Proiect Programare C++

SISTEM GESTIUNE BIBLIOTECA

Proiect realizat de:

Raul Marinescu

Robert Gasitu

CUPRINS

1.Motivarea alegerii temei

Am decis sa implementam proiectul cu titlul "Sistem Gestiune Biblioteca" din dorinta de a ajuta persoanele care se ocupa de administarea celor ce tin de o biblioteca.

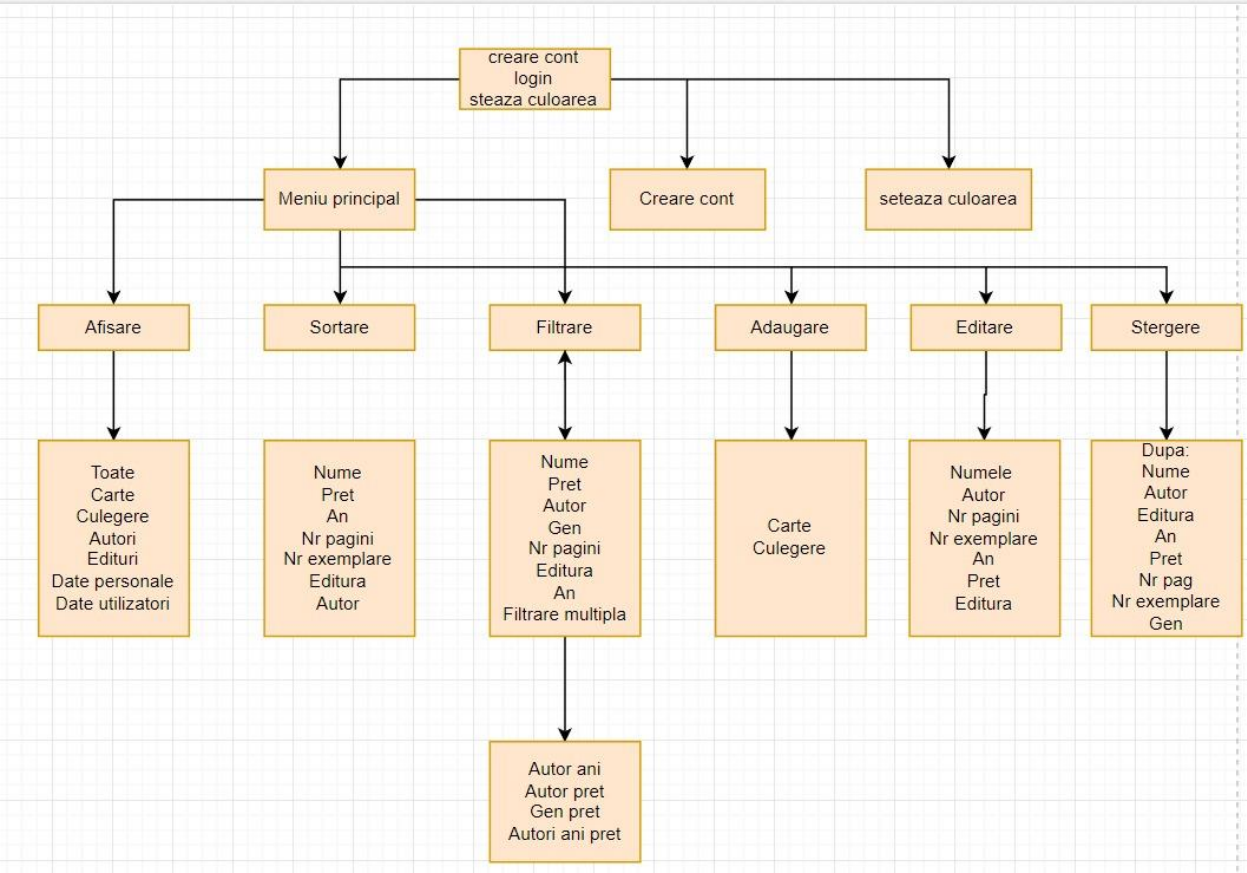
Din proprie experienta, am observat ca de cele mai multe ori, majoritatea bibliotecilor, fie ele a scolilor,oraselor,comunelor,etc. se confrunta cu problema digitalizari. Astfel, adesea, bibliotecarilor le este foarte greu sa gestioneze toate cartile pe care le au in biblioteca si sa tina o evidenta exacta a acestora.

In acest scop, am implementat un program de gestiune foarte prietenos si usor de folosit, care, cu siguranta le va fi de foarte ajutor persoanelor care se ocupa cu administrarea bibliotecilor.

Pe de alta parte, noi am ales acest proiect pentru a corela notiunile teoretice invatate in acest semestru la programarea orientata obiect cu un proiect palpabil, real, care necesita mult timp in ceea ce priveste implementarea deoarece ne-a ajutat sa ne dezvoltam si aprofundam aptitudinile in ceea ce priveste aceasta paradigma a programarii orientate obiect.

