

**Proiect la LP2**

**București, 2025**



**Analizator de loguri**

**Profesor coordonator: Student:**

**Olteanu Gabriela Grosu Andrei**

**Georgian Neculoiu**

**București, 2025**

**Cuprins**

[**1. Introducere** 4](#_Toc199087737)

[**2. Obiectivele proiectului** 5](#_Toc199087738)

[**3. Aspecte teoretice privind logurile** 7](#_Toc199087739)

[**4. Tehnologii și instrumente folosite** 9](#_Toc199087740)

[**5. Implementarea aplicației** 10](#_Toc199087741)

[**6. Studii de caz** 13](#_Toc199087742)

[**7. Concluzii și posibile îmbunătățiri** 15](#_Toc199087743)

[**8. Bibliografie** 17](#_Toc199087744)

# **1. Introducere**

**În era digitală actuală, sistemele informatice sunt tot mai complexe, iar volumul de date generate zilnic este uriaș. În acest context, analiza logurilor devine un proces crucial pentru menținerea performanței, securității și stabilității aplicațiilor și infrastructurii IT. Am ales să dezvolt un analizator de loguri deoarece acesta reprezintă un proiect practic, util și cu aplicabilitate largă în numeroase domenii ale informaticii, de la administrarea sistemelor până la dezvoltarea software și ingineria DevOps.**

**Analiza logurilor este esențială pentru monitorizarea activității unui sistem, identificarea și diagnosticarea rapidă a erorilor, precum și pentru optimizarea performanțelor acestuia. Logurile conțin o multitudine de informații relevante, cum ar fi mesaje de eroare, avertismente, informații despre starea proceselor și utilizarea resurselor, iar procesarea eficientă a acestor date poate oferi o perspectivă detaliată asupra funcționării sistemului.**

**Proiectul realizat mi-a oferit oportunitatea de a lucra intensiv cu fișiere text, exploatând capabilitățile limbajului C++ pentru manipularea și prelucrarea stringurilor. Am implementat metode diverse de parsare a datelor, tehnici de filtrare pe baza criteriilor specifice și mecanisme de agregare a informațiilor extrase, toate acestea fiind esențiale pentru un analizator performant. Prin această experiență, am aprofundat concepte fundamentale precum parsarea sintactică, gestionarea eficientă a memoriei, utilizarea structurilor de date optimizate și manipularea excepțiilor.**

**De asemenea, dezvoltarea acestui instrument m-a ajutat să înțeleg mai bine importanța monitorizării automate în ecosistemele software moderne. În mediile de producție, detectarea rapidă a unor anomalii sau a erorilor poate preveni downtime-ul și pierderile financiare semnificative. Astfel, analizatorul de loguri este un instrument indispensabil pentru administratori de sistem, dezvoltatori software și ingineri DevOps, care oferă o imagine clară și actualizată asupra stării aplicațiilor și infrastructurii.**

**În concluzie, proiectul nu doar că mi-a permis să aplic cunoștințele teoretice dobândite, dar a și demonstrat relevanța practică a acestora în rezolvarea unor probleme reale din lumea tehnologiei informației. Analizatorul de loguri reprezintă o punte între datele brute colectate și deciziile strategice bazate pe aceste date, contribuind astfel la îmbunătățirea calității și fiabilității sistemelor informatice.**

# **2. Obiectivele proiectului**

**Proiectul analizatorului de loguri a fost conceput cu scopul de a dezvolta o aplicație practică și eficientă, care să răspundă nevoilor reale de prelucrare și interpretare a datelor din fișierele de log. Pentru a atinge acest scop, au fost stabilite următoarele obiective principale:**

