securitate sporită.

**Securitate:** Dropbox criptează fișierele utilizatorilor în tranzit (TLS) și în repaus (AES-256). Pentru fișierele partajate, utilizatorii pot seta permisiuni și pot proteja linkurile cu parole sau date de expirare.[[2]](https://www.pcmag.com/reviews/dropbox)

### 1.1.3.Analiza Aplicației MEGA

****

Figura 1.5 Logo-ul MEGA

MEGA a carui logo este prezentat in Figura 1.5 este un serviciu de stocare în cloud care pune securitatea utilizatorilor pe primul loc. Cu un sistem de criptare end-to-end unic și un spațiu generos oferit gratuit, MEGA atrage utilizatorii care pun preț pe confidențialitate și siguranța datelor lor. Platforma se distinge prin funcționalități avansate de partajare și un control complet al accesului asupra fișierelor, făcând-o o opțiune ideală pentru utilizatorii preocupați de protecția informațiilor personale.

**Interfața utilizatorului (UI):** interfața MEGA este modernă și intuitivă, cu o temă albă și roșie care o diferențiază de alte servicii. Utilizatorii pot organiza fișierele în foldere și pot folosi funcția de căutare pentru a găsi rapid conținut. Aplicația oferă detalii avansate despre fiecare fișier, cum ar fi dimensiunea, locația și permisiunile.

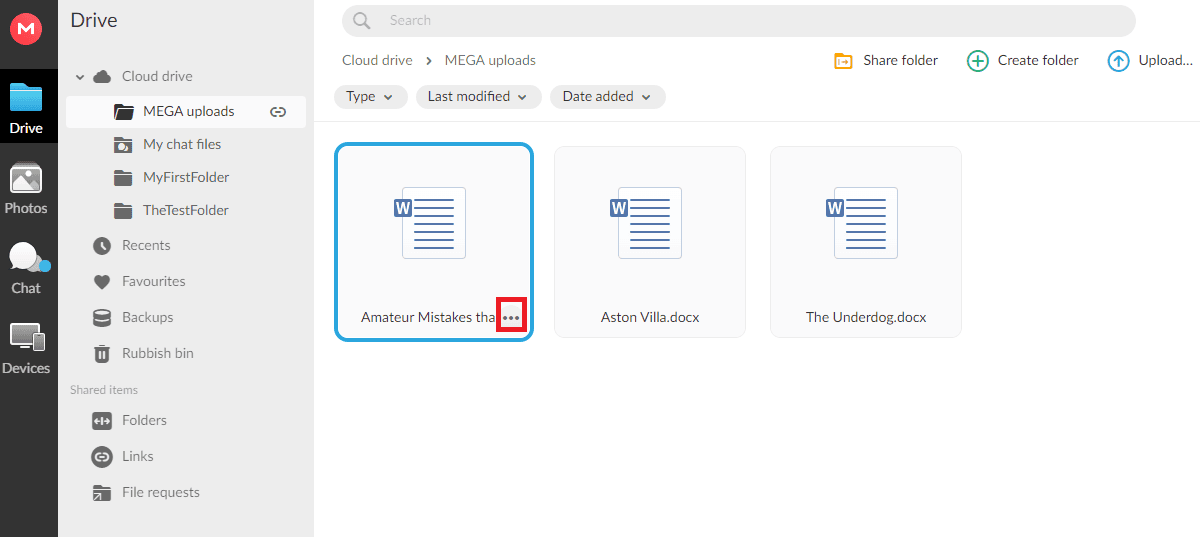


Figura 1.6 Interfața MEGA

**Partea client:** MEGA oferă aplicații pentru web, desktop și mobil ce este observat in Figura 1.6. Sincronizarea se face rapid, iar aplicațiile mobile permit încărcarea automată a fotografiilor și videoclipurilor. MEGA Chat, un serviciu suplimentar, permite comunicarea criptată între utilizatori.

**Partea server:** serverele MEGA sunt distribuite global și sunt optimizate pentru stocare securizată. Toate datele sunt criptate local pe dispozitivele utilizatorilor înainte de a fi încărcate pe servere, oferind o protecție avansată împotriva accesului neautorizat.

**Logare și autentificare:** sistemul de autentificare al MEGA este simplu, iar utilizatorii își pot proteja conturile prin criptare suplimentară. Nu oferă autentificare în doi pași în mod implicit, dar utilizatorii pot configura măsuri suplimentare de securitate manual.

**Securitate:**MEGA pune un accent puternic pe securitate. Criptarea end-to-end protejează datele atât în tranzit, cât și în repaus. Chiar și MEGA nu poate accesa datele utilizatorilor, deoarece cheile de criptare sunt controlate de aceștia.[[3]](https://www.itpro.com/cloud/cloud-storage/367993/mega-cloud-storage-review)

## 1.2. Caracteristicile aplicatiilor

Aplicațiile de stocare în cloud, cum sunt Google Drive, Dropbox și MEGA, au devenit esențiale pentru utilizatorii care doresc să își păstreze fișierele în siguranță și accesibile de oriunde. Deși fiecare platformă aduce elemente unice și puncte forte distincte, acestea au multe caracteristici comune care le fac soluții viabile pentru stocarea și partajarea fișierelor.

Tabelul de mai jos prezintă o comparație detaliată a acestor caracteristici comune și diferențe specifice între cele trei platforme de stocare, oferind o imagine clară a funcționalităților disponibile pentru fiecare dintre ele ce le putem vedea in Figura 1.7.

