

# Proiect POO

## Gestionare cheltuieli

Student: Țibuleac Bogdan-Ionuț

Grupa: 3121A

An universitar: 2022-2023

# Cuprins

1. Tema și motivația alegerii.....	2
2. Descrierea problemei și ce s-a dorit a fi implementat .....	2
3. Abordarea problemei .....	3
4. Elemente specifice POO .....	7
4.1. Diagrama UML a claselor .....	7
4.2. Clasa Data.....	8
4.3. Clasa Plata .....	9
4.4. Clasa BunuriServicii.....	10
4.5. Clasa GestionareCheltuieli .....	11
4.6. Lucrul cu fișiere.....	17
5.    Informatii despre rularea proiectului .....	23
6. Concluzii .....	24
7. Bibliografie .....	25
8. Caiet de sarcini .....	26

## **1. Tema și motivația alegerii**

Tema acestui proiect este realizarea unei aplicații de gestionare a cheltuielilor unei persoane.

Am ales această temă deoarece este foarte important ca o persoană să țină cont de modul în care își gestionează veniturile. Având la dispoziție o aplicație în care poate trece cheltuielile zilnice, ce sunt împărțite pe categorii, va fi mult mai ușor ca după o perioadă de timp, să gestioneze mai eficient economiile. Printre avantajele unei astfel de aplicații amintim:

1. Organizare și urmărire eficientă a cheltuielilor: O aplicație de gestionare a cheltuielilor permite utilizatorilor să monitorizeze și să urmărească cu ușurință toate cheltuielile pe care le fac. Aceasta oferă o imagine de ansamblu clară asupra modului în care sunt cheltuiți banii și ajută utilizatorii să fie mai organizați în administrarea finanțelor personale.
2. Planificare financiară: O aplicație de gestionare a cheltuielilor poate ajuta utilizatorii să-și creeze un buget și să-și stabilească obiective financiare realiste. Prin monitorizarea cheltuielilor și compararea lor cu veniturile, aplicația poate oferi sugestii și sfaturi personalizate pentru a ajuta utilizatorii să economisească bani și să-și îndeplinească obiectivele financiare pe termen scurt și lung.
3. Controlul cheltuielilor: Prin utilizarea unei aplicații de gestionare a cheltuielilor, utilizatorii pot identifica și analiza zonele în care se cheltuie cel mai mult. Aceasta permite identificarea obiceiurilor de cheltuieli nesănătoase sau excesive și ajută la luarea unor decizii mai bune în ceea ce privește cheltuielile viitoare. Aplicația poate trimite notificări sau avertismente când se depășesc anumite limite de cheltuieli prestabilite.
5. Sănătate financiară și reducerea stresului: O aplicație de gestionare a cheltuielilor poate contribui la creșterea sănătății financiare generale a unei persoane. Prin furnizarea unui cadru clar și structurat pentru gestionarea banilor, aplicația poate reduce stresul și anxietatea asociate cu problemele financiare. Utilizatorii pot simți o mai mare încredere în abilitatea lor de a-și gestiona eficient banii și de a lua decizii financiare înțelepte.

## **2. Descrierea problemei și ce s-a dorit a fi implementat**

O aplicație de gestionare a cheltuielilor ar fi utilă unei persoane deoarece ar ajuta la monitorizarea și controlul cheltuielilor. Aceasta ar permite utilizatorului să introducă și să stocheze informații despre toate cheltuielile făcute într-o perioadă de timp. O astfel de aplicație ar putea să organizeze aceste informații în diferite categorii, cum ar fi alimente, transport, îmbrăcăminte sau divertisment, astfel încât utilizatorul să poată vedea cu ușurință în ce domenii anume cheltuiește cel mai mult. Aplicația ar putea să ofere, de asemenea, grafice pentru a permite utilizatorului să vizualizeze cheltuielile într-un mod mai ușor de înțeles și să ia decizii informate despre cum să-și administreze mai bine banii.

Astfel, conform necesităților beneficiarului, s-a dorit crearea unei aplicații care să permită utilizatorului să adauge o plată nouă care va cuprinde informații referitoare la data, numele și tipul bunului sau serviciului, cantitatea și prețul acesteia. S-a dorit ca utilizatorul să aibă posibilitatea de a căuta, după anumite criterii o plată, să o șteargă și să afișeze toate plățile efectuate. În plus, utilizatorul poate calcula suma totală cheltuită într-un interval de timp. Aplicația vine cu o listă predefinită de bunuri și servicii pentru care utilizatorul are la dispoziție opțiunile de adăugare și ștergere. Toate plățile sunt salvate într-un fișier.

### 3. Abordarea problemei

Interfața aplicației este reprezentată de meniul principal (GESTIONARE CHELTUIELI) ce încapsulează mai multe opțiuni. Prima parte a opțiunilor se ocupa de prelucrarea plăților: de a adaugă sau de a șterge o plată, de a vedea plățile introduse, și de a căuta în funcție de niște criterii o anumită plată. Mai mult, opțiune de Calcul total plăți oferă posibilitatea de a calcula suma plăților dintr-un interval de timp. Cea de-a doua parte permite adăugarea sau ștergerea unui bun sau serviciu din lista predefinită din fișierul cvs, precum și vizualizarea celor deja existente.

Deoarece utilizatorul a dorit să aibă opțiunea de a crea grafice sau diagrame în funcție de plățile efectuate, dar acest lucru fiind aproape imposibil de implementat în consolă, s-a decis utilizarea funcțiilor deja definite în excel de creare a graficelor, diagramelor.

```
GESTIONARE CHELTUIELI

1.  Adauga plata noua
2.  Cautare plata
3.  Sterge plata
4.  Afisare plati
5.  Calcul total plati

6.  Adauga un nou bun/serviciu
7.  Afisare bun/serviciu
8.  Sterge bun/serviciu

9.  IESIRE

Foloseste tastele "up arrow" "down arrow" pentru a naviga intre optiuni.
Apasa Enter pentru a intra intr-o optiune.
```

Interfața pentru adăugarea unei plăți:

Introduceti detaliile platii:

Introduceti data (ZZ/LL/AAAA):

Zi - (ZZ) : 15

Luna - (LL) : 12

An - (AAAA): 2022

Numele bunului sau serviciului: brad Craciun

Tipul bunului sau serviciului:

Bunuri si servicii existente:

0: Mancare

1: Educatie

2: Sanatate

3: Calatorie & Transport

4: Imbracaminte

5: Facturi & Utilitati

6: Sport

7: Vacanta

8: Urgente

9: Cadouri

10: Renovare

11: Curatenie Casa

12: Electronice & Electrocasnice

13: Divertisment

14: Diverse

15: Amenzi

Optiuni:

[01] - Alegeti un bun sau serviciu din lista aceasta

[02] - Adaugati un nou tip de bun sau serviciu

Dati optiunea: : 1

Introduceti ID-ul bunului sau serviciului: 14

Cantitate (kg/l/buc): 1

Pret: 400

Plata adaugata!

Id-ul este: 20221215000

Doriti sa adaugati o noua plata din aceasta zi? ( (D)a || N(u)):

Dacă utilizatorul dorește să caute plățile din categoria Diverse:

Cauta plata dupa:

- [01] - ID
- [02] - Data
- [03] - Tip bun/serviciu
- [04] - Nume bun/serviciu
- [05] - Pret
  
- [-1] - Inapoi

Dati optiunea: 3

Bunuri si servicii existente:

- 0: Mancare
- 1: Educatie
- 2: Sanatate
- 3: Calatorie & Transport
- 4: Imbracaminte
- 5: Facturi & Utilitati
- 6: Sport
- 7: Vacanta
- 8: Urgente
- 9: Cadouri
- 10: Renovare
- 11: Curatenie Casa
- 12: Electronice & Electrocasnice
- 13: Divertisment
- 14: Diverse
- 15: Amenzi

Introduceti tipul bunului/serviciului (nu ID-ul): Diverse

ID	Data	Nume bun/serviciu	Tip	Pret/buc	Cantitate	Pret
20221215000	15/12/2022	brad Craciun	Diverse	400	1	400
20230606003	06/06/2023	magnet frigider	Diverse	12	1	12

Pag 1 / 1

[N] Inainte | [P] Inapoi | [Q] Iesire

Dati optiunea:

Dacă utilizatorul dorește să calculeze suma de bani cheltuită într-un interval de timp, o poate face la opțiunea 5.