* **Crearea unei aplicații capabile să deschidă și să parcurgă fișiere de log  
  Aplicația trebuie să poată gestiona fișiere text de dimensiuni variate, să le acceseze în mod eficient și să parcurgă conținutul acestora linie cu linie, pentru a putea prelucra datele în mod corect și rapid.**
* **Extracția informațiilor relevante din loguri. Analizatorul trebuie să identifice și să extragă elementele importante din loguri, cum ar fi mesajele de eroare, avertismentele și alte date statistice relevante. Această extragere implică filtrarea informațiilor utile, ignorând datele redundante sau neimportante, pentru a oferi o perspectivă clară asupra evenimentelor înregistrate.**
* **Oferirea unui meniu interactiv pentru utilizator, cu mai multe opțiuni personalizabile  
  Aplicația va pune la dispoziția utilizatorului un set de opțiuni prin care acesta poate filtra și segmenta datele după criterii precum intervalul de timp, tipul mesajelor (erori, avertismente, informații), nivelul de severitate sau alte parametri specifici. Acest meniu interactiv facilitează accesul rapid la informațiile de interes, adaptând analiza la nevoile specifice.**
* **Afișarea rezultatelor într-un mod clar și organizat. Rezultatele extrase vor fi prezentate într-un format ușor de înțeles și interpretat, prin structuri clare, eventual tabelare, evidențiind aspectele esențiale pentru utilizator și facilitând luarea deciziilor rapide.**
* **Salvarea rezultatelor într-un fișier de ieșire (opțional). Pentru o utilizare ulterioară sau pentru raportare, aplicația va permite exportul rezultatelor obținute într-un fișier extern, de tip text sau CSV, asigurând astfel păstrarea și partajarea facilă a datelor analizate.**
* **Posibilitatea de a analiza mai multe fișiere consecutiv fără repornirea aplicației  
  Pentru a crește eficiența și a evita pierderile de timp, aplicația va permite procesarea succesivă a mai multor fișiere, menținând sesiunile active și stările interne, astfel încât utilizatorul să poată compara și analiza date din surse multiple într-un singur flux de lucru.**
* **Gestionarea robustă a excepțiilor și erorilor. Proiectul va include mecanisme pentru detectarea și gestionarea erorilor care pot apărea în timpul rulării, precum încercarea de a deschide un fișier inexistent, întâlnirea unor formate nevalide sau corupte în loguri, asigurând astfel stabilitatea și fiabilitatea aplicației în orice situație.**
* **Crearea unei structuri de cod modulară și ușor de extins. Pentru a permite dezvoltări viitoare și adăugarea unor funcționalități suplimentare, codul sursă va fi organizat în module bine delimitate, cu responsabilități clare. Acest design modular facilitează întreținerea, testarea și scalarea aplicației pe termen lung.**

**Prin atingerea acestor obiective, proiectul urmărește să ofere un instrument util și flexibil, capabil să răspundă nevoilor diverse ale utilizatorilor din domeniul IT, în special celor care au nevoie să monitorizeze și să analizeze cantități mari de date generate de sistemele informatice.**

# **3. Aspecte teoretice privind logurile**

**Logurile, cunoscute și sub denumirea de fișiere de jurnal, reprezintă fișiere text care înregistrează în mod cronologic evenimentele care au loc într-un sistem informatic sau o aplicație software. Acestea sunt esențiale pentru monitorizarea și înțelegerea comportamentului sistemelor, oferind o sursă valoroasă de informații atât pentru dezvoltatori, cât și pentru administratorii de sistem.**

**În general, logurile conțin mai multe tipuri de informații, printre care se numără:**

* **Mesaje de informare (INFO): Acestea sunt înregistrări care reflectă starea normală a sistemului, confirmând funcționarea corectă a diverselor componente și procese.**
* **Avertismente (WARNING): Mesaje care indică situații potențial problematice, ce nu au dus încă la erori critice, dar care necesită atenție pentru a preveni eventuale defecțiuni.**
* **Erori (ERROR): Înregistrări care semnalează apariția unor probleme în sistem, de natură să afecteze funcționalitatea aplicației sau serviciului respectiv.**
* **Timestamp-uri (data și ora): Fiecare mesaj este de obicei însoțit de un marcaj temporal precis, care indică momentul exact când a fost generat evenimentul.**
* **Module sau funcții sursă: Unele loguri specifică sursa evenimentului, adică modulul, componenta sau funcția din codul sursă care a generat mesajul, facilitând localizarea problemei.**

**Structura logurilor poate varia semnificativ de la un sistem la altul, în funcție de cerințele aplicației, formatul folosit și preferințele dezvoltatorilor. Totuși, majoritatea urmează un format previzibil, cu elemente distincte și constante, ceea ce face posibilă interpretarea automată a acestora prin intermediul unor programe numite *parser-e*.**