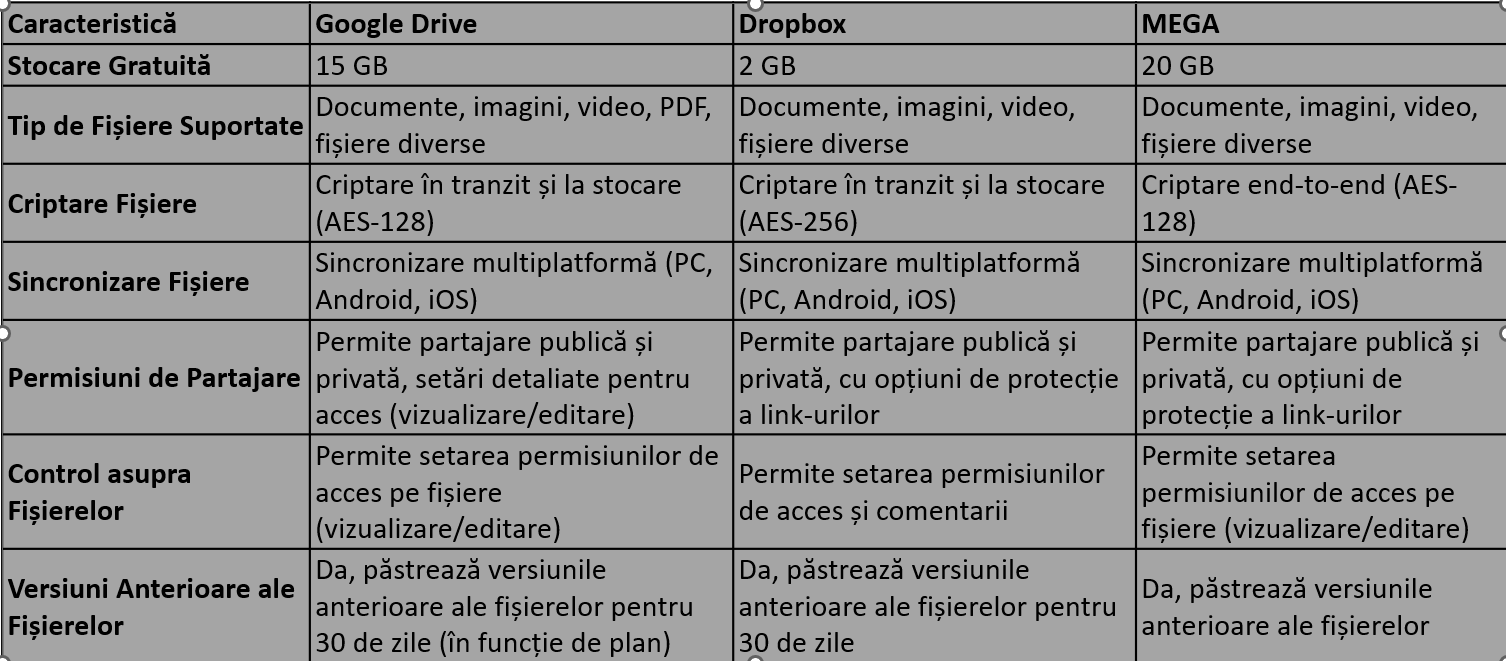


Figura 1.7 Caracteristicile Aplicatiilor Analizate

## 1.3. Justificarea alegerii caracteristicile de baza a aplicatiei

In urma unei analize asupra pieții am determinat cateva carecteristici cheie ce trebuie sa le includă aplicația mea de Management a Fișierelor.

**Gestionarea Conturilor Utilizatorilor :** Aplicația va permite fiecărui utilizator să își creeze un cont personalizat, prin care va avea acces la funcționalitățile aplicației. Procesul de autentificare este esențial pentru a asigura securitatea datelor și pentru a permite diferențierea fișierelor personale ale utilizatorului de cele ale altor utilizatori. În acest fel, fiecare cont devine un spațiu sigur pentru stocarea și gestionarea datelor personale.

**Partajarea Fișierelor cu Permisiuni** : O caracteristică cheie a aplicației este partajarea fișierelor. Utilizatorii vor putea seta fișierele ca fiind fie publice, fie private. Fișierele publice vor putea fi descărcate de oricine, în timp ce fișierele private vor necesita permisiunea explicită a proprietarului pentru a fi accesate. Aceasta oferă un control avansat asupra datelor și permite o colaborare sigură, asigurându-se că doar persoanele autorizate au acces la fișierele private.

**Interfață Ușor de Folosit :** pentru a face aplicația accesibilă unui public larg, interfața va fi simplă și intuitivă. Utilizatorii vor putea încarca, organiza și partaja fișiere cu ușurință, având opțiuni clare pentru gestionarea contului și a setărilor de partajare. Această abordare minimală în design ajută la reducerea timpului de învățare pentru utilizatori și asigură o experiență plăcută indiferent de nivelul de competență digitală al acestora.

**Caracteristici Îmbunătățite față de Competiție**

**Extinderea Spațiului Gratuit : s**pre deosebire de limitele de stocare gratuită oferite de aplicații precum Google Drive (15 GB) și Dropbox (2 GB), aplicația mea va oferi utilizatorilor posibilitatea de a accesa mai mult spațiu gratuit, atrăgând astfel utilizatorii care au nevoie de spațiu suplimentar fără costuri.

**Securitate Avansată :** pe lângă criptarea datelor, aplicația va implementa un sistem de securitate adaptiv .Acest aspect îmbunătățit va oferi un plus de siguranță comparativ cu MEGA, care deja pune accent pe securitatea datelor.

**Interfață Utilizator (UI) și Experiență Utilizator (UX) Optimizate :** aplicația va prioritiza o interfață simplă și intuitivă, cu un design modern și accesibil. Spre deosebire de alte aplicații care pot deveni aglomerate în ceea ce privește funcționalitățile și opțiunile, se va pune accent pe accesibilitate și ușurință de navigare, asigurând o experiență plăcută pentru toate categoriile de utilizatori.

**Păstrarea Fișierelor mai Mult Timp după Ștergere : s**pre deosebire de Google Drive, care păstrează fișierele șterse pentru 30 de zile, aplicația propusă va extinde această perioadă la o durată mai mare, permițând utilizatorilor să recupereze fișierele șterse accidental pentru o perioadă extinsă. Aceasta oferă o flexibilitate suplimentară și un grad ridicat de siguranță în gestionarea datelor.

**1.4. Caracteristicile Aplicației Manager de Fișiere**

În urma analizei pieței, au fost identificate mai multe nevoi ale utilizatorilor, care stau la baza unui manager de fișiere de succes. Studiul s-a axat pe aplicații populare precum Dropbox, Google Drive, MEGA și OneDrive, fiecare oferind un set distinct de caracteristici

Pe baza cercetării, au fost stabilite următoarele caracteristici de bază care vor fi implementate în aplicația "Manager de Fișiere":

* **Interfață intuitivă și prietenoasă**   
  Design-ul aplicației trebuie să fie simplu, dar atractiv vizual. Navigarea și accesul la funcționalități trebuie să fie evidente pentru utilizatori, indiferent de nivelul lor de expertiză tehnică
* **Funcționalități de înregistrare și logare**   
  Utilizatorii vor putea să își creeze conturi unice sau să se autentifice pentru a accesa funcțiile aplicației. Sistemul va include validarea informațiilor și recuperarea parolelor în caz de nevoie.
* **Sistem de gestionare a conturilor**   
  După autentificare, utilizatorul va accesa un cont personalizat, în care vor fi afișate toate directoarele și fișierele încărcate anterior. Fiecare fișier va fi listat împreună cu următoarele detalii esențiale:
* *Denumirea fișierului:* Numele atribuit de utilizator în momentul încărcării.
* *Data încărcării:* Data și ora exactă la care fișierul a fost urcat pe server.
* *Starea fișierului:* Informații despre permisiunile acestuia, indicând dacă fișierul este *public* (accesibil oricui) sau *privat* (vizibil doar pentru utilizator sau persoanele autorizate).

**Structurarea contului utilizatorului**

*Directoare personalizabile:* Utilizatorii vor putea crea foldere și subfoldere pentru a organiza fișierele în funcție de categorii sau preferințe.

*Filtrare și sortare:* Fișierele și directoarele vor putea fi sortate după criterii precum denumire, dată de încărcare

*Funcții dinamice:* Utilizatorul va putea modifica starea unui fișier (de la public la privat și invers) direct din interfața contului, printr-un simplu click.

