ÎNCHIRIERE MAȘINI

Coroamă Betuela-Mădălina

Anul I, grupa 3114

Facultatea de Inginerie Electrică și Știința Calculatoarelor, USV

Proiect de semestru realizat la disciplina PCLP II

Suceava, mai 2020

Abstract

În cadrul acestui proiect este prezentată o aplicație care oferă atât posibilitatea închirierii unui autoturism din cadrul unei liste de mașini, cât și șansa de a gestiona actualizarea acesteia.Proiectul a fost realizat utilizând TDA Listă înlănțuită și poartă denumirea de "*Închiriere Mașini*".

Aplicația reprezintă un meniu cu o formă simplă prin care, inițial se cere autentificarea persoanei care accesează aplicația.Dacă individul este administratorul ,trebuie să introducă o parolă care îi va permite accesul la gestionarea listei de mașini.Acesta va putea vizualiza lista,adăuga componente în interiorul ei sau șterge date din aceasta.În cazul în care utilizatorul este client, se va afișa lista din care poate ,sau nu, să închirieze autoturismul.Dacă acesta decide să opteze pentru varianta afirmativă, datele personale îi vor fi preluate,urmând ca cererea clientului să fie înregistrată.

Lista de mașini este salvată într-un fișier text pentru o stocare cât mai ordonată a datelor.

Keywords: TDA,site,fișier,date

ÎNCHIRIERE MAȘINI

Această aplicație a fost creată pentru persoanele care își doresc o modalitate cât mai simplă și rapidă de vizualizare a unor mașini disponibile pentru a fi închiriate , într-un timp foarte scurt.Astfel prin simpla tastare a unui nume de automobil individul poate beneficia de acesta pentru întreaga durată de timp pe care o dorește .Mai mult decât atât persoana nu trebuie sa se deplaseze pentru a intra în posesia mașinii deoarece aceasta va fi ridicată de la locația pe care clientul o selectează.

Proiectarea soluției

Această aplicație a fost realizată utilizând TDA Listă înlănțuită care are următoarea structura

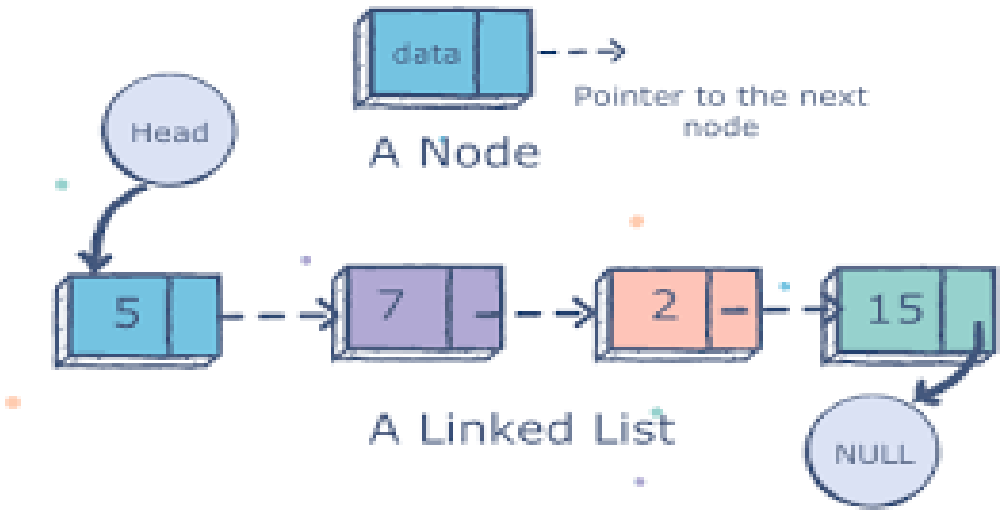

Fig. 1 Structura unei liste înlănțuite

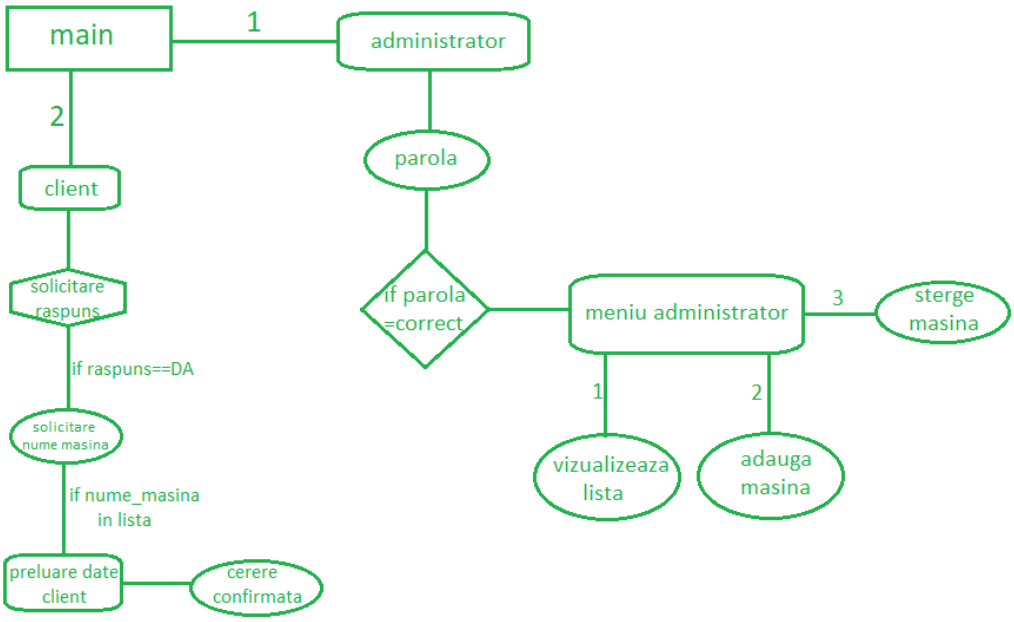

Fig.2 Organigramă proiect

TDA Închiriere maşini

Funcții publice

newl(); ins_la_urma(); nrMasini(); adauga_masina ();