**Această linie conține, în mod clar delimitat:**

* **un timestamp între paranteze pătrate, care indică data și ora evenimentului;**
* **nivelul de severitate al mesajului (ERROR);**
* **modulul sursă între paranteze drepte (AuthModule);**
* **și mesajul propriu-zis care descrie problema întâlnită.**

**Structura bine definită a unei astfel de linii de log facilitează extragerea automată a datelor relevante, prin identificarea unor *pattern-uri* fixe (de exemplu, datele delimitate de caractere specifice, cum sunt parantezele sau separatorii colon și spațiu).**

**Analiza logurilor oferă multiple beneficii practice:**

* **Identificarea rapidă a problemelor recurente: prin centralizarea și analizarea mesajelor de eroare și avertisment, se pot detecta defecte sau disfuncționalități frecvente în sistem.**
* **Evaluarea performanței sistemului: logurile pot include informații despre timpi de răspuns, utilizarea resurselor sau alte metrici relevante, care ajută la optimizarea aplicațiilor.**
* **Detectarea tentativelor de atac informatic: prin monitorizarea atentă a mesajelor anormale sau neautorizate, se pot identifica încercări de acces nepermise, activități malițioase sau vulnerabilități exploatate.**
* **Audit și conformitate: logurile servesc ca dovadă istorică pentru diverse procese, fiind indispensabile în cazuri de audit intern, securitate informatică sau investigare a incidentelor neprevăzute.**

**În concluzie, logurile sunt o componentă fundamentală în infrastructura IT, iar dezvoltarea unor instrumente eficiente pentru analiza lor este esențială pentru menținerea securității, stabilității și performanței sistemelor informatice moderne.**

**\**

# **4. Tehnologii și instrumente folosite**

**Pentru dezvoltarea analizatorului de loguri, am ales un set de tehnologii și instrumente care să asigure eficiență, performanță și o experiență de dezvoltare cât mai facilă.**

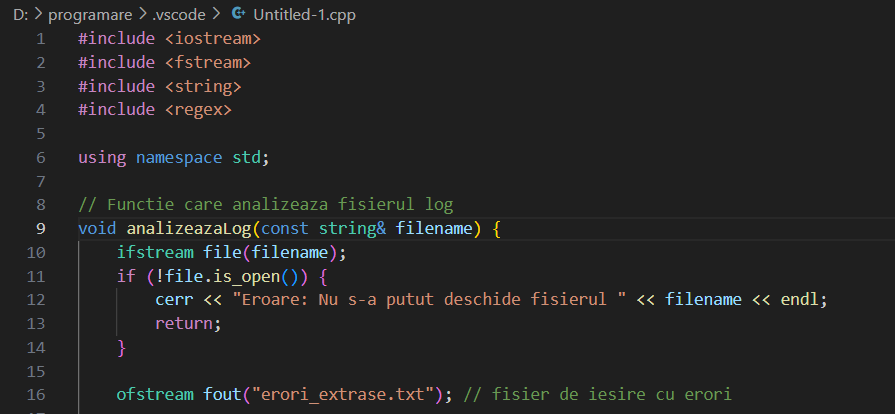
* **Limbajul C++: Am ales C++ datorită performanței sale ridicate și controlului detaliat asupra gestionării memoriei, ceea ce este esențial în prelucrarea unor volume mari de date, cum sunt fișierele de log. De asemenea, C++ oferă facilități avansate pentru manipularea stringurilor și implementarea unor algoritmi eficienți.**
* **STL (Standard Template Library): Biblioteca standard a limbajului C++ oferă structuri de date și algoritmi predefiniți, cum ar fi vectori, stringuri și facilități pentru manipularea fișierelor. Utilizarea STL simplifică implementarea, reducând complexitatea codului și crescând fiabilitatea.**
* **Visual Studio Code / Dev-C++: Pentru dezvoltarea aplicației, am folosit medii de dezvoltare integrate (IDE) precum Visual Studio Code și Dev-C++, care oferă facilități utile precum evidențierea sintaxei, debugging, integrare cu compilatoare și suport pentru gestionarea proiectelor.**
* **GIT (opțional): Controlul versiunilor a fost gestionat prin GIT, un sistem distribuit de versionare care permite urmărirea modificărilor în cod, colaborarea eficientă și păstrarea unui istoric al versiunilor, ceea ce este esențial în dezvoltarea proiectelor software de orice dimensiune.**
* **Google Test (opțional): Pentru asigurarea calității codului, am utilizat Google Test, un framework de testare automată pentru C++. Acesta permite crearea și rularea unor teste unitare care verifică funcționarea corectă a componentelor principale ale analizatorului, facilitând detectarea rapidă a erorilor și menținerea stabilității codului pe parcursul dezvoltării.**