* **Organizarea fișierelor**   
  Utilizatorii vor putea crea foldere și subfoldere, clasifica fișierele pe categorii. Funcționalitatea de căutare avansată bazată pe cuvinte-cheie va permite găsirea rapidă a fișierelor.
* **Sistem de gestionare a fișierelor private și publice**   
  Utilizatorii pot decide ce fișiere vor fi publice și disponibile pentru descărcare și care vor rămâne private, necesită aprobare pentru acces.
* **Profil public pentru utilizatori**   
  Aplicația va oferi fiecărui utilizator posibilitatea de a avea un **profil public**, care poate fi accesat de alți utilizatori prin intermediul unei funcții de căutare bazate pe nickname. Această funcționalitate va include următoarele aspecte:

*Vizualizarea fișierelor publice:* Odată ce un utilizator accesează profilul altuia, acesta va putea vedea toate fișierele marcate ca **publice** și disponibile pentru descărcare.

*Fișiere private accesibile:* Dacă utilizatorul vizitator are permisiunea explicită de la proprietar, acesta va putea vedea și descărca fișierele private pentru care a primit acces.

* **Gestionarea spațiului de stocare**   
  Un dashboard care arată spațiul utilizat și disponibil
* **Păstrarea fișierelor șterse**  
  Fișierele șterse vor fi păstrate temporar într-un folder de tip „Coș de gunoi” (Recycle Bin) pentru o perioadă definită, oferind utilizatorilor posibilitatea de a le restaura înainte de eliminarea definitivă.

# 2. ANALIZA TEHNOLOGIILOR UTILIZATE ÎN APLICATIE.

În era digitală, dezvoltarea aplicațiilor software implică utilizarea unui set divers de tehnologii și instrumente pentru a răspunde cerințelor funcționale, performanței și securității. Alegerea tehnologiilor adecvate este esențială pentru asigurarea unui sistem scalabil, robust și ușor de întreținut. Acest capitol se concentrează pe analiza tehnologiilor care vor fi utilizate pentru dezvoltarea aplicației . Pentru atingerea obiectivelor aplicației, am selectat următoarele tehnologii:

## 2.1. Limbajul de programare Java



Figura 2.1 Logo-ul Java

**Descriere generală:**

Java (Figura 2.1) este un limbaj de programare orientat pe obiecte, dezvoltat de Sun Microsystems (acum parte din Oracle) în 1995. Este unul dintre cele mai utilizate limbaje datorită caracteristicilor sale precum portabilitatea („write once, run anywhere”), securitatea, și suportul extins pentru aplicații enterprise. Java este independent de platformă, ceea ce înseamnă că poate rula pe orice dispozitiv care are o mașină virtuală Java (JVM).[[4]](https://en.wikipedia.org/wiki/Java_(programming_language))

**Avantaje:**

* **Portabilitate:** Codul Java compilat (bytecode) poate rula pe orice platformă care are JVM.
* **Securitate ridicată:** Oferă funcționalități integrate precum managerul de securitate și sistemul de permisiuni.
* **Biblioteci standard extinse:** Asigură suport pentru rețele, colecții de date, I/O și multe altele.
* **Performanță bună:** În ciuda interpretării bytecode-ului, optimizările JVM asigură performanță aproape de limbajele compilate.[[5]](https://www.takeo.ai/insights/why-java-is-still-relevant),[[6]](https://www.javatpoint.com/java-tutorial)

**Utilizare în proiect:**

Java va fi folosit pentru dezvoltarea logicii aplicației, atât pe partea de server (pentru procesarea cererilor și gestionarea fișierelor), cât și pentru implementarea comunicării între client și server, utilizând protocolul TCP.

## 2.2. Framework Spring Boot



Figura 2.2 Logo-ul Spring Boot

**Descriere generală:**

Spring Boot (Figura 2.2) este un framework open-source bazat pe Java, care simplifică dezvoltarea aplicațiilor enterprise și web. Este parte din ecosistemul Spring și permite configurarea rapidă și ușoară a aplicațiilor fără a necesita setări complexe.

**Caracteristici cheie:**

* **Configurare automată:** Reduce timpul necesar pentru configurarea aplicației prin detectarea automată a componentelor și setărilor.
* **Suport pentru REST:** Oferă un framework integrat pentru crearea serviciilor web RESTful, utilizat frecvent în aplicațiile client-server.
* **Integrare ușoară cu baze de date:** Asigură conectivitate facilă cu MySQL și alte baze de date relaționale prin intermediul JPA (Java Persistence API).

**Utilizare în proiect:**

Spring Boot va fi utilizat pentru partea de server a aplicației, unde va gestiona cererile HTTP, va implementa autentificarea utilizatorilor și va oferi acces la resurse precum fișierele stocate.[[8]](https://slashdev.io/blog/building-fast-backend-apis-in-spring-boot-in-2024)

## 2.3. Protocolul HTTP (HyperText Transfer Protocol)

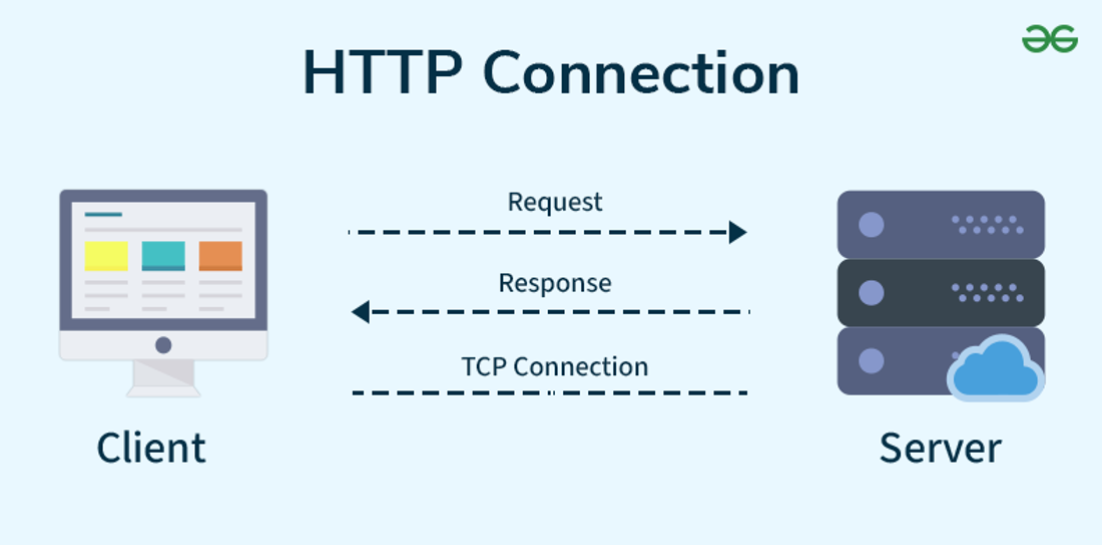
****

Figura 2.3 Protocolul TCP

**Descriere generală:**

HTTP (Figura 2.3) este un protocol de comunicare utilizat în rețelele de tip client-server pentru transferul datelor, în special în contextul web-ului. Este un protocol fără stare (stateless), ceea ce înseamnă că fiecare cerere și răspuns sunt independente. HTTP funcționează de obicei peste protocolul de transport TCP, asigurând o transmisie fiabilă a datelor între client și server.

