APLICAȚIE

**In-memory key-value database**



Îndrumător: Realizat de:

Slt. Adina VAMAN Sd. Sg. Cătălin BADEA

Sd. Sg. Alexandru-Mihai Duminică

Grupa C113D

**Cuprins**

Contents

[Capitolul 1 – Introducere 3](#_Toc150114650)

[1. Scopul proiectului 3](#_Toc150114651)

[1.1. Lista definițiilor 3](#_Toc150114652)

[1.2. Structura documentului 4](#_Toc150114653)

[Capitolul 2 – Arhitectură și componente SW 5](#_Toc150114654)

[1.1. Descrierea produsului software 5](#_Toc150114655)

[1.2. Detalierea platformei SW/HW 5](#_Toc150114656)

[1.3. Arhitectura internă 5](#_Toc150114657)

[1.4. Fluxul de lucru si testare(Descrierea solutiei): 6](#_Toc150114658)

[1.5. Descriere funcționalități 7](#_Toc150114660)

[Capitolul 3 – Code Snippets 10](#_Toc150114661)

# Capitolul 1 – Introducere

## Scopul proiectului

Crearea unei baze de date in-memory care să permită stocarea datelor sub forma de perechi cheie-valoare. Baza de date trebuie să suporte diferite tipuri de date și să ofere operații de bază precum stocarea, citirea și ștergerea datelor. De asemenea, trebuie să includă autentificare cu utilizator și parolă pentru a proteja accesul la date și să permită mai multor utilizatori să acceseze și să modifice datele simultan.

## 

## Lista definițiilor

**Cheie (Key):** O cheie este un identificator unic utilizat pentru a accesa o valoare specifica din baza de date. Aceasta poate fi o secventa de caractere sau un numar si este folosita pentru a indexa si recupera datele asociate.

**Valoare(Value):** Valoarea este informatia stocata in baza de date sub o anumita cheie. Aceasta poate fi de orice tip de date, cum ar fi siruri de caractere, numere, structuri de date.

**Stocate In-Memory**: Stocarea in memorie (in-memory) inseamna ca datele sunt pastrate temporar in memorie RAM a unui sistem, ceea ce permite un acces foarte rapid la date, datele fiind introduse in RAM la orice modificare a bazei sau rulare a server-ului.

**Operari CRUD**: acronim care se refera la operatiile de baza pe care le puteti efectua intr-o baza de date in memory key-value:

* Create: Operatia de creare sau adaugare a unei perechi cheie-valoare in baza de date;
* Read: Operatia de citire sau obtinere a valorii asociate unei chei din baza de date;
* Update: Operatia de actualizare a valorii asociate unei chei existente in baza de date;
* Delete: Operatia de stergere a unei chei si a valorii sale din baza de date.

**Set de date**: Structura de date care stocheaza un grup de valori unice, fara a tine cont de ordinea sau pozitia acestor valori in set;

**Lista**: O colectie de elemente de informatie (noduri), legate intre ele prin referinte, realizandu-se astfel o stocare necontigua a datelor in memorie;

**Multithreading**: Tehnica de programare care permite unui program sa execute mai multe parti ale codului sau in paralel sau concurential.

## Structura documentului

Documentul este împărțit în trei capitole. Capitolul 1 reprezintă introducerea. Capitolul 2 prezintă arhitectura aplicatiei si funcționalitățile pe care utilizatorul le are la dispozitie. Capitolul 3 cuprinde snippeturi de cod.

# Capitolul 2 – Arhitectură și componente SW

## Descrierea produsului software

Aplicația va fi dezvoltată în limbajul de programare C/C++. Modelul urmat este cel al serverului de baze de date REDIS.

## Detalierea platformei SW/HW

Produsul software este dezvoltat pentru dispozitivele pe care rulează sistemul de operare Linux. Vom utiliza mediul de dezvoltare Microsoft Visual Studio Code.

## Arhitectura internă

Ideea de baza a proiectului este proiectarea unui sistem de tip key-value pentru simplitatea utilizarii de catre persoanele care au nevoie, insa conceptul esential este cel de lucru in-memory, pentru rapiditatea returnarii rezultatelor.

Solutia software propusa este cea de utilizare a unui arbore binar de cautare(BST) care asigura accesul datelor din memoria ROM(datele salvate in fisiere) prin procesul de salvare al acestora in BST care asigura un timp de cautare logaritmic.

Se vor crea 3 fisiere pentru fiecare structura acceptata de serverul bazei de date(set, lista, date simple de tip key-value). Din aceste fisiere se vor extrage pentru memoria RAM si plasarea lor in arbore datele respective, pe rand. Se va folosi o functie de sortare simpla pe baza compararii intre cheile specifice fiecarui nod din arbore. Astfel, in partea stanga se vor plasa elementele „mai mici” din punct de vedere lexical si in dreapta viceversa.

Fisierele vor fi simple.txt, list.txt, set.txt si vor avea un continut de forma:

1. simple.txt

**cheia1-valoarea1**

**cheia2-valoarea2**

1. list.txt

**cheia1-element\_1,element\_2,element\_3,...,element\_n**

1. set.txt

**cheia1-element\_1,element\_2,element\_3,...,element\_n**

Serverul va contine de asemenea un fisier de credentiale prin intermediul careia se va realiza actiunea de login, introduse in memoria RAM doar prin intermediul unor simple variabile de citire a acestora din fisierul credentials.txt.