2. Structura proiect

2.1 Diagrama Functionalitati

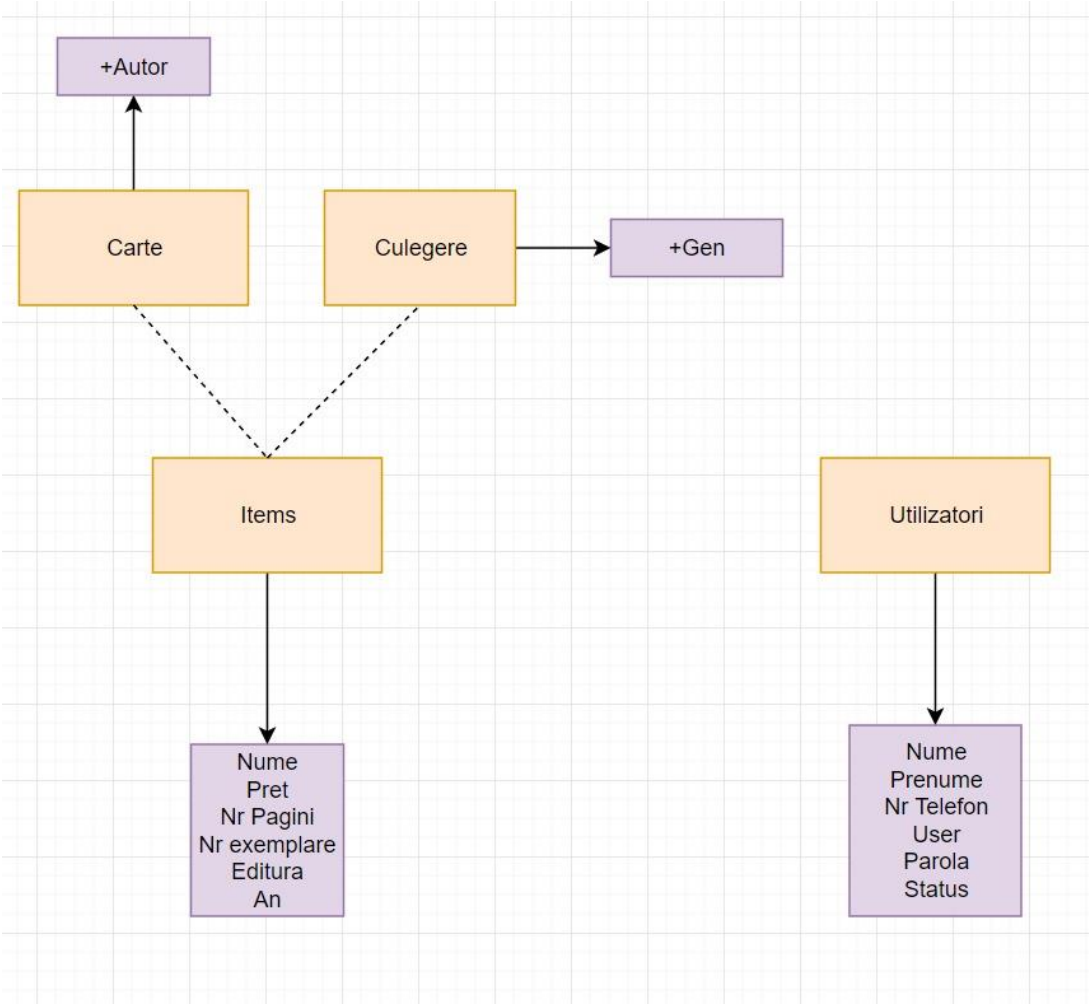


Dupa cum se poate obseva si in diagrama de mai sus, sistemul de gestiune prezinta numeroase functionalitati care pot fi folosite de catre utilizator in vedea lucrului intr-o biblioteca.

Obiectivul principal a fost de a eficientiza timpul administratorului de biblioteca, iar dupa cum se vede si mai sus, acest lucru este posibil din perspectiva diverselor operatii ce pot fi efectuate pe obiectele dintr-o

biblioteca si care, realizandu-se intr-un timp destul de scurt, vor fi de mare ajutor persoanei in cauza.

2.2 Diagrama OOP



Dupa cum se obseva si mai sus, proiectul prezinta **4 clase** care stau la baza implementarii si a tuturor operatilor ce se pot realiza in cadrul proiectului.

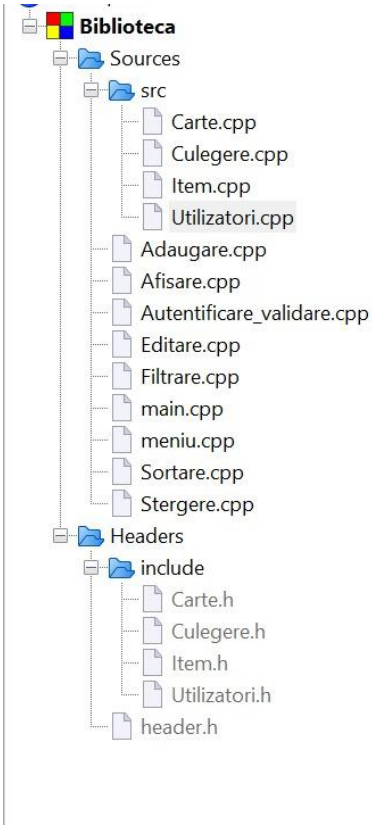
Pe de-o parte, avem clasa **Utilizatori** care a fost creata cu scopul de a tine evidenta mult mai bine a datelor referitoare la conturile utilizatorilor.

Durata acesteia de viata nu este influentata de restul claselor, cu alte cuvinte, este independenta de celelalte.

Iar pe de alta parte, avem clasa **Items** care este clasa de baza pentru clasele **Carte** si **Culegere**, evident, se remarca relatia de mostenire dintre acestea. De la clasa de baza Items se mostenesc atributele standard, ce sunt prezente si la clasele derivate (*nume , pret , numar pagini , editura ,anul*) iar celelalte doua clase , Culegere si Carte difera printr-un atribut specific : Cartea are *autor*, iar Culegere are *gen*.

3. Implementare Cod

3.1 Impartirea pe fisierele



Dupa cum se poate observa din cele doua capturi de ecran atasate , fiecare clasa, respectiv fiecare functionalitate de baza a programului este impartita intr-un fisier separat.

Aceasta practica a fost foarte utila deoarce in momentul in care sunt foarte multe obiective de implementat este mult mai usor sa identifici unde mai trebuie facute modificari sau adaugate functionalitati noi.

De precizat faptul ca fiecare fisier contine toate functionalitatiile ce tin de acesta.

Spre exemplu, in fisierul Autentificare_Validare.cpp sunt prezente functiile de creare cont, login si validarea datelor, meniu logare si deconectare.