**Avantaje:**

* **Simplitate:** HTTP este ușor de utilizat și implementat, permițând trimiterea rapidă a cererilor și primirea răspunsurilor.
* **Interoperabilitate:** Este compatibil cu toate browserele și serverele web.
* **Flexibilitate:** Suportă diverse tipuri de date (HTML, JSON, imagini, fișiere audio-video) și metode (GET, POST, PUT, DELETE).

**Utilizare în proiect:**

HTTP va fi utilizat în următoarele scopuri:

* Comunicare client-server: Trimiterea cererilor (autentificare, încărcare fișiere) de la aplicația client către server și primirea răspunsurilor.
* Transfer date: Gestionarea traficului între aplicație și backend, incluzând datele utilizatorilor și metadatele fișierelor.
* Interfață API: Protocolul va servi ca bază pentru conectarea aplicației locale cu serviciile externe (cum ar fi API-ul OAuth de la Google)

## 2.4. Limbajele HTML, CSS și JavaScript

HTML, CSS și JavaScript reprezintă fundamentul tehnologiilor web moderne, formând împreună baza pentru dezvoltarea interfețelor utilizator dinamice și interactive. Acest trio tehnologic este indispensabil în dezvoltarea oricărei aplicații web, oferind o combinație de structură, estetică și interactivitate care creează experiențe fluide și intuitive pentru utilizatori.

**HTML (HyperText Markup Language):**



Figura 2.4 Logo-ul HTML

HTML (Figura 2.4) este limbajul standard pentru crearea structurilor de bază ale paginilor web. Definește elementele unei interfețe grafice, cum ar fi butoanele, câmpurile de text și tabelele.

**CSS (Cascading Style Sheets):**



Figura 2.5 Logo-ul CSS

CSS (Figura 2.5)este utilizat pentru designul paginilor web, controlând aspectul elementelor HTML, cum ar fi culorile, fonturile, marginile și poziționarea. Oferă un aspect estetic și consistent aplicației.

**JavaScript:**

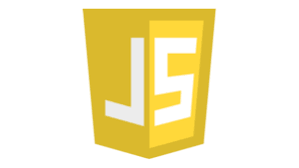


Figura 2.6 Logo-ul JavaScript

JavaScript (Figura 2.6) este un limbaj de scripting folosit pentru a crea funcționalități dinamice și interactive. Este executat direct în browser și este ideal pentru manipularea elementelor HTML și CSS, precum și pentru realizarea de apeluri către server (de exemplu, prin AJAX).

**Avantaje comune:**

* **HTML:** Simplu și intuitiv de utilizat.
* **CSS:** Separă designul de conținut, făcând codul mai ușor de întreținut.
* **JavaScript:** Interactivitate și compatibilitate cu toate browserele.

**Utilizare în proiect:**

Aceste tehnologii vor fi folosite pentru dezvoltarea interfeței aplicației. HTML va defini structura paginii, CSS va asigura stilizarea, iar JavaScript va adăuga funcționalități dinamice, precum actualizarea listei de fișiere și gestionarea cererilor de permisiune.[[9]](https://www.freecodecamp.org/news/html-css-and-javascript-explained-for-beginners/)

## 2.5. Sistemul de gestionare a bazelor de date PostgreSQL

****

Figura 2.7 Logo-ul PostgreSQL

**Descriere generală:**

**PostgreSQL** (Figura 2.5) este un sistem avansat de baze de date relaționale, open-source, proiectat pentru a oferi conformitate completă cu standardele SQL. Este apreciat pentru extensibilitatea sa, fiabilitate, și capabilitatea de a gestiona date complexe și mari.

**Caracteristici cheie:**

* Extensibilitate ridicată: Suportă tipuri de date personalizate, funcții definite de utilizator, și extensii.
* Conformitate SQL: Este foarte aproape de standardul SQL:2011, permițând o implementare riguroasă a normelor SQL.
* Suport pentru date complexe: Poate lucra cu date JSON, XML, și alte formate nestructurate, facilitând integrarea cu aplicații moderne.
* Fiabilitate: Oferă mecanisme puternice de recuperare a datelor (recovery) și protecție împotriva erorilor.
* Performanță: Poate gestiona volume mari de date și suportă paralelism pentru interogări rapide.
* Securitate avansată: Suportă autentificare avansată, criptare și politici de acces bazate pe roluri.
* Integrare multiplatformă: Se conectează ușor cu aplicații scrise în diverse limbaje (Java, Python, PHP) folosind drivere compatibile, precum JDBC.

## 2.6. Sistemul de gestionare a datelor Cloud



Figura 2.8 Logo-ul Firebase

**Descriere generală:**   
Firebase prezenta in figura 2.8 este o platformă oferită de Google care facilitează dezvoltarea de aplicații web și mobile. Este cunoscută pentru funcționalitățile sale "backend-as-a-service" (BaaS), eliminând nevoia de a configura și administra servere complexe. Firebase oferă soluții scalabile și integrate pentru autentificare, baze de date în timp real, stocare de fișiere și funcții de notificare.

**Avantaje:**

* **Integrare nativă cu Google:** Compatibilitate perfectă cu alte servicii Google și suport complet pentru autentificarea prin conturi Google.
* **Bază de date în timp real:** Permite sincronizarea datelor între clienți în timp real, asigurând o experiență fluidă pentru utilizatori.
* **Stocare în cloud:** Oferă un sistem de stocare securizat pentru fișiere, imagini și date mari, cu acces rapid prin API-uri.
* **Scalabilitate automată:** Resursele sunt ajustate automat în funcție de cerințele aplicației, eliminând riscul de supraîncărcare.
* **Securitate avansată:** Integrarea cu protocoale de securitate, precum HTTPS, și gestionarea accesului prin reguli personalizabile.

**Utilizare în proiect:**

Firebase este utilizat pentru:

* **Stocare fișiere:** Încărcarea și descărcarea fișierelor utilizând serviciul de stocare în cloud oferit de Firebase.