Calculeaza totalul platilor intre 2 date.

Introduceti data de inceput:

Introduceti data (ZZ/LL/AAAA):

Zi - (ZZ) : 2

Luna - (LL) : 12

An - (AAAA): 2022

Introduceti data finala:

Introduceti data (ZZ/LL/AAAA):

Zi - (ZZ) : 6

Luna - (LL) : 6

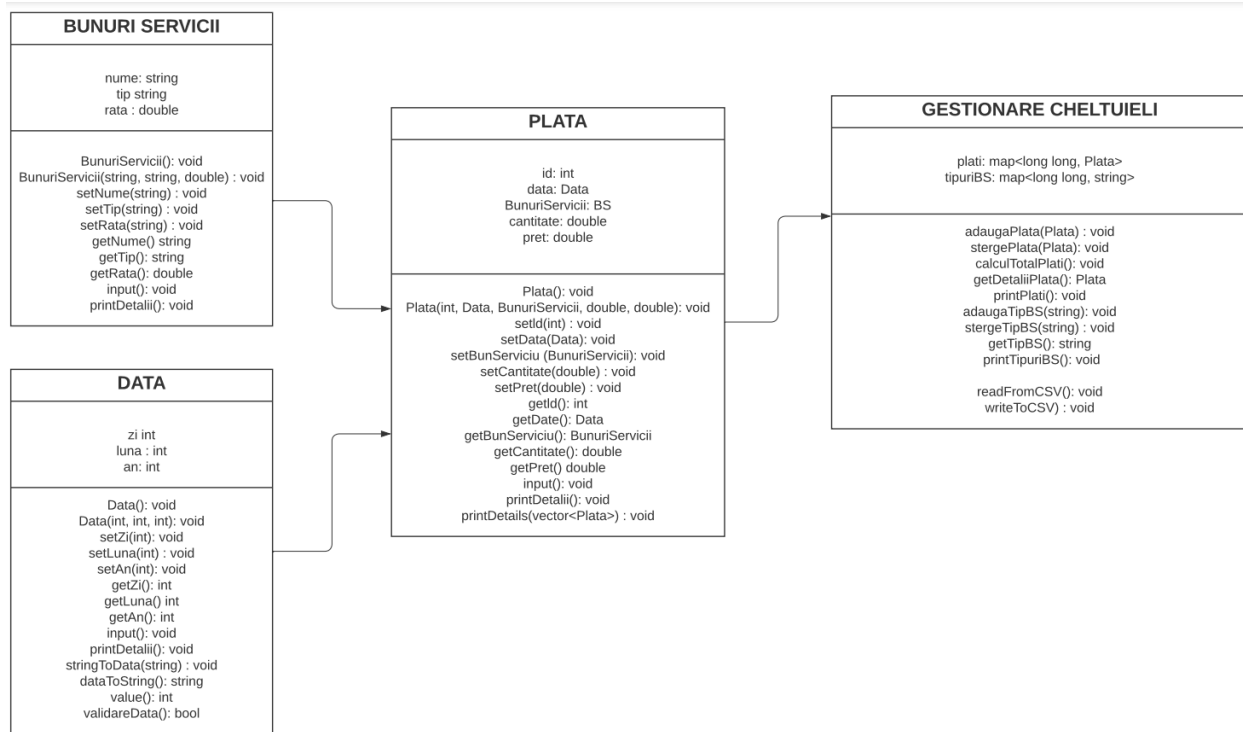
An - (AAAA): 2023

Suma totala a platilor din perioada 02/12/2022 - 06/06/2023 este 23768.5 RON.

Press any key to continue . . .

## 4. Elemente specifice POO

### 4.1. Diagrama UML a claselor



Bibliotecile standard din limbajul C++ incluse și utilizate sunt:

- ``iostream``: biblioteca standard pentru intrare/ieșire în C++.
- ``fstream``: biblioteca standard pentru citirea și scrierea fișierelor în C++.
- ``string``: biblioteca standard pentru manipularea șirurilor de caractere în C++.
- ``sstream``: biblioteca standard pentru manipularea fluxurilor de caractere în C++.
- ``ios``: biblioteca standard pentru manipularea stării de intrare/ieșire a fluxurilor în C++.
- ``limits``: biblioteca standard pentru obținerea limitelor numerice în C++.
- ``stdlib.h``: biblioteca standard pentru funcții de utilitate în C++.
- ``windows.h``: biblioteca standard pentru programare Windows în C++.
- ``conio.h``: biblioteca standard pentru funcții de intrare/ieșire în modul consolă în C++.
- ``map``: conține definiții pentru un container de asociere, care asociază valori unice cu chei.



## 4.2. Clasa Data

Clasa `Data` este utilizată pentru a reprezenta și manipula date calendaristice, cum ar fi ziua, luna și anul.

Constructorul `Data()` initializează obiectul `Data` cu valorile implicite pentru zi, lună și an (-1).

Constructorul `Data(long long zi, long long luna, long long an)` initializează obiectul `Data` cu valorile specificate pentru zi, lună și an.

Metodele `setZi()`, `setLuna()` și `setAn()` sunt utilizate pentru a seta valorile individuale pentru zi, lună și an.

Metodele `getZi()`, `getLuna()` și `getAn()` sunt utilizate pentru a obține valorile individuale pentru zi, lună și an.

Metoda `validareData()` verifică dacă data specificată este validă. Aceasta verifică dacă ziua este între 1 și 31, luna este între 1 și 12, anul este nenegativ și respectă regulile pentru anii bisecți și numărul de zile în fiecare lună.

Metoda `input()` este utilizată pentru a citi datele de la utilizator de la tastatură și le validează folosind metoda `validareData()`. Dacă data introdusă este invalidă, utilizatorul este avertizat și rugat să introducă o altă dată validă.

Metoda `printDetalii()` afișează data formatată pe consolă.

Metoda `stringToData(string data)` convertește un șir de caractere în obiectul `Data`. Aceasta descompune șirul de caractere în token-uri separate de caracterul '/' și le convertește în numere întregi.

Metoda `dataToString()` convertește obiectul `Data` într-un șir de caractere în formatul "ZZ/LL/AAAA", unde ZZ reprezintă ziua, LL reprezintă luna și AAAA reprezintă anul. Dacă ziua, luna sau anul sunt mai mici decât 10, se adaugă un "0" în fața lor și, în cazul anului, se asigură că acesta are cel puțin 4 cifre. Funcția utilizează un obiect de tip `stringstream` pentru a transforma valorile în șirul final, care este returnat cu ajutorul metodei `str()`.

Metoda `value()` returnează o valoare numerică reprezentând data sub forma AAAALLZZ, unde AAAA reprezintă anul, LL reprezintă luna și ZZ reprezintă ziua. Valoarea returnată este calculată folosind formula `an * 10000 + luna * 100 + zi`.

Destructorul `~Data()` este gol, deoarece clasa `Data` nu deține resurse dinamice care să necesite eliberare la distrugere.