3.2 Descrierea claselor

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;

class Utilizatori
{
      protected:
    string nume;
    string prenume;
    string user;
    string nrTlf;
    string parola;
    int status;
    public:
        Utilizatori(string nume, string prenume, string user, string nrTlf, string parola, int status);
        Utilizatori(const Utilizatori &u);
        friend ostream& operator <<(ostream& st, const Utilizatori& u);
        string getNume()const{
          return this->nume;
        }
         string getPrenume()const{
          return this->prenume;
        }
         string getUser()const{
          return this->user;
        }
         string getNrTlf()const{
          return this->nrTlf;
        }
         string getParola()const{
          return this->parola;
        }
        int getStatus()const{
          return this->status;
        }

        virtual ~Utilizatori();

    private:
        virtual ostream& afisare(ostream& st)const;
};

#endif // UTILIZATORI_H
```

Clasa Utilizatori are atributele : nume , prenume,user, nrTlf,parola si respectiv status (care se refera la conditia de admin)

respectiv metodele aferente : Constructorul , Getter pentru fiecare atribut de mai sus , respectiv suprascrierea operatorului de afisare.

```
#ifndef ITEM_H
#define ITEM_H
#include <iostream>
#include <string>

using namespace std;

class Item
{
protected:
    string nume;
    double pret;
    int nr_pagini;
    int nr_exemplare;
    string editura;
    int an;
    public:
        Item(string nume, double pret, int nr_pagini, int nr_exemplare, string editura, int an);
        friend ostream& operator<<(ostream& st, const Item& i);

        string getNume()const{
          return nume;
        }
        double getPret()const{
          return pret;
        }
        int getNr_pagini()const{
         return nr_pagini;
        }
        int getNr_exemplare()const{
         return nr_exemplare;
        }
        string getEditura()const{
         return editura;
        }
         int getAn()const{
          return an;
         }

         void setPret(double pret){
          this->pret = pret;
         }

         void setNr_pagini(int nr_pagini){
          this->nr_pagini = nr_pagini;
         }

         void setNr_exemplare(int nr_exemplare){
          this->nr_exemplare = nr_exemplare;
         }

         void setEditura(string editura){
          this->editura = editura;
         }


        virtual ~Item();


    private:
        virtual ostream& afisare(ostream& st)const;
};

#endif // ITEM_H
```

Asemanator clasei Utilizator , Item are aceleasi metode , dar in plus, aceasta prezinta si metoda Setter pentru fiecare atribut intrucat sunt necesare la functionalitatea de Editare a programului (vezi mai jos)

```
class Carte : public Item
{
    string autor;
    public:
        Carte(string nume,string autor, double pret, int nr_pagini, int nr_exemplare, string editura, int an);
         friend ostream& operator<<(ostream& st, const Carte& a);
```

```
class Culegere: public Item
{
    string gen;
    public:
        Culegere(string nume,string gen, double pret, int nr_pagini, int nr_exemplare, string editura, int an);
         friend ostream& operator<<(ostream& st, const Culegere& a);
```

Din clasa item se deriva clasa Carte si Culegere (prezentate mai sus) , unde exista o relatie de mostenire intre acestea. De precizat faptul ca fiecare dintre ele are suprascris operatorul de afisare.

3.3 Sistem de autentificare

```
void creaza_cont( vector<Utilizatori*> v,vector<Item*> item)
{
    string nume,prenume,user,nr,parola;
    int i=0;
    system("cls");
    cout<<"Introduceti:"<<endl;
    cout<<"Nume: ";
    do
    {
        if(i!=0)
            cout<<"Numele trebuie sa contina doar litere si sa inceapa cu majuscula!!"<<endl<<endl;
        cin>>nume;
        i++;

    }
    while(!valid_nume(nume));

    i=0;

    cout<<"Prenume: ";
    do
    {
        if(i!=0)
            cout<<"Prenumele trebuie sa contina doar litere si sa inceapa cu majuscula!!"<<endl<<endl;
        cin>>prenume;
        i++;

    }
    while(!valid_nume(prenume));
    i=0;

    cout<<"User: ";
    cin>>user;
    cout<<"Nr Telefon: ";
```

```
do
{
    if(i!=0)
        cout<<"Numarul de telefon trebuie sa contina 10 cifre!"<<endl<<endl;
    cin>>nr;
    i++;

}
while(!verif_nr(nr));

cout<<"Parola: ";
cin>>parola;
int x=0;
Utilizatori *u = new Utilizatori(nume,prenume,user,nr,parola,0);
for (auto it:v)
    if(it->getUser()==u->getUser() || it->getNrTlf()==u->getNrTlf())
        x=1;
if(x==1)
    cout<<"Exista deja un utilizator cu acest user sau numar de telefon!"<<endl<<endl;
else
{
    v.push_back(u);
}
ScriereFisier(v,item);
MeniuLogare(v,item);
```

Mai sus, este prezentata , prin intermediul a doua capturi de ecran , functia de creare cont. Utilizatorul trebuie sa isi adauge numele , prenumele , user , numarul de telefon si parola. In aceata functie exista anumite restrictii (validari). In momentul creari contului , utilizatorul trebuie sa isi introduca numele si prenumele cu litera mare la inceput si de asemenea numarul de telefon si user-ul sunt identificatori unici, astfel ca daca un utilizator introduce un user sau un nr. de telefon deja existent, va primi un mesaj pentru a introduce alta varianta.

```
bool verif_nr(string nr)
{
    if(nr.length()!=10)
        return false;
    for(int i=0; i<nr.length(); i++)
        if(nr[i]<'0' || nr[i]>'9')
            return false;
    return true;
}

bool valid_nume(string nume)
{
    if(nume[0]<'A' || nume[0]>'Z')
        return false;
    for(int i=1; i<nume.length(); i++)
        if((nume[i]<'a' || nume[i]>'z') && nume[i]!='-')
            return false;
    return true;
}
```

Aici sunt exemplificate cele doua functii de validare pentru numarul de telefon (trebuie sa fie format din 10 cifre) , respectiv pentru nume referitor la majuscula.