# 3 . CREAREA ALGORITMULUI SI PROGRAMULUI

## 3.1. Descrierea schemei de functionare

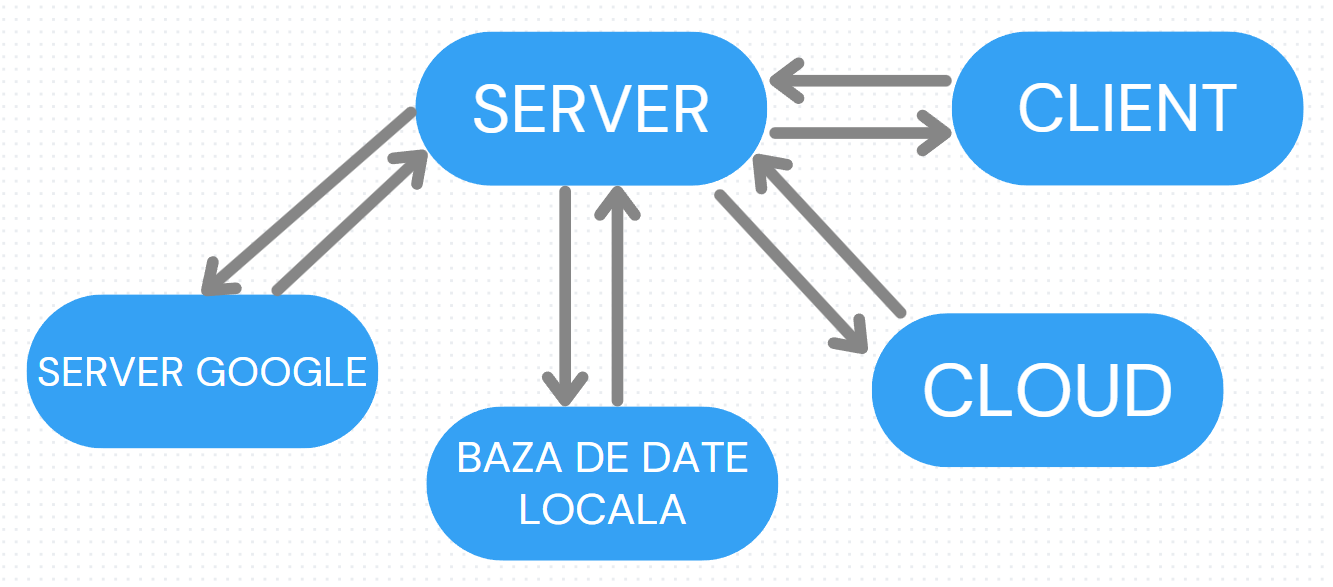


Figura 3.1 Schema de functionare

Schema din figura 3.1 reprezintă fluxul de informații în aplicație.Aceasta evidențiază conexiunile și interacțiunile dintre componentele principale: clientul (utilizatorul final), serverul aplicației, serverul Google pentru autentificare, baza de date locală și serviciile de stocare în cloud.

**Fluxul complet al aplicației și rolurile fiecărui component**

**Client**

* **Rol:** Interfața utilizatorului, responsabilă pentru inițierea interacțiunii cu aplicația**.**
* **Fluxul:**
  1. Clientul inițiază o cerere către aplicație (ex. autentificare, upload fișiere, descărcare).
  2. Primește răspunsuri procesate de la server și afișează rezultatele.
* **Acțiuni posibile:**
  1. Trimiterea cererilor HTTP către server pentru autentificare, upload/download fișiere.
  2. Afișarea mesajelor de eroare sau succes bazate pe răspunsurile aplicației**.**

**Server**

* **Rol:** Componente centrală, gestionează logica aplicației și interacționează cu alte sisteme (Google, baza de date, cloud).
* **Fluxul:**
  1. Primește cereri HTTP de la client.
  2. Redirecționează utilizatorul către Serverul Google pentru autentificare.
  3. Validează token-ul primit de la Google și accesează baza de date locală pentru informații adiționale despre utilizator.
  4. Dacă cererea implică manipularea fișierelor, trimite cererile de upload/download către Cloud.
  5. Răspunde clientului cu rezultatele procesării (fișiere, mesaje, etc.).
* **Acțiuni posibile:**
  1. Validarea autentificării prin Google OAuth2.
  2. Accesarea bazei de date locale pentru metadate.
  3. Transferul fișierelor către sau din cloud.
  4. Procesarea și răspunsul la cererile HTTP primite de la client.

**Server Google**

* **Rol:** Sistem de autentificare prin OAuth2 utilizat pentru validarea identității utilizatorilor.
* **Fluxul:**
  1. Primește cereri de autentificare redirecționate de serverul aplicației.
  2. Solicită utilizatorului credențiale pentru contul Google.
  3. Generează un token de acces (care confirmă autentificarea).
  4. Returnează token-ul serverului aplicației.
* **Acțiuni posibile:**
  1. Procesarea autentificării utilizatorului.
  2. Generarea token-ului sigur pentru acces.

**Baza de date locală**

* **Rol**: Stochează informații despre utilizatori și metadate despre fișiere.
* **Fluxul:**
  1. Serverul aplicației salvează sau preia date din baza de date în funcție de cereri.
  2. Datele sunt returnate către server pentru răspunsurile clientului.
* **Acțiuni posibile:**
  1. Salvarea metadatelor fișierelor (ex. nume, dimensiune, locație).
  2. Stocarea datelor utilizatorilor (ex. ultima autentificare).
  3. Actualizarea și ștergerea datelor la cererea aplicației.

**Cloud**

* **Rol:** Sistem de stocare responsabil pentru gestionarea fișierelor utilizatorilor.
* **Fluxul:**
  1. Primește fișiere de la serverul aplicației pentru upload.
  2. Returnează fișierele stocate la cererea serverului (descărcare).
  3. Șterge fișierele în funcție de cererile serverului aplicației.
* **Acțiuni posibile:**
  1. Stocarea și gestionarea fișierelor utilizatorilor.
  2. Furnizarea accesului la fișiere prin download.
  3. Eliminarea fișierelor vechi sau neutilizate.