### 4.3. Clasa Plata

Clasa "Plata" reprezintă o entitate care conține informații despre o plată efectuată pentru un bun sau serviciu. Aceasta are următoarele caracteristici și funcționalități:

Variabile membru private:

- id: reprezintă identificatorul unic al plății (de tip long long)
- data: obiect de tip "Data" care conține informațiile despre data efectuării plății (zi, lună, an)
- BS: obiect de tip "BunuriServicii" care reprezintă bunul sau serviciul pentru care s-a efectuat plata
- cantitate: cantitatea de bun sau serviciu plătită (de tip double)
- pret: prețul total plătit pentru cantitatea respectivă de bun sau serviciu (de tip double)

Constructori:

- Plata(): constructorul implicit care inițializează variabilele membru cu valorile implicite (-1 pentru id, Data() pentru data, BunuriServicii() pentru BS, -1 pentru cantitate și -1 pentru pret)
- Plata(long long id, Data data, BunuriServicii BS, double cantitate, double pret): constructorul care primește valorile pentru toate variabilele membru și le atribuie corespunzător

Funcții de setare :

- setId(long long id): setează valoarea id-ului plății
- setData(Data data): setează obiectul Data al plății
- setBunServiciu(BunuriServicii BS): setează obiectul BunuriServicii al plății
- setCantitate(double cantitate): setează valoarea cantității plătite
- setPret(double pret): setează valoarea prețului plății

Funcții de obținere :

- getId(): returnează id-ul plății
- getData(): returnează obiectul Data al plății
- getBunServiciu(): returnează obiectul BunuriServicii al plății
- getCantitate(): returnează cantitatea plătită
- getPret(): returnează prețul plății

Funcții pentru interacțiunea cu utilizatorul:

- input(bool \_data): primește detalii despre plata de la utilizator, inclusiv informații despre data plății dacă parametrul \_data este true
- printDetalii(): afișează detaliile plății pe ecran

- `printDetalii(vector<Plata> plati)`: afișează detaliile unei liste de plăți într-un format tabelar paginat, permițând navigarea între pagini

Destructor:

- `Plata()`: destructorul clasei (nu are nicio implementare specifică)
- 

#### 4.4. Clasa BunuriServicii

Clasa `BunuriServicii` reprezintă o entitate care descrie un bun sau serviciu în cadrul unui sistem de gestionare a cheltuielilor. Această clasă conține informații referitoare la numele, tipul și rata bunului sau serviciului.

Membrii clasei:

- ``nume``: reprezintă numele bunului sau serviciului și este de tip ``string``.
- ``tip``: reprezintă tipul bunului sau serviciului și este de tip ``string``.
- ``rata``: reprezintă rata bunului sau serviciului și este de tip ``double``.

Constructori:

- `BunuriServicii()`: inițializează membrii clasei cu valori implicite.
- `BunuriServicii(string nume, string tip, double rata)`: inițializează membrii clasei cu valorile specificate.

Metodele clasei:

- `setNume(string nume)`: setează numele bunului sau serviciului.
- `setTip(string tip)`: setează tipul bunului sau serviciului.
- `setRata(double rata)`: setează rata bunului sau serviciului.
- `getNume()`: returnează numele bunului sau serviciului.
- `getTip()`: returnează tipul bunului sau serviciului.
- `getRata()`: returnează rata bunului sau serviciului.
- `input()`: permite utilizatorului să introducă detalii despre bunul sau serviciul curent, inclusiv numele și tipul acestuia.
- `printDetalii()`: afișează detalii despre bunul sau serviciul curent, inclusiv nume, tip și rata.

De asemenea, clasa `BunuriServicii` are un destructor gol, deoarece nu există resurse alocate dinamic care necesită eliberare explicită.

## 4.5. Clasa GestionareCheltuieli

Această clasă utilizează `map` ce este un container asociativ în limbajul de programare C++. Aceasta stochează perechi cheie-valoare, unde cheile și valorile pot fi de diferite tipuri de date.

În cadrul clasei `GestionareCheltuieli`, sunt definite două variabile statice de tip `map`:

1. `map<long long, Plata> GestionareCheltuieli::plati = map<long long, Plata>();`

Această variabilă `plati` este un `map` care asociază un `long long` (număr întreg foarte mare) cu un obiect de tip `Plata`. Este utilizat pentru a stoca plățile în program, unde cheile reprezintă ID-ul unic al fiecărei plăți, iar valorile sunt obiecte de tip `Plata` care conțin informațiile corespunzătoare.

2. `map<long long, string> GestionareCheltuieli::tipuriBS = map<long long, string>();`

Această variabilă `tipuriBS` este un `map` care asociază un `long long` cu un șir de caractere (`string`). Este utilizat pentru a stoca tipurile de bunuri sau servicii disponibile în program, unde cheile reprezintă ID-ul unic al fiecărui tip, iar valorile sunt numele acestora.

Metoda `GestionareCheltuieli::adaugaPlata` are rolul de a adăuga o plată într-un container de plăți numit `plati`. În această metodă, se parcurge fiecare plată existentă în container și se verifică dacă data plății este aceeași cu data plății primite ca argument. Dacă se găsesc plăți cu aceeași dată, se numără câte sunt și se incrementează variabila `cnt`. În caz contrar, dacă se întâlnește o plată cu o dată mai mare decât cea a plății primite, bucla de parcurgere este întreruptă. După aceasta, se atribuie plății primite un ID unic calculat pe baza formulei `data \* 1000 + cnt`, iar plata este adăugată în containerul `plati` folosind acest ID. Dacă exista deja o plată cu același ID, aceasta va fi înlocuită cu plata nouă. La final, se afișează un mesaj de confirmare că plata a fost adăugată cu succes, împreună cu ID-ul plății adăugate. Astfel, prin această metodă se realizează adăugarea eficientă și gestionarea plăților în sistemul de cheltuieli.

```
void GestionareCheltuieli::adaugaPlata(Plata plata)
{
    long long cnt = 0;
    for(auto P: plati)
    {
        if(P.second.getData().value() == plata.getData().value())
            cnt++;
        else if(P.second.getData().value() > plata.getData().value())
            break;
    }
    plata.setId(plata.getData().value() * 1000 + cnt);
    plati[plata.getId()] = plata;
    cout<<"\nPlata adaugata!\n";
    cout<<"Id-ul este: "<<plata.getId()<<endl;
    return;
}
```

Metoda `GestionareCheltuieli::stergePlata` are rolul de a șterge o plată specificată din containerul de plăți `plati`. În această metodă, se obțin data și ID-ul plății care urmează să fie ștersă. Utilizând funcția `erase`, plata cu ID-ul respectiv este eliminată din container. Apoi, se parcurge fiecare plată rămasă în container și se verifică dacă data este aceeași cu cea a plății șterse și dacă ID-ul este mai mare decât ID-ul plății șterse. În caz afirmativ, se incrementează variabila `cnt`. Dacă se întâlnește o plată cu o dată mai mare decât cea a plății șterse, bucla de parcurgere este întreruptă. Urmează o buclă `for` în care plățile rămase din container, începând de la poziția `id+1`, sunt mutate în poziții anterioare pentru a umple golul creat de ștergerea plății inițiale. ID-urile acestor plăți sunt actualizate corespunzător. La finalul buclei, plata de pe poziția `cnt+id` este eliminată din container folosind funcția `erase`. Astfel, prin intermediul acestei metode, se realizează ștergerea unei plăți din sistemul de cheltuieli, asigurându-se corectitudinea ID-urilor și actualizarea adecvată a containerului `plati`.

```
void GestionareCheltuieli::stergePlata(Plata plata)
{
    Data data = plata.getData();
    long long id = plata.getId();
    plati.erase(plata.getId());
    cout<<"Plata cu ID-ul "<<plata.getId()<<" a fost stearsa!\n";

    long long cnt = 0;
    for(auto P: plati)
    {
        if (P.second.getData().value() == data.value() && P.second.getId() > id)
            cnt++;
        else if (P.second.getData().value() > data.value())
            break;
    }
    for(long long i = 0; i < cnt; i++)
        plati[i+id] = plati[i+id+1], plati[i+id].setId(i+id);
    plati.erase(cnt+id);
    return;
}
```