```
void login(vector<Utilizatori*> v,vector<Item*> item)
{
    system("cls");
    int ok = 0,k;
    string user, parola;
    char ch;
    const char ENTER = 13;
    cout<<"User: ";
    cin>>user;
    cout<<"Parola: ";
    while((ch = _getch()) !=ENTER)
    {
        parola += ch;
        cout<<'*';
    }

    Utilizatori *u;
    for(auto it: v)
        if(it->getUser()==user && it->getParola() == parola)
        {
            ok=1;
            u = it;
            k = it->getStatus();
            break;
        }

    if(ok==1)
    {
        system("cls");
        cout<<endl<<"Bine ati venit "<<user<<"!"<<endl;
        if(k == 0)
            Meniu_Principal(v,*u,item);
        else if( k == 1)
            Meniu_Principal_Admin(v,*u,item);
```

Functia de login functioneaza in felul urmator : Utilizatorul isi introduce user-ul si parola, dupa care acestea sunt cautate in vectorul de utilizatori pentru a verifica daca exista sau nu acel utilizator in baza de date al programului.

In cazul in care utilizatorul a fost gasit, se verifica status-ul si se redirectioneaza catre meniul aferent lui , fie pentru meniul de admin, fie catre cel standard.

De precizat faptul ca in momentul in care utilizatorul isi introduce parola, aceasta nu va fi afisata pe ecran in mod natural , ci in locul literelor introduse , se vor afisa , pe rand, stelute.

3.4 Sistem gestionare fisiere

```
void ScriereFisier(vector<Utilizatori*> &v, vector<Item*> &item)
{
    ofstream myfile;
    myfile.open("Utilizatori.txt");
    for(auto it : v)
    {
        string s = it->getNume();
        s = s.append(",");
        s = s.append(it->getPrenume());
        s = s.append(",");
        s = s.append(it->getUser());
        s = s.append(",");
        s = s.append(it->getNrTlf());
        s = s.append(",");
        s = s.append(it->getParola());
        s = s.append(",");
        s = s.append(to_string(it->getStatus()));
        myfile<<s<<endl;
    }
    myfile.close();

    ofstream myfile1;
    string s;
    myfile1.open("Item.txt");
    for(auto it : item)
    {

        if(dynamic_cast<Carte*>(it))
        {
            Carte * c = dynamic_cast<Carte*>(it);
            s = "Carte";
            s = s.append(",");
            s =  s.append(c->getNume());
            s = s.append(",");
            s = s.append(c->getAutor());
            s = s.append(",");
            s = s.append(to_string(c->getPret()));
            s = s.append(",");
            s = s.append(to_string(c->getNr_pagini()));
            s = s.append(",");
            s = s.append(to_string(c->getNr_exemplare()));
            s = s.append(",");
            s = s.append(c->getEditura());
            s = s.append(",");
            s = s.append(to_string(c->getAn()));

        }
        else if(dynamic_cast<Culegere*>(it))
        {
            s = "Culegere";
            Culegere *c = dynamic_cast<Culegere*>(it);
            s = s.append(",");
            s =  s.append(c->getNume());
            s = s.append(",");
            s = s.append(c->getGen());
            s = s.append(",");
            s = s.append(to_string(c->getPret()));
            s = s.append(",");
            s = s.append(to_string(c->getNr_pagini()));
            s = s.append(",");
            s = s.append(to_string(c->getNr_exemplare()));
            s = s.append(",");
            s = s.append(c->getEditura());
            s = s.append(",");
            s = s.append(to_string(c->getAn()));
        }
        myfile1<<s<<endl;
    }
    myfile1.close();

}
```

Mai sus este prezentata functia de scriere in fisier care actioneaza in doua directii diferite. Pe de-o parte se face scrierea in fisier pentru conturile de utilizator , iar pe de alta parte se face scrierea in fisier pentru iteme, care sunt de tip Carte sau Culegere.

Pentru a introduce in fisier datele , toate atributele unui obiect de tip utilizator sunt concatenate intr-un string, iar intre ele sunt separate prin virgula.

Acest procedeu este asemanator si pentru scrierea item-ului , fie el culegere sau carte, dar mai intai se testeaza prin intermediul functiei dynamic_cast tipul acesteia si se concateneaza la inceputul string-ului.

Item.txt - Notepad

File Edit Format View Help

```
Carte,Moara cu Noroc,Ioan Slavici,30.000000,180,60,Litera,1881
Culegere,Micul matematician,Matematica,750.000000,100,30,Paralela45,2007
Culegere,Bac,educational,50.000000,200,300,Sch,2009
Carte,Dumbrava minunata,Mihail Sadoveanu,20.000000,57,20,Libris,1974
Carte,Infern in Paradis,Lucy Clarke,44.900000,448,45,Bookzone,2022
```

```
void IncarcareDate(vector<Utilizatori*> v, vector<Item*> item)
{
    backup(item);
    string line,nume,prenume,user,parola,nr;
    int status;
    ifstream myfile ("Utilizatori.txt");
    if (myfile.is_open())
    {
        while (getline(myfile,line))
        {
            istringstream f(line);
            string s1;
            getline(f, s1, ',');
            nume=s1;
            getline(f, s1, ',');
            prenume = s1;
            getline(f, s1, ',');
            user = s1;
            getline(f, s1, ',');
            nr = s1;
            getline(f, s1, ',');
            parola = s1;
            getline(f, s1, ',');
            status = stoi(s1);
            Utilizatori *u = new Utilizatori(nume,prenume,user,nr,parola,status);
            v.push_back(u);

        }
        myfile.close();
    }

    else cout << "Unable to open file";
```

```
string tip,autor,editura,gen;
double pret;
int nr_pag,nr_ex,an;
ifstream myfile1 ("Item.txt");
if (myfile1.is_open())
{
    while (getline(myfile1,line))
    {
        istringstream f(line);
        string s1;
        getline(f, s1, ',');
        tip=s1;
        if(tip == "Carte")
        {
            getline(f, s1, ',');
            nume=s1;
            getline(f, s1, ',');
            autor=s1;
            getline(f, s1, ',');
            pret=stod(s1);
            getline(f, s1, ',');
            nr_pag=stoi(s1);
            getline(f, s1, ',');
            nr_ex=stoi(s1);
            getline(f, s1, ',');
            editura = s1;
            getline(f, s1, ',');
            an=stoi(s1);
            Carte* c = new Carte(nume
            item.push_back(c);
        }
        else if(tip=="Culegere")
        {
            getline(f, s1, ',');
            nume=s1;
            getline(f, s1, ',');
            gen=s1;
            getline(f, s1, ',');
            pret=stod(s1);
            getline(f, s1, ',');
            nr_pag=stoi(s1);
            getline(f, s1, ',');
            nr_ex=stoi(s1);
            getline(f, s1, ',');
            editura = s1;
            getline(f, s1, ',');
            an=stoi(s1);
            Culegere* c = new Culegere(nume,gen,pret,nr_pag,nr_ex,editura,an);
            item.push_back(c);
        }