## Fluxul de lucru si testare(Descrierea solutiei):

* Utilizatorul va trimite un autenthication request catre server prin intermediul socket-ului care va contine ca mesaj username-ul si parola de autentificare data la input de catre utilizator, cat si codul de autentificare stabilit pentru a recunoaste protocolul utilizat de socket(De exemplu functia de send va contine mesajul (’’1 alex cata’’). Serverul va fi capabil sa interpreteze codul 1 ca fiind un authentication request si va raspunde un mesaj sugestiv catre client(’’Succes/Insucces’’)
* La rularea server-ului, datele vor fi deja incarcate in memoria RAM pentru a putea fi accesate rapid.
* User-ul va putea transmite comenzi in SHELL, care vor fi insa interpretate, iar in cazul unor erori se vor afisa mesaje sugestive odata cu verificarea corectitudinii comenzilor. Comenzile nu vor fi key-sensitive!
* In functie de codul specific fiecarei comenzi, se va modifica live atat arborele aflat in memoria RAM(pentru accesare concurentiale si a nu fi nevoiti sa rerulam server-ul), cat si fisierele din memoria ROM pentru a permite la o noua rulare a server-ului incarcarea corecta.
* Se va utiliza threading pool si incrementarea acestuia cu valori considerate potrivite pentru a permite accesul concurential(User1 modifica baza de date si User2 are acces imediat la modificarile lui User1)
* Existenta comenzii LOGGER care va executa functia „cat” asupra unui fisier logger.txt ce evidentiaza fiecare modificare, timestamp-ul acesteia, cat si user-ul care a realizat-o.
* Mijloace de detectie a erorilor diverse, precum verificarea dublarii cheilor in set-uri, stergerea unei chei inexistente,etc.

struct listNode

{

char\* value;

listNode\* next;

};

struct BST

{

char\* key;

listNode\* values;

bool isList;

BST\* leftNode;

BST\* rightNode;

## };

## Descriere funcționalități

Programul va fi rulat in terminal dupa structura: ./program. După rularea programului in terminal, utilizatorul va trebui sa introduca un id\_utilizator si o parola pentru a se loga la serverul bazei de date. Apoi utilizatorul va avea la dispozitie sa foloseasca sintaxe de tipul COMANDA KEY-VALUE si vor exista mai multe tipuri de comenzi:

SET key value: Stocheaza o valorea sub o cheie specifica;

GET key: Obtine valoarea asociata unei chei;

DEL key: Sterge o cheie si valoarea asociata acesteia;

RPUSH key value: Adauga o valoare la sfarsitul unei liste asociata unei chei;

LPUSH key value: Adauga o valoare la inceputul unei liste asociata unei chei;

LPOP key: Extrage si sterge prima valoare dintr-o lista asociata unei chei;

RPOP key: Extrage si sterge ultima valoare dintr-o lista asociata unei chei;

LRANGE KEY start stop: Extrage elementele dintr-o lista asociata cu o cheie incepand de la pozitia „start” si pana la pozitia „stop”;

SADD key member: Adauga un membru intr-o multime (set) asociata unei chei;

SREM key member: Sterge un membru dintr-o multime asociata unei chei;

SMEMBERS key: Obtine toti membrii dintr-un set;

SISMEMBER key member: Verifica daca un membru exista intr-un set;

SCARD key: Obtine numarul total de membri dintr-un set;

SINTER key1 key2: Obtine intersectia a doua seturi.

SUNION key1 key2: Obtine reuniunea a doua seturi.

Dupa fiecare executare a unei comenzi de mai sus, se va afisa un mesaj specific, eg. „Adaugare realizata cu succes!”, mesaje de eroare detaliate sau afisarea raspunsului cererii.

Alte comenzi pe care le va avea la dispozitie utilizatorul:

LOGOUT: Utilizatorul curent va fi delogat;

FINISH: Terminarea executiei programului client, cat si a serverului;

HELP: Va afisa o lista de comenzi disponibile ( cele mentionate mai sus).

In urma apelarii unei comenzi de catre orice client, se va salva intr-un fisier „logger” cine a executat-o, cat si momentul executiei.

LOGGER: Afisarea loggerului folosind functia linux „cat” sau „more”.

# Capitolul 3 – Code Snippets

Cod pentru a forma legatura intre un client si server, multithreading:

Client:

int client\_socket;

char message[100];

client\_socket = socket(AF\_INET, SOCK\_STREAM, 0);

struct sockaddr\_in server\_address;

server\_address.sin\_family = AF\_INET;

server\_address.sin\_port = htons(12345);

server\_address.sin\_addr.s\_addr = INADDR\_ANY;

int connection\_status = connect(client\_socket, (struct sockaddr\*)&server\_address, sizeof(server\_address);

if (connection\_status == -1) {

printf("Eroare la conectarea la server.\n");

return 1;

}

//Aici se citeste mesajul

fgets(message, sizeof(message), stdin);

send(client\_socket, message, sizeof(message), 0);

//Primire mesaj de la server

recv(client\_socket, message, sizeof(message), 0);

close(client\_socket);

Server:

int server\_socket, client\_socket;

char message[100];

server\_socket = socket(AF\_INET, SOCK\_STREAM, 0);

struct sockaddr\_in server\_address;

server\_address.sin\_family = AF\_INET;

server\_address.sin\_port = htons(12345);

server\_address.sin\_addr.s\_addr = INADDR\_ANY;

bind(server\_socket, (struct sockaddr\*)&server\_address, sizeof(server\_address));

listen(server\_socket, 5);

client\_socket = accept(server\_socket, NULL, NULL);

recv(client\_socket, message, sizeof(message), 0);

send(client\_socket, message, sizeof(message), 0);

close(server\_socket);

close(client\_socket);