Metoda `GestionareCheltuieli::calculTotalPlati()` calculează suma totală a plăților efectuate într-o anumită perioadă de timp. Utilizatorul este întâmpinat cu un mesaj pentru a introduce data de început și data finală a perioadei. Apoi, pentru fiecare plată din colecția `plati`, se verifică dacă data plății se încadrează în intervalul specificat. Dacă este cazul, valoarea plății este adăugată la variabila `total`. La final, este afișată suma totală a plăților efectuate în perioada specificată, împreună cu datele de început și de final.

```
void GestionareCheltuieli::calculTotalPlati()
{
    int choice;
    cout<<"Calculaaza totalul platilor intre 2 date.\n";
    Data startDate, endDate;
    cout << "\nIntroduceți data de început:\n";
    startDate.input();
    cout << "\nIntroduceți data finala:\n";
    endDate.input();
    double total = 0;
    for (auto plata : plati)
        if (plata.second.getData().value() >= startDate.value() && plata.second.getData().value() <= endDate.value())
            total += plata.second.getPret();
    cout << "\nSuma totala a platilor din perioada " << startDate.dataToString() << " - " << endDate.dataToString() << " este " << total << " RON."<< endl;
}
```

Metoda `GestionareCheltuieli::getDetaliiPlata` are rolul de a permite utilizatorului să caute și să selecteze o plată specifică din sistemul de cheltuieli. În cadrul acestei metode, utilizatorul este întâmpinat cu un meniu interactiv care oferă opțiuni de căutare după diferite criterii, cum ar fi ID-ul, data, tipul bunului/serviciului, numele bunului/serviciului și prețul. Utilizatorul selectează o opțiune și introduce valorile corespunzătoare criteriului de căutare. Apoi, se realizează căutarea în containerul `plati` pentru a identifica plățile care se potrivesc cu criteriul specificat. Dacă se găsesc plăți care îndeplinesc criteriul, acestea sunt afișate utilizatorului, iar acesta are posibilitatea de a selecta o plată specifică prin intermediul ID-ului. Dacă nu se găsesc plăți care să corespundă criteriului, utilizatorul este informat în consecință și i se oferă opțiunea de a relua căutarea sau de a reveni la meniul anterior. La final, se returnează plata selectată sau un obiect de plată vid în cazul în care utilizatorul a ales să se întoarcă la meniul anterior sau nu a selectat nicio plată. Astfel, această metodă facilitează interacțiunea utilizatorului cu sistemul de cheltuieli, permițându-i să găsească și să acceseze plățile dorite în funcție de criteriile specificate.

Spre exemplu, logica funcției de *căutare* în raportată la numele plății este:

```
{
    string numeBSReq;
    cout << "Introduceți numele bunului sau serviciului: ";
    cin >> numeBSReq;
    vector<Plata> matchingExpenses;
    for (auto plata : plati)
        if (plata.second.getBunServiciu().getNum().find(numeBSReq) != string::npos)
            matchingExpenses.push_back(plata.second);
    if (matchingExpenses.size() == 0)
    {
        cout << "\nNu au fost gasite plati cu acest nume " << numeBSReq << ".\n"
            << endl;
        cout << "\n\nDoriti sa incercati din nou? ((D)a || N(u)) : ";
        char opt;
        cin >> opt;
        while (opt != 'D' && opt != 'N')
        {
            cout << "Introduceți o optiune valida: ";
            cin >> opt;
        }
        return (opt == 'D' || opt == 'd') ? getDetaliiPlata() : Plata();
    }
    else
    {
        cout << matchingExpenses.size() << " au fost gasite urmatoarele plati cu acest nume: " << numeBSReq << ".\n"
            << endl;
        Plata P = Plata();
        Plata::printDetalii(matchingExpenses);
        char opt = 'D';
        while (opt == 'D')
        {
            cout << "\nIntroduceți ID-ul platii: ";
            long long idReq;
            cin >> idReq;
            for (auto plata : matchingExpenses)
                if (plata.getId() == idReq)
                    return plata;
            cout << "Nicio plata din lista nu are acest ID " << idReq << ".\n"
                << endl;
            cout << "\n\nDoriti sa incercati din nou? ((D)a || N(u)) : ";
            char opt;
            cin >> opt;
            while (opt != 'D' && opt != 'N')
            {
                cout << "Introduceți o optiune valida: ";
                cin >> opt;
            }
        }
    }
}
```

1. Se solicită utilizatorului să introducă numele bunului sau serviciului căutat prin intermediul comenzii `cout << "Introduceti numele bunului sau serviciului: ";` și se stochează în variabila `numeBSReq`.
2. Se declară un vector gol `matchingExpenses` în care se vor stoca plățile care corespund criteriului de căutare.
3. Se parcurge fiecare element (`plata`) din lista `plati` utilizând un `for` loop.
4. În fiecare iterație a buclei, se verifică dacă numele bunului sau serviciului din plățile (`plata.second`) conține șirul introdus (`numeBSReq`). Această verificare se realizează utilizând funcția `find()` în combinație cu `string::npos` care indică că șirul nu a fost găsit. Dacă numele conține șirul introdus, plata respectivă este adăugată în vectorul `matchingExpenses` utilizând `push_back()`. S-a decis utilizarea funcției `find` deoarece utilizatorul poate nu își amintește numele întreg al plății, și introduce doar o parte din nume (spre exemplu, introduce doar "brad" din plata "brad Craciun"). Această funcție permite căutarea subșirurilor identice din șir.
5. După parcurgerea tuturor plăților, se verifică dacă vectorul `matchingExpenses` este gol folosind `matchingExpenses.size() == 0`. Dacă este gol, se afișează un mesaj corespunzător și se oferă utilizatorului opțiunea de a încerca din nou sau de a ieși din funcție.
6. Dacă vectorul `matchingExpenses` nu este gol, se afișează numărul de plăți găsite și apoi se afișează detaliile acestora utilizând funcția `printDetalii()` din clasa `Plata`.
7. Se intră într-un ciclu `while` în care utilizatorului i se solicită să introducă ID-ul plății dorite. Se verifică dacă ID-ul introdus se potrivește cu vreuna dintre plăți din vectorul `matchingExpenses`. Dacă da, se returnează plata corespunzătoare și se încheie funcția.
8. Dacă nicio plată nu are ID-ul cerut, se afișează un mesaj corespunzător și utilizatorului i se oferă opțiunea de a încerca din nou sau de a ieși din funcție.
9. Funcția se repetă în modul ciclic până când se returnează o plată sau utilizatorul alege să iasă din funcție.