    }
    myfile1.close();

}

else cout << "Unable to open file";


MeniuLogare(v,item);

}
```

Functia de preluare date din fisier este de asemenea predispusa pe doua directii : cea pentru preluarea datelor utilizatorilor si cea pentru preluarea cartilor si culegerilor.

Dupa ce se preia string-ul din fisier , acesta se “sparge” in token-uri in functie de catre separatorul virgula si se creaza obiectul de tip Utilizator, iar mai apoi este introdus in vectorul de Utilizatori , iar pentru obiectele din vectorul item m-ai intai se verifica primul cuvant din string (pentru a vedea daca este Culegere sau Carte) iar obiectul este creat in functie de aceasta conditie, iar mai apoi , la fel ca in cazul obiectelor de tip Utilizator , este introdus in vectorul de iteme.

3.5 Sistem Backup

```
void backup( vector<Item*> item)
{



    time_t   now = time(0);
    struct tm  tstruct;
    char     buf[80];
    tstruct = *localtime(&now);

    strftime(buf, sizeof(buf), "%Y-%m-%d-%X", &tstruct);
    string filename = buf;
    for(int i=0; i<filename.size(); i++)
        if(filename[i]==':')
            filename[i]='-';
    //cout<<filename;
    filename+=".txt";
    string path = "D:/Facultate/PP/Biblioteca/Biblioteca/Backup/";
    path+=filename;
    ofstream myfile (path);
    ifstream f("item.txt");
    string line;
    //copy_file(f,myfile);
    if (myfile.is_open())
    {
        if(f.is_open())
        {
            while(getline(f,line))
            {
                myfile<<line<<endl;
            }
            f.close();
            myfile.close();

        }

    }



}
```

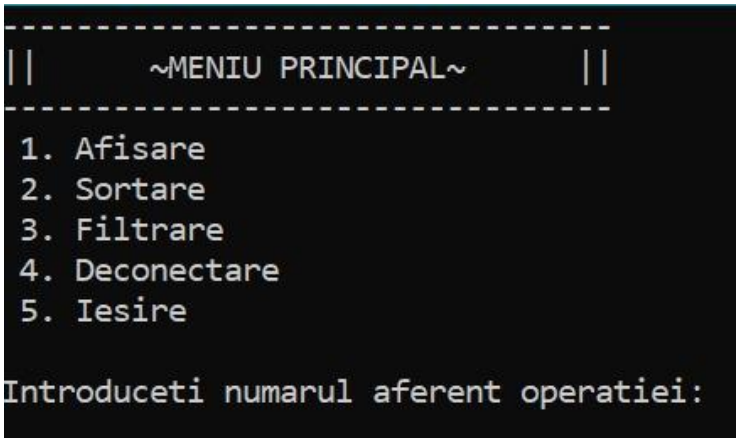
Functia de backup se apeleaza automat la fiecare rulare a programului si are ca scop crearea unei copii a datelor existente in fisierul principal unde sunt stocate obiectele de tip item (din fisierul item.txt).

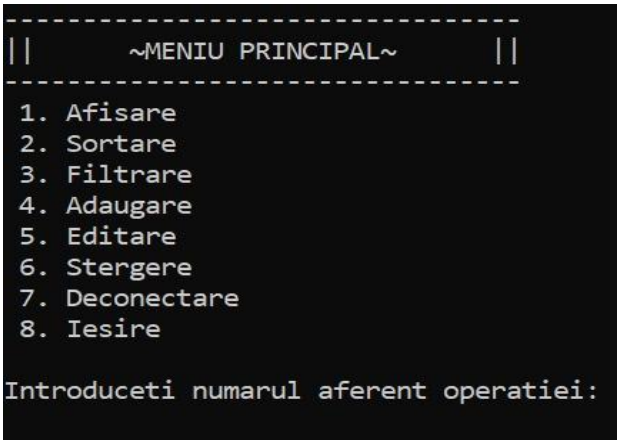
Aceasta creaza dinamic fisiere numite in functie de data curenta (exmplu YYYY-MM-DD-HH-MM-SS) si le salveaza automat intr-un alt folder special numit Backup.

Functia lucreaza in felul urmator : data curenta este salvata intr-un string cu ajutorul unei functii din libraria ctime dupa care modificam string-ul deoarece acesta este salvat cu caracterul ":" , iar acesta situatie nu este acceptata in numele unui fisier. Astfel ca . este inlocuit cu caracterul "-". La string-ul curent se adauga path-ul catre fisier, iar la final se adauga extensia corespunzatoare .txt .

Cu ajutorul functiei ofstream din string se deschide in acelasi timp fisierul curent si fisierul principal si se transcriu datele din cel principal in cel curent.

3.6 Meniuri principale





Dupa autentificare, in functie de status-ul utilizatorului , acesta va fi redirectionat catre unul dintre cele doua meniuri de mai sus. In cazul in care utilizatorul este administrator va fi redirectionat catre meniul din partea dreapta.

Cele doua meniuri difera de functionalitati. Cel din stanga este inclus in cel din dreapta , doar ca cel de admin are in plus meniurile de : adaugare , editare si stergere.