# **5. Implementarea aplicației**

**Aplicația analizatorului de loguri a fost proiectată și dezvoltată într-un mod modular, pentru a facilita mentenanța, extinderea și testarea ușoară a componentelor sale. Structura sa principală include următoarele componente esențiale:**

**a) Citirea fișierului de log**

**Pentru a procesa fișierele de log, aplicația utilizează clasa ifstream din biblioteca standard C++. Fișierul este deschis și citit linie cu linie, fiecare linie fiind stocată temporar într-un vector de stringuri. Înainte de procesare, se efectuează verificări pentru a asigura existența fișierului și validitatea accesului la acesta, astfel încât să se evite erorile de citire sau blocarea aplicației.**

****

**b) Parsarea și analizarea liniilor**

**Fiecare linie citită este supusă unei operațiuni de parsare care extrage elementele esențiale ale unui mesaj de log. Acestea includ:**

* **Timestamp-ul: extras folosind funcții dedicate ce identifică și izolează subsecvența ce conține data și ora evenimentului.**
* **Tipul mesajului: identificarea nivelului de severitate, cum ar fi INFO, WARNING sau ERROR, prin căutarea unor șabloane fixe în text.**
* **Componenta sursă: dacă este specificată, se extrage denumirea modulului sau funcției care a generat mesajul.**
* **Conținutul mesajului propriu-zis: partea descriptivă a logului, care oferă detalii despre eveniment.**

**Aceste date sunt stocate într-o structură de date personalizată, de exemplu struct LogEntry, ce reunește toate câmpurile relevante. Această structură facilitează o manipulare clară și eficientă a informațiilor pe tot parcursul aplicației.**

**c) Funcții de analiză**

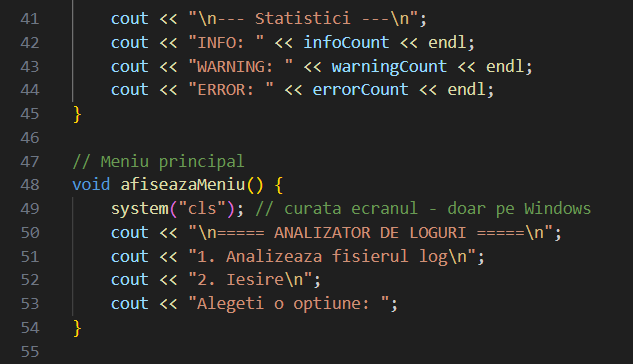
**Aplicația oferă o serie de funcționalități pentru procesarea și interpretarea datelor din loguri:**

* **Calcularea numărului total de linii din fișier, pentru a avea o imagine de ansamblu asupra volumului de date.**
* **Contorizarea mesajelor de tip erori, avertismente și informații, pentru o evaluare rapidă a stării sistemului.**
* **Posibilitatea de filtrare după dată, oră sau cuvinte cheie, ceea ce permite utilizatorului să focalizeze analiza pe perioade sau evenimente specifice.**
* **Funcție de căutare după modul sau componenta sursă, ajutând la identificarea zonelor din cod cu probleme frecvente.**
* **Generarea de statistici relevante, cum ar fi numărul de erori înregistrate pe zi sau ora în care apar cele mai multe probleme.**
* **Identificarea celor mai frecvente mesaje de eroare, cu un clasament de tip top 5 erori, care poate ajuta la prioritizarea remedierilor.**

**d) Interfață simplă de tip consolă**

**Pentru interacțiunea cu utilizatorul, aplicația oferă un meniu text interactiv, cu opțiuni clare și ușor de folosit:**

1. **Afișează statistici generale despre fișierul de log curent.**
2. **Permite filtrarea datelor după dată sau interval orar specificat.**
3. **Afișează exclusiv mesajele de tip ERROR.**
4. **Exportă rezultatele analizei într-un fișier de ieșire, pentru documentare sau prelucrare ulterioară.**
5. **Oferă posibilitatea de a reîncărca un fișier log nou fără a reporni aplicația.**
6. **Ieșirea din aplicație.**