Metoda `GestionareCheltuieli::adaugaTipBS` are rolul de a adăuga un tip nou de bun/serviciu în sistemul de gestionare a cheltuielilor. În implementarea metodei, se parcurge containerul `tipuriBS` care conține tipurile de bunuri/servicii existente. Pentru fiecare element din acest container, se verifică dacă valoarea (tipul bunului/serviciului) coincide cu tipul pe care dorim să-l adăugăm. Dacă se găsește o coincidență, metoda se încheie și nu se adaugă nimic. În caz contrar, tipul nou de bun/serviciu este adăugat în container, cu un nou index generat automat (`tipuriBS.size()`), iar metoda se încheie. Astfel, această metodă asigură că nu se adaugă duplicate de tipuri de bunuri/servicii în sistem și permite extinderea listei de tipuri disponibile pentru utilizatorul sistemului de gestionare a cheltuielilor.

```
void GestionareCheltuieli::adaugaTipBS(string tip)
{
    for(auto tipBS : tipuriBS)
        if(tipBS.second == tip)
            return;
    tipuriBS[tipuriBS.size()] = tip;
    return;
}
```

Metoda `GestionareCheltuieli::stergeTipBS` are rolul de a șterge un tip de bun/serviciu din sistemul de gestionare a cheltuielilor. În implementarea metodei, se parcurge containerul `tipuriBS` care conține tipurile de bunuri/servicii existente. Pentru fiecare element din acest container, se verifică dacă valoarea (tipul bunului/serviciului) coincide cu tipul pe care dorim să-l ștergem. Dacă se găsește o coincidență, se efectuează o reorganizare a containerului prin deplasarea elementelor în stânga, pentru a elimina tipul respectiv. La final, se elimină ultimul element din container, iar metoda se încheie. Astfel, această metodă permite eliminarea unui tip de bun/serviciu din lista disponibilă în sistemul de gestionare a cheltuielilor.

```
void GestionareCheltuieli::stergeTipBS(string tip)
{
    for(long long i=0; i<tipuriBS.size(); i++)
        if(tipuriBS[i] == tip)
        {
            for(long long j=i; j<tipuriBS.size() - 1; j++)
                tipuriBS[j] = tipuriBS[j+1];
            tipuriBS.erase(tipuriBS.size()-1);
            return;
        }
    return;
}
```



Metoda `GestionareCheltuieli::printTipuriBS` are rolul de a afișa lista de bunuri și servicii existente în sistemul de gestionare a cheltuielilor. În implementarea sa, se parcurge containerul `tipuriBS`, care conține tipurile de bunuri și servicii în perechi cheie-valoare. Pentru fiecare element din acest container, se afișează în mod formatat perechea cheie și valoare, folosind funcția `setw` pentru a alinia coloanele în mod corespunzător. În special, se utilizează `setw(log10((int)(tipuriBS.size() > 1) ? tipuriBS.size() - 1 : 1))` pentru a calcula lățimea necesară coloanei cheie, astfel încât afișarea să fie uniformă indiferent de numărul de elemente din container. Lungimea maximă este determinată de `log10((int)(tipuriBS.size() > 1) ? tipuriBS.size() - 1 : 1)`, care este logaritmul în baza 10 al valorii maxime dintre `tipuriBS.size() - 1` și 1. Aceasta asigură că toate indexurile vor fi aliniate corect în coloană. La final, se afișează o linie goală pentru a delimita în mod clar lista de bunuri și servicii. Prin urmare, această metodă permite afișarea clară și ordonată a tipurilor de bunuri și servicii disponibile în sistemul de gestionare a cheltuielilor.

```
void GestionareCheltuieli::printTipuriBS()
{
    cout << "\nBunuri si servicii existente:\n\n";
    for (auto tipBS : tipuriBS)
        cout << left << setw(log10((int)(tipuriBS.size() > 1) ? tipuriBS.size() - 1 : 1)) << tipBS.first,
            cout << " : " << tipBS.second << endl;
    cout << endl;
    return;
}
```

Metoda `GestionareCheltuieli::getTipBS` are rolul de a obține tipul unui bun sau serviciu pe baza unui ID introdus de utilizator. În primul rând, se cere utilizatorului să introducă un ID prin intermediul comenzii `cout << "\nIntroduceti ID-ul bunului sau serviciului: ";`, iar această valoare este stocată în variabila `id` utilizând `cin >> id`. Apoi, se verifică validitatea ID-ului introdus prin intermediul unui ciclu `while`, care se execută atât timp cât ID-ul este mai mic decât 0 sau mai mare sau egal cu dimensiunea listei `tipuriBS`. În cazul în care ID-ul introdus este invalid, se afișează un mesaj de eroare și utilizatorului i se cere să introducă un ID valid. După ce utilizatorul introduce un ID valid, funcția returnează tipul bunului sau serviciului asociat acelui ID din lista `tipuriBS` utilizând `return tipuriBS[id]`.

```
string GestionareCheltuieli::getTipBS()
{
    long long id;
    cout << "\nIntroduceti ID-ul bunului sau serviciului: ";
    cin >> id;
    while (id < 0 || id >= tipuriBS.size())
    {
        cout << "ID invalid. Incercati din nou: ";
        cin >> id;
    }
    return tipuriBS[id];
}
```

## 4.6. Lucrul cu fișiere

Metoda `readFromCSV()` are rolul de a citi datele din două fișiere CSV și de a le stoca în structurile de date corespunzătoare ale clasei `GestionareCheltuieli`. În prima parte a metodei, fișierul `"Plati.csv"` este deschis și se citesc liniile corespunzătoare informațiilor despre plăți. Fiecare linie este împărțită în token-uri utilizând `'stringstream'`, iar acestea sunt convertite la tipurile de date adecvate. Apoi, un obiect `'Plata'` este creat cu valorile extrase și adăugat în map-ul `'plati'`. După încheierea citirii din acest fișier, se trece la deschiderea și citirea fișierului `"BunuriServicii.csv"`. Aici se extrage numele tipului de bun sau serviciu din fiecare linie și este adăugat în map-ul `'tipuriBS'`. În final, toate datele au fost citite și stocate corespunzător, iar fișierele sunt închise. Această metodă asigură preluarea datelor din fișierele CSV și inițializarea structurilor de date necesare pentru gestionarea cheltuielilor.

```
void GestionareCheltuieli::readFromCSV()
{
    ifstream file;
    file.open("./data/Plati.csv");
    string line;
    getline(file, line); // sare peste prima linie
    while (getline(file, line))
    {
        stringstream ss(line);
        string token;
        long long id;
        Data data;
        BunuriServicii BS;
        double cantitate;
        double pret;
        getline(ss, token, ',');
        id = stoll(token);
        getline(ss, token, ',');
        data.stringToData(token);
        getline(ss, token, ',');
        BS.setNume(token);
        getline(ss, token, ',');
        BS.setTip(token);
        getline(ss, token, ',');
        BS.setRata(stod(token));
        getline(ss, token, ',');
        cantitate = stod(token);
        getline(ss, token, ',');
        pret = stod(token);
        plati[id] = Plata(id, data, BS, cantitate, pret);
    }
    file.close();
    file.open("./data/BunuriServicii.csv");
    getline(file, line); // skip first line
    while (getline(file, line))
    {
        stringstream ss(line);
        string token;
        getline(ss, token, ',');
        getline(ss, token, ',');
        tipuriBS[tipuriBS.size()] = token;
    }
    return;
}
```

Metoda `writeToCSV()` are rolul de a scrie datele din structurile de date ale clasei `GestionareCheltuieli` în fișierele CSV corespunzătoare. În prima parte a metodei, se deschide și se scrie în fișierul temporar "temp.csv" antetul cu titlurile coloanelor. Apoi, pentru fiecare obiect de tip `Plata` din map-ul `plati`, se scriu valorile aferente fiecărei coloane în fișier. După încheierea scrierii în acest fișier, se închide și se înlătură fișierul "Plati.csv" și se redenumeste fișierul temporar la numele corect. În partea a doua a metodei, se deschide și se scrie în fișierul temporar "temp.csv" antetul pentru tipurile de bunuri și servicii. Apoi, pentru fiecare pereche din map-ul `tipuriBS`, se scriu valorile corespunzătoare în fișier. După încheierea scrierii în acest fișier, se închide și se înlătură fișierul "BunuriServicii.csv" și se redenumeste fișierul temporar la numele corect.

```
void GestionareCheltuieli::writeToCSV()
{
    ofstream file;
    file.open("./data/temp.csv");
    file << "ID,Data,Nume bun/serviciu,Tip bun/serviciu,Rata,Cantitate,Pret\n";
    for (auto plata : plati)
    {
        file << plata.first << ",";
        file << plata.second.getData().dataToString() << ",";
        file << plata.second.getBunServiciu().getNum() << ",";
        file << plata.second.getBunServiciu().getTip() << ",";
        file << plata.second.getBunServiciu().getRata() << ",";
        file << plata.second.getCantitate() << ",";
        file << plata.second.getPret() << "\n";
    }
    file.close();
    remove("./data/Plati.csv");
    rename("./data/temp.csv", "./data/Plati.csv");

    file.open("./data/temp.csv");
    file << "S.No.,Bunuri/Servicii\n";
    for (auto tipBS : tipuriBS)
    {
        file << tipBS.first << ",";
        file << tipBS.second << "\n";
    }
    file.close();
    remove("./data/BunuriServicii.csv");
    rename("./data/temp.csv", "./data/BunuriServicii.csv");

    return;
}
```