```
    do
    {
        if(i>0)
            cout<<"Numarul trebuie sa fie cuprins intre 1 si 8!"<<endl;
        i++;
        cin>>n;
    }
    while(n<"1"||n>"8");
    int m = stoi(n);
    switch(m)
    {
    case 1:
        Afisare(v,u,item);
        break;
    case 2:
        Sortare(v,u,item);
        break;
    case 3:
        Filtrare(v,u,item);
        break;
    case 4:
        Adaugare(v,u,item);
        break;
    case 5:
        Editare(v,u,item);
        break;
    case 6:
        Stergere(v,u,item);
        break;
    case 7:
        Deconectare(v,item);
        break;

    case 8:
        Iesire(v,item);
        break;
    }
}
```

Dupa cum se vede in poza alaturata , pentru a putea trece catre un meniu, trebuie introdusa cifra corespunzatoare operatiei dorite.

Acest procedeu este facut prin intermediul unui switch iar in cazul in care utilizatorul introduce altceva, cu ajutorul instructiunii repetitive do while , trebuie sa introduca o cifra valida.

Afisare

```
void  Afisare(vector<Utilizatori*> &v,Utilizatori &u,vector<Item*> &item);
void Afis_Toate(vector<Utilizatori*> &v,Utilizatori &u,vector<Item*> &item);
void Afis_Carte(vector<Utilizatori*> &v,Utilizatori &u,vector<Item*> &item);
void  Afis_Culegere(vector<Utilizatori*> &v,Utilizatori &u,vector<Item*> &item);
void  Afis_Autor(vector<Utilizatori*> &v,Utilizatori &u,vector<Item*> &item);
void Afis_Editurile(vector<Utilizatori*> &v,Utilizatori &u,vector<Item*> &item);
void  Afis_Date(vector<Utilizatori*> &v,Utilizatori &u,vector<Item*> &item);
void Afis_Utiliz(vector<Utilizatori*> &v,Utilizatori &u,vector<Item*> &item);
```

```
--------------------------------
||       ~MENIU AFISARE~      ||
--------------------------------

  Alegeti ce doriti sa afisati

 1. Toate
 2. Carti
 3. Culegeri
 4. Autori
 5. Editurile
 6. Date Personale
 7. Date utilizatori
 8. Inapoi

Introduceti numarul aferent operatiei:
```

Functionalitatiile de mai sus fac parte din submeniul de afisare, iar dupa cum se poate vedea exista 7 la numar. In cazul in care se alege una din primele 3 optiuni se vor afisa sub forma de tabel itemele respective.

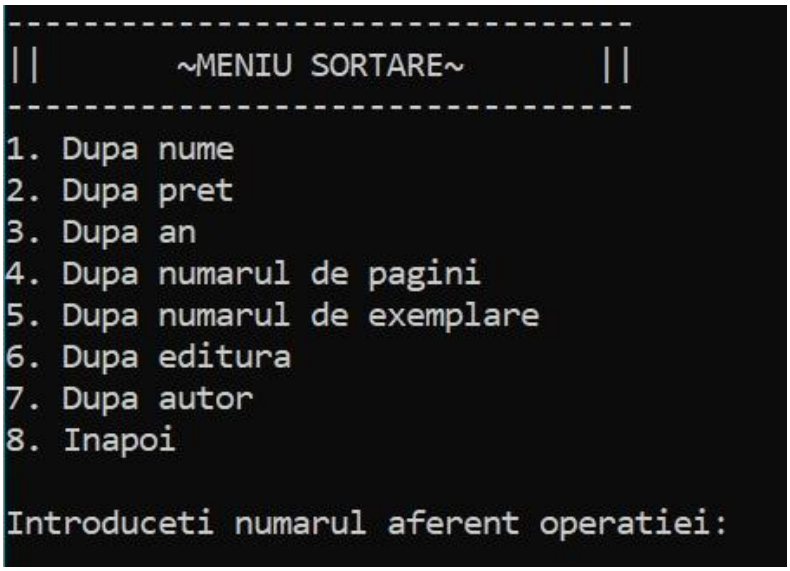
NUME	AUTOR	PRET	NR_PAGINI	EXEMPLARE	EDITURA	ANUL
Moara cu Noroc	Ioan Slavici	30	180	60	Litera	1881
Dumbrava minunata	Mihail Sadoveanu	20	57	20	Libris	1974
Infern in Paradis	Lucy Clarke	44.9	448	45	Bookzone	2022
Luceafarul	Mihai Eminescu	10	271	30	Agora	2008
Floarea albastra	Mihai Eminescu	10	288	40	Nemira	2017
Tata bogat tata sarac	Robert T. Kitosaki	30	232	55	Curtea Veche	2008
Sfantul surferul si CEO-ul	Robin Sharma	25	224	100	Livingstone	2013
Calugarul care si-a vandut ferrariu-ul	Robin Sharma	35.15	221	43	Excalibur	2010
Cautand-o pe Alaska	Jhon Green	45	288	321	Trei	2014
Mara	Ioan Slavici	9.75	304	15	Excalibur	1906
Elevul Dima dintr-a saptea	Mihai Drumes	29.9	608	30	Art	2020
Casa fara oglinzi	Tove Alstredal	44	280	47	RAO	2022

In cazul in care se alege una dintre optiunile de afisare Autori sau Edituri se va crea un map<string,vector<string>> , unde pe prima pozitie va fi autorul / editura , iar pe a doua pozitie se va afisa vectorul . De asemenea, inainte de afisarea vector-ului de entitati , se va afisa si numarul de entitati existente pentru o anumita cheie.

```
map<string,vector<string>> m;
for(auto it: item)
{

    if(dynamic_cast<Carte*>(it))
    {
        Carte* c = dynamic_cast<Carte*>(it);
        m[c->getAutor()].push_back(c->getNume());
    }
}
for(map<string, vector<string>>::iterator it = m.begin(); it != m.end(); it++)
{
    cout<<it->first<<": "<<(it->second).size()<<" ";
    for(auto i: (it->second))
        cout<<i<<", ";
    cout<<endl;
}
```

Sortare



Sortarea se face in functie de unul dintre criterile din poza din stanga. Criteriile fiind luate dupa atributele corespunzatoarea claselor Carte si Culegere.

In momentul in care se alege o optiune din cele 7, utilizatorul va fi redirectionat catre alt submeniu unde va trebui sa selecteze cum doreste sa fie sortate obiectele respective (in ordine crescatoare sau in ordine descrescatoare).