****

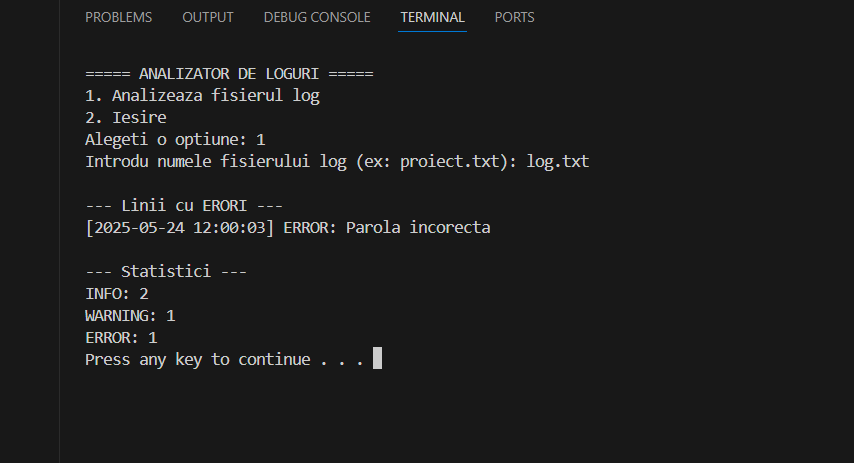
**Întregul cod a fost organizat pe module separate, cu fișiere .cpp și .h distincte, ceea ce a permis o dezvoltare clară și modulară. Această structurare nu doar că asigură o lizibilitate sporită, dar facilitează și testarea unităților funcționale, precum și adăugarea unor noi caracteristici în viitor, fără a afecta funcționalitatea existentă.**

# **6. Studii de caz**

**Pentru a valida funcționalitatea și eficiența analizatorului de loguri, am realizat mai multe teste folosind fișiere de log simulate, cu un volum semnificativ de date, depășind 1000 de linii fiecare. Aceste fișiere au conținut mesaje diverse, ordonate cronologic, reprezentând scenarii cât mai apropiate de situațiile reale întâlnite în administrarea sistemelor software.**

**Rezultatele testelor au demonstrat următoarele aspecte esențiale:**

* **Identificarea corectă a mesajelor de tip ERROR: Aplicația a recunoscut cu precizie toate intrările de tip eroare, indiferent de poziția lor în fișier, facilitând astfel detectarea problemelor majore din sistem.**
* **Filtrarea eficientă după criterii multiple: Funcționalitățile de filtrare au permis extragerea mesajelor în funcție de data și ora evenimentelor, precum și după cuvinte cheie specifice, oferind un instrument puternic pentru analiza detaliată a logurilor.**
* **Generarea rapoartelor sumare: Aplicația a creat rapoarte concise cu statistici despre frecvența diferitelor tipuri de mesaje (INFO, WARNING, ERROR), oferind o imagine clară asupra stării generale a sistemului monitorizat.**
* **Salvarea rezultatelor în fișiere CSV: Datele prelucrate au fost exportate cu succes în formate CSV, permițând ulterior analiza și vizualizarea cu alte instrumente specializate sau includerea în rapoarte oficiale.**

****

**Într-un alt scenariu de test, am introdus în fișierul de log un set de erori repetate pe o perioadă scurtă de timp. Aplicația a evidențiat acest „vârf” de erori în rapoartele statistice, semnalând astfel o problemă critică survenită la o anumită oră. Această capacitate de a detecta anomalii temporale este vitală pentru intervenții rapide și eficiente în medii de producție.**

**Aceste studii de caz au confirmat că analizatorul dezvoltat este un instrument robust și util pentru monitorizarea și diagnosticarea problemelor în sistemele informatice, oferind suport real pentru activități de mentenanță și optimizare.**