Programul vine cu o serie de bunuri și servicii pe care utilizatorul o poate modifica, fie din aplicație, fie accesând fișierul csv.

```
Bunuri si servicii existente:

0: Mancare
1: Educatie
2: Sanatate
3: Calatorie & Transport
4: Imbracaminte
5: Facturi & Utilitati
6: Sport
7: Vacanta
8: Urgente
9: Cadouri
10: Renovare
11: Curatenie Casa
12: Electronice & Electrocasnice
13: Divertisment
14: Diverse

Press any key to continue . . .
```

Afișarea plătilor efectuate se realizează sub formă tabelară, fiind printate câte 15 plăți pe pagină. Utilizatorul va urma instrucțiunile meniului din partea de jos pentru a naviga între pagini.

ID	Data	Nume bun/serviciu	Tip	Pret/buc	Cantitate	Pret
221107000	07/11/0022	menu KFC	Mancare	40	3	120
221107001	07/11/0022	sacou	Imbracaminte	220	1	220
20220606000	06/06/2022	tranzistoare pnp	Diverse	0.06	1e+07	600000
20221103000	03/11/2022	bratara fitness	Electronice & Electrocasnice	205	1	205
20221103001	03/11/2022	termostat	Electronice & Electrocasnice	119.99	1	119.99
20221104000	04/11/2022	aspirator	Electronice & Electrocasnice	389	1	389
20221105000	05/11/2022	mixer	Electronice & Electrocasnice	169	1	169
20221106000	06/11/2022	prajitor	Electronice & Electrocasnice	200	2	400
20221107000	07/11/2022	lenjerie pat	Curatenie Casa	120	1	120
20221108000	08/11/2022	covor	Renovare	366.667	3	1100
20221108001	08/11/2022	perne	Renovare	125	4	500
20221110000	10/11/2022	masa calcat	Renovare	300	1	300
20221110001	10/11/2022	uscator	Curatenie Casa	130	1	130
20221110002	10/11/2022	cos rufe	Curatenie Casa	155	2	310
20221112000	12/11/2022	pomi fructiferi	Diverse	50	5	250

Pag 1 / 5  
 [N] Inainte | [P] Inapoi | [Q] Iesire  
 Dati optiunea:

ID	Data	Nume bun/serviciu	Tip	Pret/buc	Cantitate	Pret
20221112001	12/11/2022	amenda viteza	Amenzi	800	1	800
20221114000	14/11/2022	gratar	Renovare	390	1	390
20221114001	14/11/2022	factura internet&TV	Facturi & Utilitati	170	1	170
20221114002	14/11/2022	jucarii	Cadouri	180	5	900
20221115000	15/11/2022	vitamine	Sanatate	67	1	67
20221115001	15/11/2022	separator medicamente	Sanatate	70	1	70
20221116000	16/11/2022	cafea	Mancare	25	2	50
20221116001	16/11/2022	parfum	Cadouri	300	1	300
20221117000	17/11/2022	rucsac	Sport	99	1	99
20221117001	17/11/2022	deodorant	Curatenie Casa	20	3	60
20221120000	20/11/2022	oglinza	Renovare	349.5	2	699
20221121000	21/11/2022	cuier	Renovare	700	1	700
20221125000	25/11/2022	pantofar	Renovare	300	1	300
20221125001	25/11/2022	becuri LED	Renovare	33.3333	6	200
20221213000	13/12/2022	fierbator	Electronice&Electrocasnice	210	1	210

Pag 2 / 5

[N] Inainte | [P] Inapoi | [Q] Iesire

Dati optiunea:

ID	Data	Nume bun/serviciu	Tip	Pret/buc	Cantitate	Pret
20221213001	13/12/2022	uscator de par	Electronice&Electrocasnice	140	1	140
20221215000	15/12/2022	brad Craciun	Diverse	400	1	400
20230301000	01/03/2023	vopsea	Renovare	90	5	450
20230302000	02/03/2023	cort	Sport	600	1	600
20230302001	02/03/2023	role	Sport	210	1	210
20230302002	02/03/2023	masca de protectie	Sport	80	1	80
20230312000	12/03/2023	covrigi	Mancare	4	5	20
20230320000	20/03/2023	apa	Mancare	2.7	10	27
20230320001	20/03/2023	ciocolata	Mancare	32.5	2	65
20230321000	21/03/2023	abonament TPL	Calatorie&Transport	100	1	100
20230321001	21/03/2023	Nurofen	Sanatate	50	1	50
20230321002	21/03/2023	Paracetamol	Sanatate	6.5	1	6.5
20230501000	01/05/2023	abonament TPL	Calatorie&Transport	100	1	100
20230501001	01/05/2023	hanorac	Imbracaminte	150	1	150
20230501002	01/05/2023	blugi	Imbracaminte	100	1	100

Pag 3 / 5

[N] Inainte | [P] Inapoi | [Q] Iesire

Dati optiunea:

ID	Data	Nume bun/serviciu	Tip	Pret/buc	Cantitate	Pret
20230503000	03/05/2023	factura E.on energie electrica	Facturi&Utilitati	488	1	488
20230503001	03/05/2023	factura E.on gaze	Facturi&Utilitati	234	1	234
20230511000	11/05/2023	flori	Cadouri	60	1	60
20230512000	12/05/2023	tricou	Imbracaminte	57.5	2	115
20230520000	20/05/2023	pix	Educatie	3	6	18
20230520001	20/05/2023	obiecte papetarie	Educatie	8.16667	6	49
20230527000	27/05/2023	detergent rufe	Curatenie Casa	14	5	70
20230527001	27/05/2023	detergent vase	Curatenie Casa	12	1	12
20230601000	01/06/2023	telefon	Electronice&Electrocasnice	6000	1	6000
20230603000	03/06/2023	placinta	Mancare	4	2	8
20230605000	05/06/2023	periuta electrica	Electronice&Electrocasnice	400	1	400
20230605001	05/06/2023	baterie externa	Electronice&Electrocasnice	250	1	250
20230605002	05/06/2023	geaca	Imbracaminte	600	1	600
20230605003	05/06/2023	cafea	Mancare	25.6667	3	77
20230605004	05/06/2023	ciment	Renovare	2.5	400	1000

Pag 4 / 5

[N] Inainte | [P] Inapoi | [Q] Iesire

Dati optiunea:

ID	Data	Nume bun/serviciu	Tip	Pret/buc	Cantitate	Pret
20230605005	05/06/2023	covor	Curatenie Casa	570	1	570
20230605006	05/06/2023	parchet	Renovare	85	20	1700
20230605007	05/06/2023	sejur Grecia	Vacanta	4500	2	9000
20230605008	05/06/2023	pasta de dinti	Sanatate	22	1	22
20230605009	05/06/2023	cablu USB	Electronice & Electrocasnice	70	1	70
20230606000	06/06/2023	ciocolata	Mancare	4.5	20	90
20230606001	06/06/2023	capsuni	Mancare	20	3	60
20230606002	06/06/2023	vitamine	Sanatate	45	1	45
20230606003	06/06/2023	magnet frigider	Diverse	12	1	12
20230606004	06/06/2023	pijama	Imbracaminte	110	1	110