```
void Sort_AutorD(vector<Utilizatori*> v, Utilizatori &u,vector<Item*> item)
{
    vector<Carte*> itm;
    for(auto it: item)
    {
        if(dynamic_cast<Carte*>(it))
        {
            Carte *c = dynamic_cast<Carte*>(it);
            itm.push_back(c);
        }
    }
    sort(itm.begin(),itm.end(),[](auto a, auto b)
    {
        return  a->getAutor() == b->getAutor()?  a->getNume() > b->getNume() : a->getAutor() > b->getAutor();
    });
```

La inceputul functiei, se creaza o copie a vectorului , iar pe acela se va face sortarea. Spre exemplu , in cazul sortarii descrescatoare in functie de autor, in cazul in care se gasesc doua iteme cu acelasi autor se va alege al doilea criteriu de sortare, care este lexicografix descrescator dupa nume.

Filtrari

In cadrul filtrarilor exista doua optiuni : fie o filtrare simpla in functie de un singur atribut, fie o filtrare multipla in functie de doua sau trei atribute.

```
--------------------------------
||      ~MENIU FILTRARE~       ||
--------------------------------
Filtreaza dupa:

1. Nume
2. Pret
3. Autor
4. Gen
5. Numar pagini
6. Editura
7. An
8. Filtrare multipla
9. Inapoi

Introduceti numarul aferent operatiei:
```

```
-----------------------------------------
||        ~MENIU FILTRARE MULTIPLA~     ||
-----------------------------------------
Filtreaza dupa:

1. Numele autorului si interval de ani.
2. Numele autorului si interval de pret.
3. Numele autorului, intervalul de ani si pret.
4. Gen si pret.
5. Inapoi.

Introduceti numarul aferent operatiei:
```

In cazul filtrarii din prima poza, de exemplu, dupa nume , nu trebuie scris cu exactitate numele item-ului intrucat programul va cauta numele introdus in numele cartii si va afisa rezultatele valide.

```
void Filt_nume(vector<Utilizatori*> v, Utilizatori &u,vector<Item*> &item)
{
    cout<<"Introduceti numele: ";
    string s,s1,s2;

    cin.ignore();
    getline(cin,s);
    cout<<endl;
    s2=s;
    transform(s.begin(), s.end(),s.begin(), ::tolower);
    s1=s;
    transform(s1.begin(), s1.begin()+1,s1.begin(), ::toupper);
    afis_tabel(item,s,s1,s2);
```

```
Introduceti numele: m

NUME                              AUTOR/GEN           PRET    NR_PAGINI  EXEMPLARE   EDITURA         ANUL
Moara cu Noroc                    Ioan Slavici        30      180        60          Litera          1881
Micul matematician                Matematica          750     100        30          Paralela45      2007
Dumbrava minunata                 Mihail Sadoveanu    20      57         20          Libris          1974
Mara                              Ioan Slavici        9.75    304        15          Excalibur       1906
Elevul Dima dintr-a saptea        Mihai Drumes        29.9    608        30          Art             2020
Matematica clasa a 4-a            Matematica          12.74   96         28          Sinapsis        2018
Chimie pentru gimnaziu            educational         19.92   411        16          ART GRUP        2019
Teste chimie                      educational         25.5    224        9           INFOMEDIA       2014
Vocabularul limbii romane         Romana              47.5    128        97          ART GRUP        2022
Atlas geografic Romania-Lupa      Geografie           22.5    63         89          Carta Atlas     2000
Biologie vegetala si animala      Biologie            29.75   168        46          Nominatrix      2002
```

Iar in capul filtrarii multiple trebuie introduse mai multe atribute dupa care cautam ceva anume si se vor afisa toate cele care verifica conditiile impuse.

```
void Filt_mult_autor_ani(vector<Utilizatori*> v, Utilizatori &u, vector<Item*> &item)
{
    system("cls");
    string nume;
    int an_min,an_max;
    cout<<"Introduceti numele autorului:";
    cin.ignore();
    getline(cin,nume);

    cout<<"Introduceti intervalul de ani in care se efectueaza cautarea:"<<endl;
    cout<<"De la: ";
    cin>>an_min;
    cout<<"Pana la: ";
    cin>>an_max;
    int ok=0;
    system("cls");
    for(auto it:item)
        if(dynamic_cast<Carte*>(it))
        {
            Carte*i = dynamic_cast<Carte*>(it);
            if(i->getAutor()==nume && (i->getAn()>=an_min && i->getAn()<=an_max))
            {
```

Spre exemplu daca introducem in cazul filtrarii multiple dupa autor si ani urmatoarele date : Ioan slavici intre anii 1850 – 1900 se va afisa :

NUME	AUTOR	PRET	NR_PAGINI	EXEMPLARE	EDITURA	ANUL
Moara cu Noroc	Ioan Slavici	30	180	60	Litera	1881

Stergere :

In cazul stergerii, se pastreaza aceleasi criterii ca la sortare , astfel ca se pot face stergeri in functie de : autor, nume , editura , an , numar paginii, etc….

In cazul in care se selecteaza autor sau editura ca criteriu de stergere se vor elimina din memorie toate itemele care sunt scrise de acel autor sau sunt publicate de aceiasi editura.

Editare

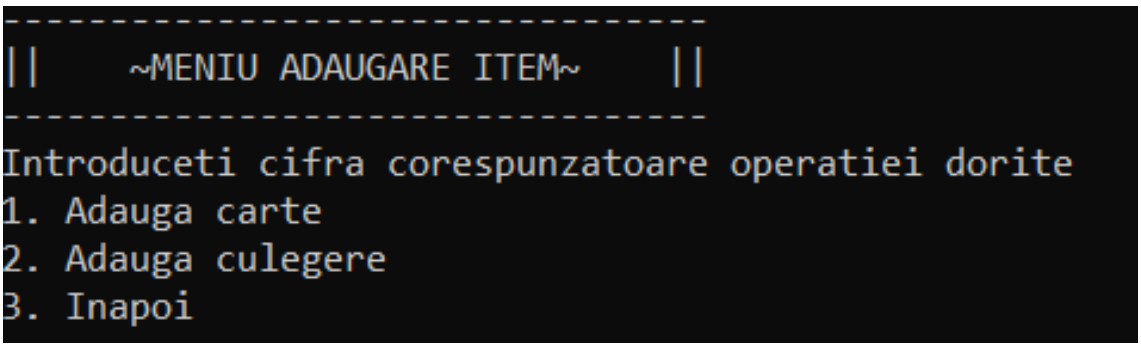
Functia de editare are ca scop editarea unor atribute ale unui item, fie el carte sau culegere. Scopul acestei functionalitatii este de putea modifica ceva care , spre exemplu , in momentul introducerii, s-au introdus date gresite.

Dupa editare , acestea vor fi introduse din nou in fisier in varianta editata.

Adaugare

Functia de adaugare permite administratorului sa introduca noi obiecte in biblioteca. Trebuie completate toate datele referitoare la acel obiect , fie el de tip carte sau culegere.

De mentionat faptul ca se verifica daca acel obiect exista deja in baza de date. In cazul in care nu exista , se adauga , iar altfel nu se va adauga.



4. Stilizare

In momentul deschiderii programului , utilizatorul poate opta pentru a-si stiliza consola dupa cum doreste. Acesta poate sa isi aleaga culoarea de fundal , respectiv culoarea font-ului.