## 3.2. Crearea algoritmului programului

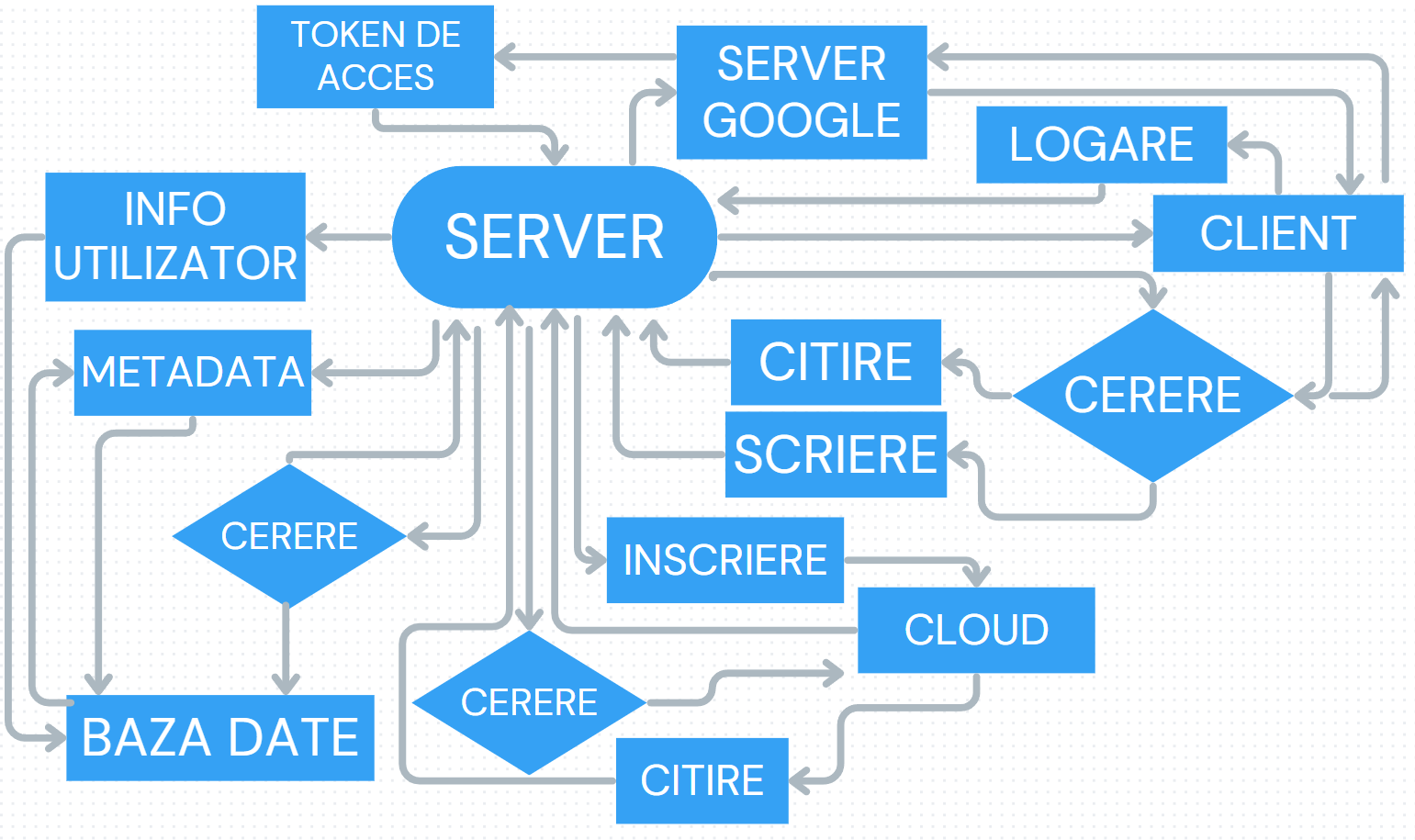


Figura 3.2 Schema generala a aplicatiei

Figura 3.2 este schema generală a aplicației, care ilustrează interacțiunile dintre diferitele componente și fluxul principal al datelor. Voi explica principalele elemente și conexiuni:

**CLIENT**

* Reprezintă utilizatorul final care interacționează cu aplicația printr-o interfață grafică sau API.
* **Logare:** Clientul inițiază procesul de autentificare, trimițând cereri către server pentru a obține acces la datele sale.

**SERVER**

* Este componenta centrală care orchestrează toate operațiunile aplicației.
* Preia cererile de la client, validează accesul, și coordonează operațiunile de citire și scriere asupra bazei de date și a cloud-ului.
* Este responsabil pentru obținerea **token-urilor de acces** utilizate pentru operațiuni securizate cu alte servicii (ex. Google).

**Bază de date**

* Stochează informațiile utilizatorului (date personale, setări, permisiuni).
* Include și **metadata** despre fișiere (dimensiuni, tipuri, timestamp-uri etc.).
* Cererile către baza de date includ:
  + Actualizarea informațiilor utilizatorului.
  + Interogări legate de fișiere (nume, tipuri, marcaje).

**Cloud (ex. Google Storage)**

* Este locația efectivă de stocare a fișierelor încărcate de utilizatori.
* Operațiile efectuate includ:
  + **Inscriere:** Stocarea fișierelor în cloud.
  + **Citire:** Descărcarea sau afișarea fișierelor la cererea utilizatorului.
* Serverul mediază accesul la cloud folosind token-uri și cereri securizate.

**Token de acces**

* Este necesar pentru autentificarea și autorizarea interacțiunilor cu servicii externe (cum ar fi Google Drive).
* Token-ul este generat sau preluat la logare și utilizat ulterior pentru cereri la nivel de cloud.

**Fluxul cererilor**

* **Cerere:** Orice acțiune a utilizatorului (cum ar fi încărcarea, descărcarea sau vizualizarea fișierelor) este preluată sub forma unei cereri de server.
* În funcție de tipul cererii:
  + Dacă implică date din baza de date, serverul face interogări sau actualizări acolo.
  + Dacă implică fișiere, serverul interacționează cu cloud-ul pentru a efectua operațiunile necesare.

**Citire/Scriere**

* **Citire:** Serverul extrage date din baza de date sau din cloud și le transmite utilizatorului.
* **Scriere:** Serverul inserează/actualizează date fie în baza de date, fie în cloud.

**Server Google**

* Reprezintă un furnizor de servicii externe care facilitează funcționalități suplimentare (ex. Google Drive).
* Serverul aplicației comunică cu serverul Google pentru sincronizarea fișierelor sau alte operațiuni.