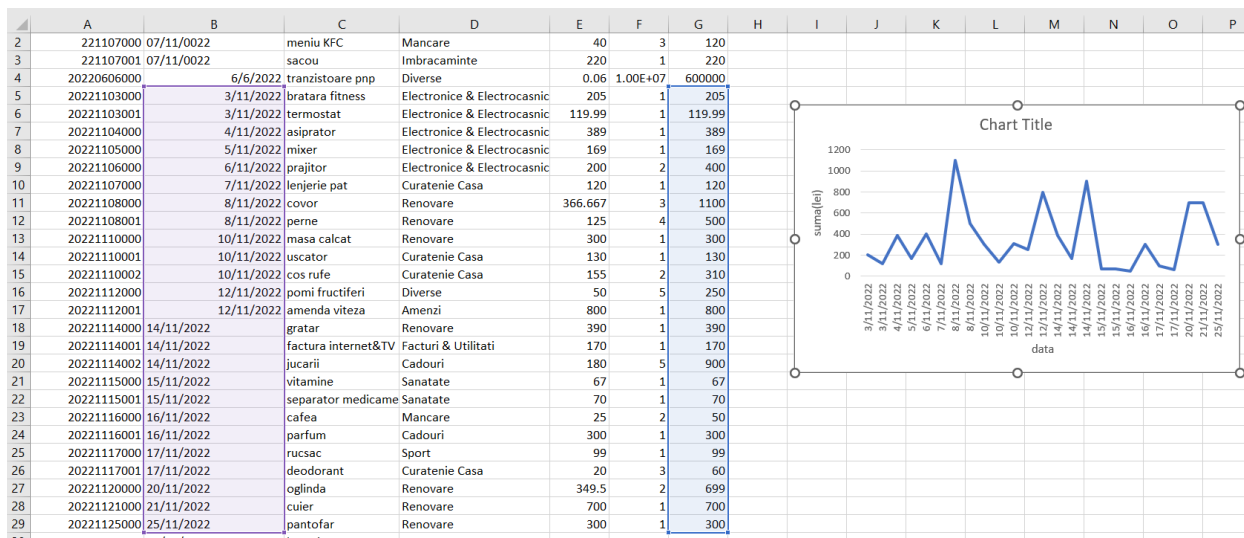
Pag 5 / 5  
 [N] Inainte | [P] Inapoi | [Q] Iesire  
 Dati optiunea:

Fișierul csv rezultat la ieșirea din program:

	A	B	C	D	E	F	G
1	ID	Data	Nume bun/serviciu	Tip bun/serviciu	Rata	Cantitate	Pret
2	221107000	07/11/0022	menu KFC	Mancare	40	3	120
3	221107001	07/11/0022	sacou	Imbracaminte	220	1	220
4	20220606000	6/6/2022	tranzistoare pnp	Diverse	0.06	1.00E+07	600000
5	20221103000	3/11/2022	bratară fitness	Electronice & Electrocasnic	205	1	205
6	20221103001	3/11/2022	termostat	Electronice & Electrocasnic	119.99	1	119.99
7	20221104000	4/11/2022	aspirator	Electronice & Electrocasnic	389	1	389
8	20221105000	5/11/2022	mixer	Electronice & Electrocasnic	169	1	169
9	20221106000	6/11/2022	prajitor	Electronice & Electrocasnic	200	2	400
10	20221107000	7/11/2022	lenjerie pat	Curatenie Casa	120	1	120
11	20221108000	8/11/2022	covor	Renovare	366.667	3	1100
12	20221108001	8/11/2022	perne	Renovare	125	4	500
13	20221110000	10/11/2022	masa calcat	Renovare	300	1	300
14	20221110001	10/11/2022	uscator	Curatenie Casa	130	1	130
15	20221110002	10/11/2022	cos rufe	Curatenie Casa	155	2	310
16	20221112000	12/11/2022	pomi fructiferi	Diverse	50	5	250
17	20221112001	12/11/2022	amenda viteza	Amenzi	800	1	800
18	20221114000	14/11/2022	gratar	Renovare	390	1	390
19	20221114001	14/11/2022	factura internet&TV	Facturi & Utilitati	170	1	170
20	20221114002	14/11/2022	jucarii	Cadouri	180	5	900
21	20221115000	15/11/2022	vitamine	Sanatate	67	1	67
22	20221115001	15/11/2022	separator medicame	Sanatate	70	1	70
23	20221116000	16/11/2022	cafea	Mancare	25	2	50
24	20221116001	16/11/2022	parfum	Cadouri	300	1	300
25	20221117000	17/11/2022	rucsac	Sport	99	1	99
26	20221117001	17/11/2022	deodorant	Curatenie Casa	20	3	60
27	20221120000	20/11/2022	oglină	Renovare	349.5	2	699
28	20221121000	21/11/2022	cuier	Renovare	700	1	700
29	20221125000	25/11/2022	pantofar	Renovare	300	1	300

Plati (+)

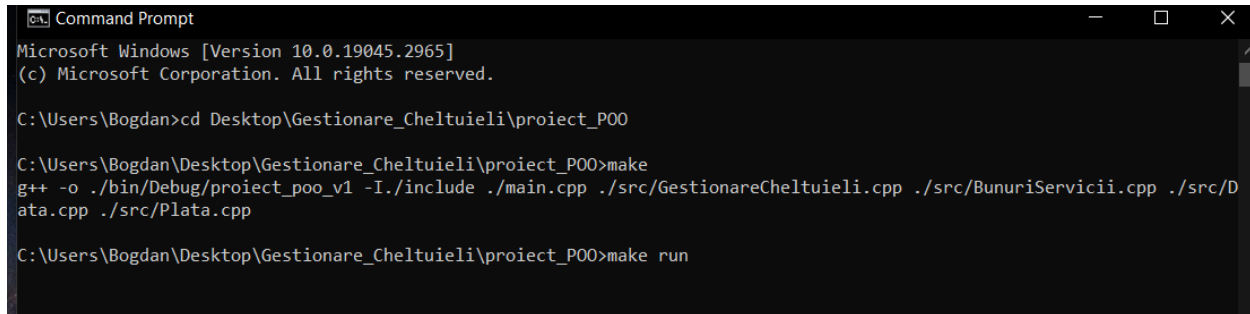
Utilizatorul poate folosi opțiunile din excel pentru a crea grafice în funcție de datele salvate. Spre exemplu evoluția cheltuielilor din luna noiembrie 2022 este următoare:



## 5. Informatii despre rularea proiectului

Pentru rularea proiectului din Command Prompt cu ajutorul Makefile se vor da urmatoarele comenzi si se vor urmari instructiunile din fisierul ReadMe de la adresa

<https://github.com/BogdanTibuleac/Proiect-POO> :