```
void culoare(vector<Utilizatori*> v,vector<Item*> item)
{
    system("cls");
    cout<<"1. Text albastru"<<endl;
    cout<<"2. Text verde"<<endl;
    cout<<"3. Text rosu"<<endl;
    cout<<"4. Text galben"<<endl;
    cout<<"5. Default"<<endl;
    cout<<"6. Text albastru fundal rosu"<<endl;
    cout<<"7. Text rosu fundal galben"<<endl;
    cout<<"8. Text galben fundal violet"<<endl;
    cout<<"9. Text negru fundal gri"<<endl;


    int i,n;
    do
    {
        if(i>0)
            cout<<"Numarul trebuie sa fie cuprins intre 1 si 9!"<<endl;
        i++;
        cin>>n;
    }
    while(n<1||n>9);
    switch(n)
    {


    case 1:
        system("color 09");
        break;
    case 2:
        system("color 02");
        break;
    case 3:
        system("color 04");
        break;
    case 4:
        system("color 06");
        break;
    case 5:
        system("color 07");
        break;
    case 6:
        system("color C1");
        break;
    case 7:
        system("color E4");
        break;
    case 8:
        system(" Color 56");
        break;
    case 9:
        system("color 70");
        break;

    }
    MeniuLogare(v,item);
}
```

```
---------------------------------
||      ~MENIU PRINCIPAL~      ||
---------------------------------
 1. Afisare
 2. Sortare
 3. Filtrare
 4. Adaugare
 5. Editare
 6. Stergere
 7. Deconectare
 8. Iesire

Introduceti numarul aferent operatiei:
```

Din punct de vedere al afisarii datelor in urma unei optiuni alese din meniu, acestea se vor identa sub forma de tabel. Aceata modalitate de afisare a fost implementata prin intermediul bibliotecii <iomanip> .

```
void Afis_Toate(vector<Utilizatori*> &v,Utilizatori &u,vector<Item*> &item)
{
    system("cls");

    cout << left << setw(40) << "NUME" << left << setw(20) << "AUTOR/GEN" << left
         << setw(8)<< "PRET" << left << setw(11) << "NR_PAGINI"  << left << setw(12) << "EXEMPLARE" << left << setw(16) <<"EDITURA" << left << setw(4) << "ANUL"<<endl;
    for(auto it: item)
    {
        if(dynamic_cast<Carte*>(it))
        {
            Carte*i=dynamic_cast<Carte*>(it);
            cout << left << setw(40) << i->getNume()
                 << left << setw(20) << i->getAutor()
                 << left << setw(8) << i->getPret()
                 << left << setw(11) << i->getNr_pagini()
                 << left << setw(12) << i->getNr_exemplare()
                 << left << setw(16) << i->getEditura()
                 <<left << setw(4) <<  i->getAn()
                 << endl;
        }
        else
        {
            Culegere*i=dynamic_cast<Culegere*>(it);
            cout << left << setw(40) << i->getNume()
                 << left << setw(20) << i->getGen()
                 << left << setw(8) << i->getPret()
                 << left << setw(11) << i->getNr_pagini()
                 << left << setw(12) << i->getNr_exemplare()
                 << left << setw(16) << i->getEditura()
                 <<left << setw(4) <<  i->getAn()
                 << endl;
        }

    }
    char ch;
    do
    {
        cin>>ch;

    }
    while(ch==' ');
    Afisare(v,u,item);
}
```

Prin intermediul functiei left se identeaza catre stanga tot textul iar prin intermediul functiei setw( ) se seteaza latimea care este acordata unui camp.

```
Introduceti numele: m

NUME                                    AUTOR/GEN           PRET    NR_PAGINI   EXEMPLARE   EDITURA         ANUL
Moara cu Noroc                          Ioan Slavici        30      180         60          Litera          1881
Micul matematician                      Matematica          750     100         30          Paralela45      2007
Dumbrava minunata                       Mihail Sadoveanu    20      57          20          Libris          1974
Mara                                    Ioan Slavici        9.75    304         15          Excalibur       1906
Elevul Dima dintr-a saptea              Mihai Drumes        29.9    608         30          Art             2020
Matematica clasa a 4-a                  Matematica          12.74   96          28          Sinapsis        2018
Chimie pentru gimnaziu                  educational         19.92   411         16          ART GRUP        2019
Teste chimie                            educational         25.5    224         9           INFOMEDIA       2014
Vocabularul limbii romane               Romana              47.5    128         97          ART GRUP        2022
Atlas geografic Romania-Lupa            Geografie           22.5    63          89          Carta Atlas     2000
Biologie vegetala si animala            Biologie            29.75   168         46          Nominatrix      2002
```

5. Raport activitate

Proiectul a fost in proportie de 80% implementat prin intalniri fizice, iar media de lucru per intalnire a fost de 4 ore. Aproximativ, intalnirile fizice au fost in numar de 7-8 , iar restul de 20% au fost intalniri online (in jur de 3) .

Codul este alcatuit din 3767 linii de cod si aproximativ 77 de functii care reprezinta functionalitatiile programului de gestiune a bibliotecii.

6. Bibliografie

Materiale curs de pe classroom + stepik
GeeksForGeeks
Youtube
StackOverFlow
+ alte site-uri