# **7. Concluzii și posibile îmbunătățiri**

**Realizarea acestui proiect mi-a oferit o experiență practică valoroasă în domeniul prelucrării fișierelor text și manipulării stringurilor, aspecte esențiale pentru orice analiză de loguri. Limbajul C++ s-a dovedit a fi o alegere inspirată, datorită performanței sale ridicate și controlului detaliat asupra memoriei, dar și pentru suportul puternic oferit prin Standard Template Library (STL). Dezvoltarea acestui analizator mi-a permis să-mi îmbunătățesc semnificativ gândirea algoritmică, în special prin conceperea și implementarea unor funcții eficiente și modulare pentru parsarea și extragerea datelor relevante din fișiere mari.**

**De asemenea, proiectul a evidențiat importanța organizării clare a codului și a separării funcționalităților în module distincte, ceea ce facilitează extinderea și întreținerea ulterioară a aplicației.**

**Posibile direcții de dezvoltare și îmbunătățiri viitoare:**

* **Suport pentru mai multe formate de log: Extinderea aplicației pentru a putea interpreta și analiza fișiere de log provenite din sisteme diferite, cum ar fi serverele web Apache sau Nginx, care au formate specifice. Această diversitate ar face instrumentul mai flexibil și util într-un spectru mai larg de aplicații.**
* **Interfață grafică (GUI): Dezvoltarea unei interfețe grafice folosind biblioteci precum Qt sau SFML pentru a oferi utilizatorilor o experiență mai intuitivă și mai prietenoasă. Un GUI ar permite interacțiunea vizuală cu datele, cu grafice, filtre dinamice și rapoarte vizuale.**
* **Integrarea unui motor de căutare cu expresii regulate (regex): Adăugarea suportului pentru căutări complexe folosind expresii regulate ar spori puterea de filtrare și analiză, oferind posibilitatea identificării unor tipare complexe în loguri.**
* **Exportul rezultatelor în formate multiple: Pe lângă CSV, includerea posibilității de a exporta datele în formate JSON sau XML ar facilita integrarea cu alte sisteme și fluxuri de lucru, precum aplicații web sau servicii de monitorizare automatizată.**
* **Sisteme de alertare automată: Implementarea unui mecanism de setare a alertelor care să notifice utilizatorul în timp real la apariția unor erori critice sau a unor pattern-uri repetitive de erori, pentru intervenții rapide.**
* **Integrare cu aplicații mobile: Dezvoltarea unei aplicații mobile sau a unui serviciu care să permită monitorizarea de la distanță a logurilor și primirea notificărilor, sporind accesibilitatea și capacitatea de reacție a administratorilor.**
* **Monitorizarea automată a directoarelor: Funcționalitate care să permită detectarea și încărcarea automată a fișierelor noi de log generate într-un director monitorizat, eliminând nevoia de reîncărcare manuală și asigurând o analiză continuă și actualizată.**

**În concluzie, proiectul a fost un pas important în dezvoltarea mea tehnică și înțelegerea proceselor de analiză a datelor textuale din sistemele informatice. Implementarea unor funcționalități suplimentare va transforma acest analizator într-un instrument complet, util pentru o gamă largă de utilizatori și scenarii de monitorizare.**

# **8. Bibliografie**

* **cppreference.com — Documentația detaliată a limbajului C++ și a Standard Template Library (STL). Disponibil online: https://en.cppreference.com/**
* **cplusplus.com — Resurse și tutoriale despre limbajul C++ și manipularea stringurilor, fișierelor și colecțiilor. Disponibil online:** [**https://www.cplusplus.com/**](https://www.cplusplus.com/)
* **GeeksforGeeks.org — Articole și exemple practice despre parsarea textului și analiza logurilor în C++ și alte limbaje. Disponibil online:** [**https://www.geeksforgeeks.org/**](https://www.geeksforgeeks.org/)
* **Documentația ISO C++17 — Specificațiile oficiale ale standardului C++17, care include caracteristici moderne utilizate în proiect.**
* **GitHub — Repozitorii publice cu exemple de fișiere de log și proiecte open-source pentru analiză de loguri.**
* **Tutoriale video pe YouTube — Serii de tutoriale despre analiza logurilor și manipularea fișierelor text în C++, cum ar fi cele realizate de The Cherno, CodeBeauty sau freeCodeCamp.**
* **"The Art of Logging" (whitepaper) — Document de specialitate care discută metodele și bunele practici în colectarea și analiza logurilor în aplicații software.**
* **Medium & Dev.to — Articole despre monitorizarea sistemelor distribuite, managementul logurilor și utilizarea uneltelor moderne de analiză.**