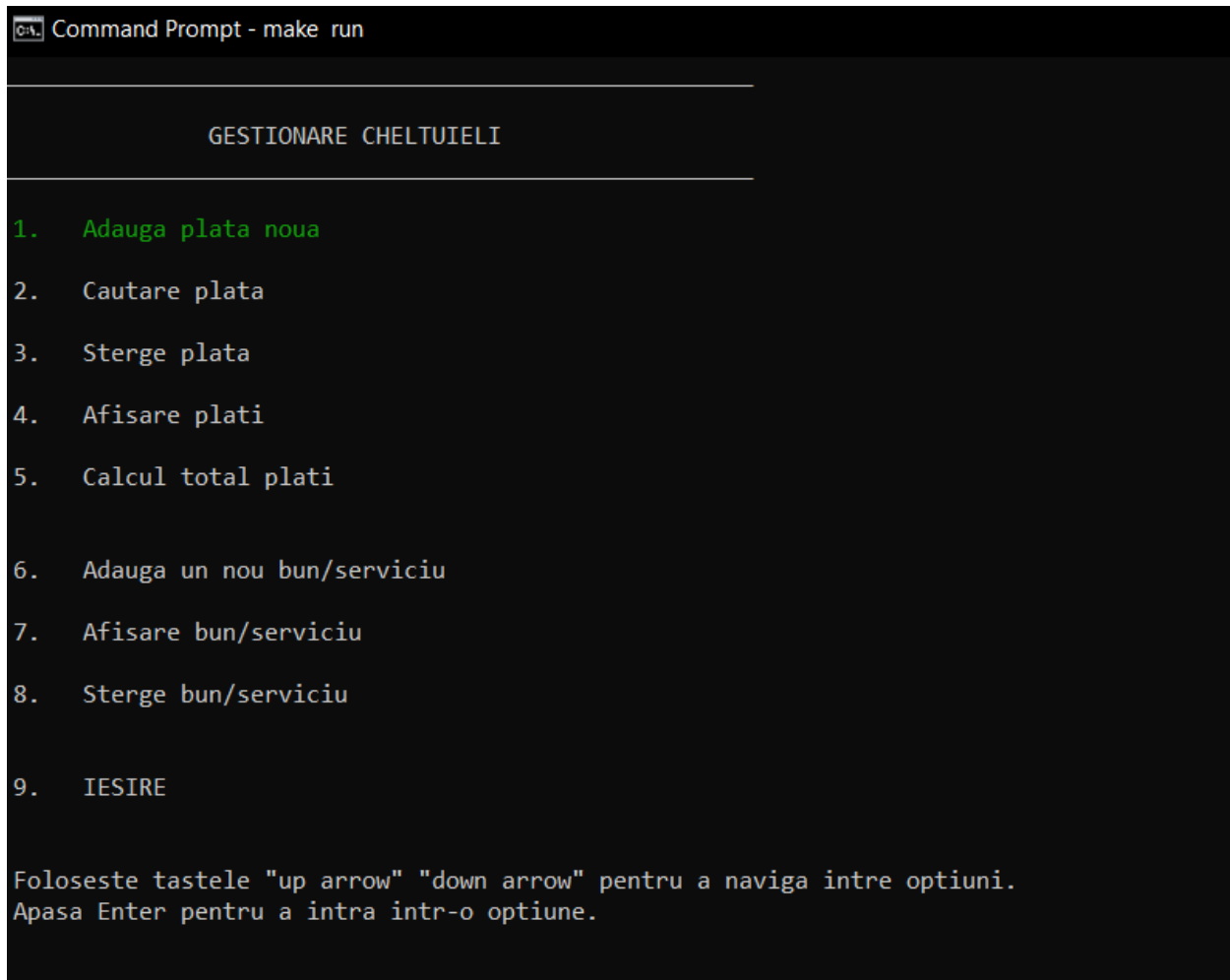
```
Command Prompt
Microsoft Windows [Version 10.0.19045.2965]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Bogdan>cd Desktop\Gestionare_Cheltuieli\proiect_P00

C:\Users\Bogdan\Desktop\Gestionare_Cheltuieli\proiect_P00>make
g++ -o ./bin/Debug/proiect_poo_v1 -I./include ./main.cpp ./src/GestionareCheltuieli.cpp ./src/BunuriServicii.cpp ./src/D
ata.cpp ./src/Plata.cpp

C:\Users\Bogdan\Desktop\Gestionare_Cheltuieli\proiect_P00>make run
```

Rezultatul este acelasi ca si in cazul rularii cu Codeblocks:



```
Command Prompt - make run

GESTIONARE CHELTUIELI

1.  Adauga plata noua
2.  Cautare plata
3.  Sterge plata
4.  Afisare plati
5.  Calcul total plati
6.  Adauga un nou bun/serviciu
7.  Afisare bun/serviciu
8.  Sterge bun/serviciu
9.  IESIRE

Foloseste tastele "up arrow" "down arrow" pentru a naviga intre optiuni.
Apasa Enter pentru a intra intr-o optiune.
```



## 6. Concluzii

După dezvoltarea acestui proiect, am ajuns la concluzia că este deosebit de important să stabilesc din start modul în care doresc să arate tipurile de date utilizate în aplicație. Dacă nu fac această alegere corectă de la început, există riscul să fiu nevoit să refac întreaga aplicație. De aceea, în viitor, voi planifica cu mare atenție aspectul final al aplicației înainte de a începe partea de implementare, pentru a evita pierderea timpului în rescrierea codului. Pe parcursul acestui proiect, am învățat lucruri utile și mi-am reamintit multe noțiuni de bază, pe care, oricât de bine aș fi crezut că le stăpânesc din punct de vedere teoretic, punerea în practică a dus la unele dificultăți. A fost plăcut să lucrez din nou cu elemente învățate în primul semestru la C, spre exemplu prelucrarea șirurilor de caractere, adaptate la paradigma C++. Am învățat să folosesc elemente noi, precum map-urile, care sunt utilizate des în practică, mi-am reamintit cum se pot folosi ghilimele în afișarea mesajelor cu cout, cum se poate naviga între opțiuni utilizând nu doar opțiunile prin intermediul unui număr, ci și cu ajutorul tastelor arrow up, arrow down, și nu în ultimul rând, cum să lucrez cu fișiere.

Pe viitor, aplicația ar trebui îmbunătățită din punct de vedere al tratării excepțiilor, deoarece există unele bug-uri dacă datele de intrare nu sunt de tipul corect. Utilă ar fi introducerea și ID-ului bunului sau serviciului în ID-ul plății pentru a nu crea confuzii cu tipurile diferite, dar efectuate în aceeași dată. Pentru utilizarea aplicației de către o persoană, cum de altfel a fost și creată, aplicația nu are probleme din punctul de vedere al memoriei. În schimb, dacă ar fi utilizată de către o firmă, într-o zi pot fi efectuate 1000 de plăți. Dacă se depășește acest număr, ar crea un bug.

În plus, ar fi utilă o interfață user-friendly, cu butoane, unde utilizatorul nu trebuie să folosească la fiecare pas tastatura. De asemenea, o interfață de autentificare a utilizatorului ar fi o idee bună dacă ne gândim la securitatea plăților efectuate. Nu este de dorit ca persoane necunoascute care intră în posesia aplicației să acceseze plățile. Acest lucru ar presupune și salvarea datelor într-o bază de date și nu în fișiere excel.

## 7. Bibliografie

<https://www.wellybox.com/blog/how-to-use-an-expense-manager-app/>

<https://cplusplus.com/>

<https://www.scaler.com/topics/string-size-in-cpp/>

<https://www.geeksforgeeks.org/map-associative-containers-the-c-standard-template-library-stl/>

<https://www.geeksforgeeks.org/stringstream-c-applications/>

<https://www.lucidchart.com/pages/uml-class-diagram>

<dev.fiesc.usv.ro/remus/>

<http://apollo.eed.usv.ro/~laura.bilius/>

<https://www.youtube.com/watch?v=rzd9AEfjCxU>

## 8. Caiet de sarcini

DATA	OBSERVATII
15.03.2023	Cautare modele de aplicatii Schitare notiuni generale despre proiect (cate clase va avea, catre cine este indreptat)
27.03.2023	Implementare clase de baza si testarea functionalitatii prin creare unor obiecte
30.30.2023	Schitarea diagramei UML a claselor
08.04.2023	Crearea mai multor metode a clasei GestionareCheltuieli Incercare utilizare fisiere
01.06.2023	Rezolvare problema fisiere + implementare Adaugare functionalitate de a calcula totalul platilor Modificare meniu + implementare navigare cu taste
3.06.2023	Adaugare vizualizare tabelara cu numar fix de plati afisate
4.06.2023	Modificare fisiere csv
06.06.2023	Incarcare proiect pe github Creare makefile + instalare Chocolatey pentru rularea proiectului din Command Prompt Retusuri documentatie + concluzii