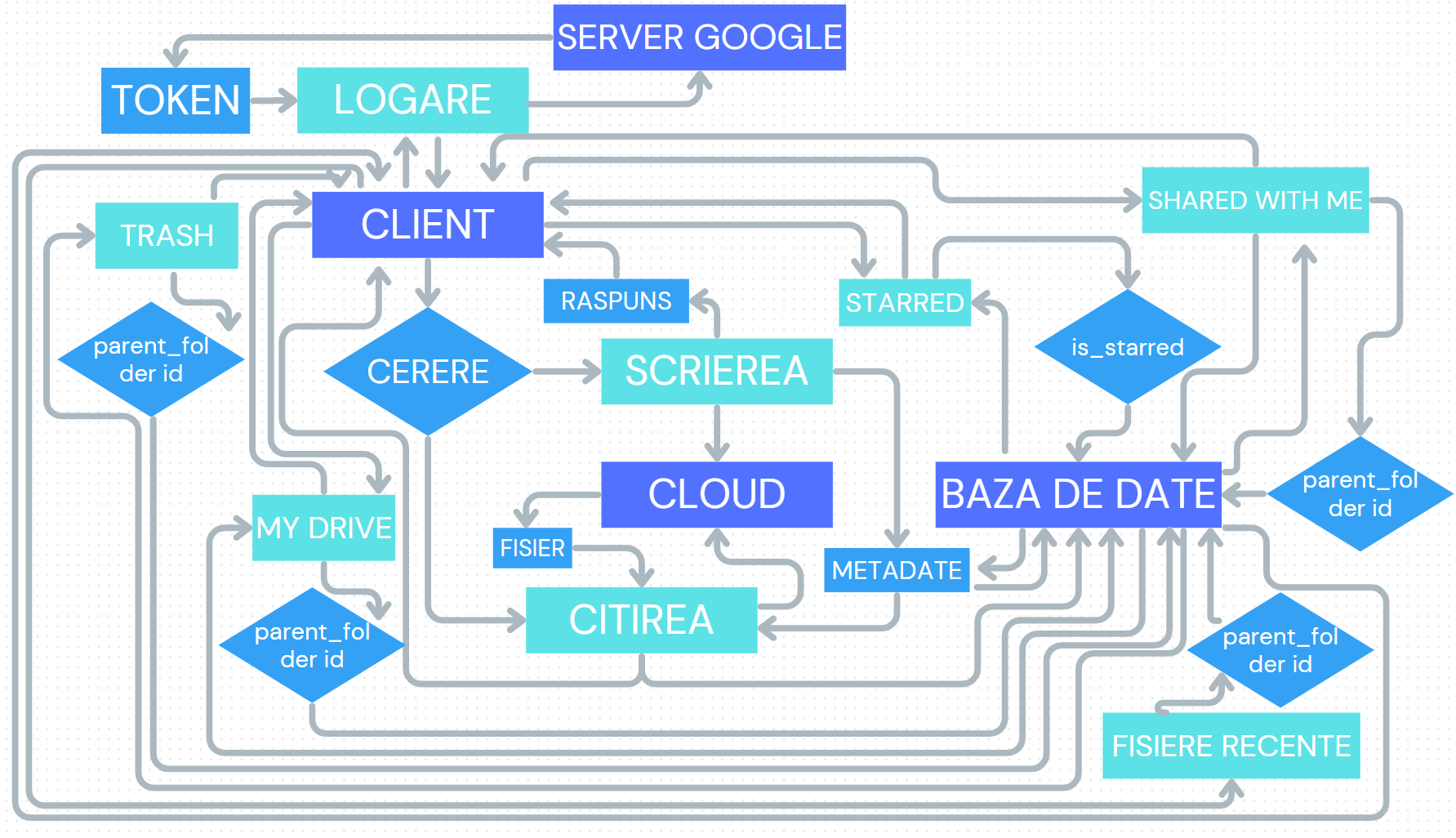


Figura 3.3 Schema detaliata a serverului

Schemă din figura 3.3 este o schemă ce detaliază fluxurile de date în cadrul aplicației, punând accent pe interacțiunea dintre server și componentele esențiale: logare, scriere, citire, managementul fișierelor și funcționalități precum **Trash**, **Starred**, **Shared with me**, și **My Drive**.

**Logare:**

* **Flux:** Client → Server → Server Google → Token → Server → Client.
* **Proces:**
  + Clientul trimite cererea de autentificare către server.
  + Serverul redirecționează cererea către Server Google pentru autentificare OAuth2.
  + După validarea identității, Server Google returnează un token de acces serverului aplicației.
  + Serverul utilizează acest token pentru a permite accesul clientului.

**Scriere:**

* **Flux:** Client → Server → Cloud → Server →Baza de Date (Metadata).
* **Proces:**
  + Clientul inițiază încărcarea unui fișier.
  + Serverul primește fișierul și îl stochează în Cloud.
  + După stocare, serverul salvează informațiile despre fișier (nume, locație, dimensiune) în Baza de date sub formă de metadate.
  + Clientul primește o confirmare că fișierul a fost încărcat cu succes.

**Citire:**

* **Flux:** Client → Server → Metadate (din baza de date) → Cloud (fișier) → Client.
* **Proces:**
  + Clientul solicită descărcarea unui fișier.
  + Serverul accesează metadatele fișierului pentru a găsi locația acestuia.
  + Fișierul este descărcat din Cloud și transmis clientului.

**Fișiere Recente:**

* **Flux:** Client → Server → Baza de Date → Client.
* **Proces:**
  + Serverul accesează baza de date pentru a căuta fișierele adăugate recent.
  + Datele despre aceste fișiere sunt trimise clientului pentru afișare**.**

**Trash:**

* **Flux**: Client → Server → Baza de Date → Client.
* **Proces:**
  + Când un fișier este șters, serverul actualizează locatia lui în baza de date.
  + Clientul poate vizualiza fișierele din Trash accesând datele fisierele din folderul trash.

**Starred:**

* **Flux: C**lient → Server → Baza de Date (is\_starred flag) → Client.
* **Proces:**
  + Fișierele marcate ca Starred sunt identificate în baza de date printr-un flag is\_starred = true.
  + Serverul returnează lista de fișiere marcate către client pentru afișare.

**Shared with me:**

* **Flux**: Client → Server → Baza de Date (shared files metadata) → Client.
* **Proces:**
  + Fișierele partajate cu utilizatorul sunt identificate în baza de date pe baza informațiilor de partajare (ex. ID-ul utilizatorului).
  + Serverul transmite clientului lista de fișiere partajate.

**My Drive:**

* **Flux:** Client → Server → Baza de Date → Client.
* **Proces:**
  + Serverul extrage din baza de date toate fișierele asociate utilizatorului care nu sunt marcate ca șterse sau partajate.
  + Datele sunt trimise către client pentru afișare în secțiunea My Drive.

## 3.3. Descrierea programului

Această aplicație client-server reprezintă o platformă de gestionare a fișierelor inspirată de serviciile de tip "cloud storage" (ex. Google Drive), oferind utilizatorilor o interfață prietenoasă și funcționalități avansate pentru a-și organiza și gestiona datele. Sistemul este structurat pe arhitectura client-server.

**Autentificarea utilizatorului (Logare și Autentificare)**

**Descriere funcționalitate:**   
Autentificarea se bazează pe protocolul OAuth2, utilizând contul Google. Utilizatorul este redirecționat către Google pentru a se autentifica, iar după validare, backend-ul primește un token de acces utilizat pentru accesarea resurselor. După autentificare, utilizatorul are acces la fișierele și folderele stocate.

**Flux tehnic:**

1. Clientul accesează resurse protejate, iar serverul îl redirecționează către Google pentru autentificare.
2. Google returnează un cod de autorizare, care este schimbat de backend cu un token de acces.
3. Token-ul este stocat și utilizat pentru accesul la resurse.

**Tehnologii utilizate:**

* **Spring Security OAuth2 Client**
  + Clase: SecurityConfig, OAuth2LoginSuccessHandler.
  + Metode: oauth2Login(), onAuthenticationSuccess().
* **Motive pentru utilizarea Spring Security:** Integrare rapidă cu provideri OAuth2, configurare flexibilă și scalabilitate.

**Încărcarea fișierelor**

**Descriere funcționalitate:**   
Utilizatorii pot încărca fișiere în aplicație. Acestea sunt salvate în Firebase Storage, iar metadatele lor (nume, dimensiune, link public) sunt stocate în PostgreSQL.

**Flux tehnic:**

1. Clientul trimite fișierul către server printr-o cerere POST.
2. Serverul validează fișierul și îl salvează în Firebase Storage.
3. Metadatele fișierului sunt salvate în baza de date PostgreSQL.

**Tehnologii utilizate:**

* **Firebase Admin SDK**
  + Clase: FirebaseConfig, FirebaseStorageService.
  + Metode: uploadFile(File file).
* **Spring Data JPA**
  + Clase: FileRepository, FileService.
  + Metode: save().

**De ce aceste tehnologii?**

* Firebase Storage este scalabil și permite generarea de link-uri publice securizate.
* PostgreSQL oferă o bază de date robustă pentru gestionarea metadatelor.

**Vizualizarea fișierelor**

**Descriere funcționalitate:**   
Aplicația permite utilizatorilor să vizualizeze fișierele încărcate, cu detalii precum numele, dimensiunea și un link de descărcare. Datele sunt extrase din PostgreSQL, iar link-urile sunt generate din Firebase.

**Flux tehnic:**

1. Serverul interoghează baza de date pentru fișierele utilizatorului.
2. Datele sunt transmise către client pentru afișare.

**Tehnologii utilizate:**

* **Spring Data JPA**
  + Clasă: FileRepository.
  + Metodă: findByUserId(Long userId).

**De ce aceste tehnologii?**

* Spring Data JPA simplifică accesul la baza de date.
* Thymeleaf permite afișarea datelor dinamice într-o interfață prietenoasă.

**Trash (Coșul de reciclare)**

**Descriere funcționalitate:**   
Fișierele șterse sunt mutate într-un folder virtual numit "Trash". Aceste fișiere pot fi restaurate sau șterse definitiv.

**Flux tehnic:**

1. Fișierul este actualizat în baza de date cu o marcă "Trash".
2. Poate fi restaurat sau eliminat definitiv la cerere.

**Tehnologii utilizate:**

* **Spring Boot**
  + Clasă: TrashController.
  + Metodă: moveFileToTrash(Long fileId).
* **PostgreSQL**
  + Structură dedicată pentru asocierea fișierelor cu folderul Trash.

**De ce această abordare?**   
Permite recuperarea fișierelor șterse accidental și optimizează experiența utilizatorului.

**Fișiere marcate (Starred)**

**Descriere funcționalitate:**Utilizatorii pot marca fișierele și folderele favorite, iar acestea vor fi evidențiate într-o secțiune dedicată a interfeței, accesibilă printr-un meniu separat.

**Flux tehnic:**

1. Serverul verifică atributele isStarred pentru fișiere și foldere.
2. Datele sunt filtrate și transmise către client pentru afișarea în interfață, în secțiunea *Starred*.

**Tehnologii utilizate:**

* **Spring MVC**
  + Clasă: StarredController
  + Metodă: starred(Model model, Principal principal) - preia folderele și fișierele marcate cu isStarred = true și le afișează utilizatorului**.**
* **Spring Data JPA**
  + Metodă: Nu se utilizează o metodă explicită pentru marcarea fișierelor, ci filtrele sunt aplicate la nivel de serviciu (prin iterare).
  + Datele sunt obținute și procesate folosind serviciile ParentFolderService și UserService.

**Partajarea fișierelor**

**Descriere funcționalitate:**   
Fișierele pot fi partajate cu alți utilizatori prin link-uri publice sau relații directe între utilizatori și fișiere.

**Flux tehnic:**

1. Serverul generează link-uri publice sau actualizează relațiile din baza de date.
2. Link-urile sunt trimise utilizatorilor pentru partajare.

**Tehnologii utilizate:**

* **Firebase Admin SDK**
  + Metodă: generatePublicLink(String fileName).
* **Spring Data JPA**
  + Relații many-to-many între utilizatori și fișiere.

**Căutarea fișierelor**

**Descriere funcționalitate:**Utilizatorii pot căuta fișiere după nume, cu suport pentru căutare insensibilă la majuscule. Căutările returnează toate fișierele ale căror nume conțin termenul introdus.

**Flux tehnic:**

1. Clientul trimite termenul de căutare către server printr-o cerere HTTP.
2. Serverul interoghează baza de date utilizând expresii LIKE în JPQL pentru a găsi fișierele care corespund termenului de căutare.
3. Serverul returnează lista fișierelor către client sub formă de răspuns JSON.

**Tehnologii utilizate:**

* **Spring Data JPA**
  + Clasă: FileRepository.
  + Metodă: findAllByFileNameContainingIgnoreCase(String search) - interogare definită pentru a căuta fișiere după nume fără a ține cont de majuscule.
* **Spring MVC**
  + Clasă: FileController.
  + Metodă: searchFiles(String search) - expune un endpoint HTTP pentru a prelua fișierele corespunzătoare.

**Descărcarea fișierelor**

**Descriere funcționalitate:**Fișierele pot fi descărcate prin link-uri securizate generate din Firebase Storage. Serverul generează link-uri temporare care permit accesul la fișierele dorite.

**Flux tehnic:**

1. Serverul generează un link temporar pentru descărcare folosind Firebase Admin SDK.
2. Link-ul generat este returnat utilizatorului pentru descărcare.

**Tehnologii utilizate:**

* **Firebase Admin SDK**
  + Metodă: generateDownloadLink(String fileName) - pentru generarea unui link securizat temporar către fișierul stocat în Firebase Storage.
* **Spring MVC**
  + Clasă: FileController.
  + Metodă: downloadFile(Long fileId) - responsabilă pentru apelarea serviciului care generează link-ul de descărcare.

**Arhitectura generală a aplicației**

Aplicația urmează o arhitectură stratificată:

1. **Stratul de control (Controller):** Primește cererile utilizatorului și apelează stratul de servicii.
2. **Stratul de servicii (Service):** Conține logica aplicației pentru funcționalitățile descrise.
3. **Stratul de acces la date (Repository):** Gestionează interacțiunile cu baza de date.

# CONCLUZIE

În concluzie, aplicația a fost dezvoltată cu succes, respectând toate obiectivele și cerințele stabilite la începutul proiectului. Funcționalitățile cheie implementate includ: Autentificare și Logare utilizatori, Încărcarea și Descărcarea fișierelor, Vizualizarea fișierelor, Căutarea fișierelor, Gestionarea Trash-ului, Fișiere Favorite și Partajarea fișierelor, toate integrate într-o arhitectură solidă și eficientă.

Pe parcursul dezvoltării aplicației, am avut ocazia să învăț și să aplic o gamă largă de tehnologii, ceea ce mi-a permis să îmi îmbunătățesc abilitățile de programare. Implementarea tehnologiilor precum Spring Boot, Java, PostgreSQL și integrarea cu Firebase m-au ajutat să dobândesc experiență practică valoroasă și să aprofundez cunoștințele tehnice necesare dezvoltării aplicațiilor moderne.

De asemenea, aplicația prezintă un potențial considerabil de dezvoltare pe termen lung. Planurile de extindere includ implementarea unui sistem de tarifare accesibil pentru utilizatori, care va permite extinderea spațiului de stocare, și o funcționalitate care va oferi utilizatorilor o vizualizare detaliată a resurselor ocupate din spațiul de stocare disponibil, facilitând astfel o gestionare mai eficientă a acestora.

Astfel, aplicația îndeplinește toate scopurile propuse inițial, iar cu noi îmbunătățiri și adăugiri planificate, aceasta are un mare potențial pentru creștere și îmbunătățire continuă.

## BIBLIOGRAFIE

Google Drive

1. <https://www.techradar.com/reviews/google-drive>

Acesat la data : 14.11.24

Dropbox

1. <https://www.pcmag.com/reviews/dropbox>

Acesat la data : 14.11.24

MEGA

1. <https://www.itpro.com/cloud/cloud-storage/367993/mega-cloud-storage-review>

Acesat la data : 14.11.24

Descrierea limbajului Java

1. <https://en.wikipedia.org/wiki/Java_(programming_language)>

Acesat la data : 21.11.24

Relevanța limbajului Java

1. <https://www.takeo.ai/insights/why-java-is-still-relevant>

Acesat la data : 21.11.24

Java tutorial

1. <https://www.javatpoint.com/java-tutorial>

Acesat la data : 21.11.24

TCP

1. <https://www.rfc-editor.org/rfc/rfc793>

Acesat la data : 21.11.24

Springboot

1. <https://slashdev.io/blog/building-fast-backend-apis-in-spring-boot-in-2024>

Acesat la data : 21.11.24

HTML,CSS,JavaScript

1. <https://www.freecodecamp.org/news/html-css-and-javascript-explained-for-beginners/>

Acesat la data : 21.11.24

MySQL

1. <https://www.postgresql.org/docs/>

Acesat la data : 21.11.24

Firebase

1. <https://firebase.google.com/docs>

Acesat la data : 22.11.24