isEmptyl(); isFull(); toStringl(); destroyl();

remove_primul(); sterge_masina(); remove_ultimul();

citirefisier(); adaugafisier(); administrator(); client();

preluare_date_client(); numar_tel(); parola(); confirmare();

Funcții private

creareMasina();

Fig.3 Funcțiile TDA

Considerații de implementare

Pentru modelarea nodului din listă am folosit structura „*masina* “ :

```
struct masina
{
    // datele efective memorate

    int id;
    char *nume;
    char *carburant;
    char *culoare;
    int locuri;
    int km;
    int pret;
    // legatura catre nodul urmator
    struct masina *urm;
};
typedef struct masina Masina;
```

Fig,4 Structura mașină

Proiectul a fost realizat utilizând următoarele funcții:

1.Inițializarea unei liste(newl):

Rolul acestei funcții este de a acolo spațiul necesar pentru stocarea unei liste, folosind malloc(), iar apoi de a returna adresa spațiului acordat.

```
LISTA newl()
{
    LISTA header = (LISTA) malloc( sizeof(Lista) );
    if(header != NULL)
    {
        header->nr = 0;
        header->primul = NULL;
        header->ultimul = NULL;
    }
    return header;
}
```

Fig.5 Funcția newl()

2.Crearea unei mașini (creareMașină):

Această funcție are rolul de a aloca spațiul necesar pentru crearea unui element,utilizând funcția *malloc()*;.În cazul în care elementul este diferit de NULL, el va primi datele din structura masina, după care va fi returnat.

```
static MASINA creareMasina(int idu, char *numec, char *combustibil, char *culoarem, int nr_locuri, int kmp, int pretm, MASINA urmator)
{
    MASINA w= (MASINA) malloc( sizeof(Masina) );
    if(w!=NULL)
    {

        w->id = idu;
        w->nume = numec;
        w->carburant = combustibil;
        w->culoare = culoarem;
        w->locuri = nr_locuri;
        w->km = kmp;
        w->pret= pretm;
        w->urm  = urmator;
    }
    return w;
}
```

Fig.6 Funcţia creareMasina

3.Inserarea unui element la sfârşitul listei (ins_la_urma):

În primul rând,în cadrul acestei funcții se verifică dacă lista are elemente.Dacă aceasta este plină,va fi afișată.În caz contrar se adaugă elemente în ea după care se returnează lista.

```
LISTA ins_la_urma(LISTA l, int idu, char *numec, char *combustibil, char *culoarem, int nr_locuri,int km, int pretm)
{
    assert(l!=NULL);

    if(isFull(l))
        return l;
    MASINA w = creareMasina(idu, numec, combustibil, culoarem, nr_locuri,km, pretm, NULL);
    if(w!=NULL)
    {
        if(isEmptyl(l))
        {
            l->primul = l->ultimul = w;
        }
        else
        {
            l->ultimul->urm = w;
            l->ultimul = w;
        }
        l->nr++;
    }
    return l;
}
```

Fig.7 Funcţia ins_la_urma

4.Numărul de mașini din listă (nrMasini):

Are rolul de a returna numărul de mașini prezente în listă.

5.Verificare stare listă (isEmptyl și isFull):

Funcția isEmptyl verifică dacă lista este goală iar funcția isFull verifică dacă lista este plină.

6.Afișsarea listei (toString):

În cadrul acestei funcții se verifică starea listei.Dacă aceasta este goală,mesajul vida va fi adăugat în cadrul char s,primit ca variabilă în momentul declarării funcției.Altfel se parcurge lista element cu element, datele fiind stocate în cadrul variabilei buf,după care concatenate în s.La final variabila s este afișată.

```
char *toStringl(LISTA l,char *s)
{
    char buf[5000];
    sprintf(s, "\t\t\t\t\t\t--Lista Masini-- \n ");

    assert(l!=NULL);

    if(isEmptyl(l))
    {
        strcat(s, "vida ");
    }
    else
    {
        MASINA p;
        for(p=l->primul; p!=NULL; p = p->urm)
        {
            sprintf(buf,  "\nID: %-4d \tNume: %-20s \tCarburant: %-15s \tCuloare: %-15s  \tNumar locuri: %-4d \tKm/h: %-4d \tPret(RON)
            strcat(s, buf);
        }
    }
    return s;
}
```

Fig.8 Funcția toString

7.Adăugare mașină în listă (adauga_masina):

Această funcție are rolul de a adăuga o mașină în listă.Datele sunt preluate de la tastatură.Înainte de preluarea fiecărei date este curățată consola utilizând *fflush(stdin).*Pentru elementele de tip char, se alocă spațiu și se inițializează memoria alocată cu 0 folosind funcția *calloc().*Datele preluate sunt copiate într-o variabilă *char zona* după care vor fi inserate la sfârșitul listei apelând funcția *ins_la_urma().*Deoarece lista de mașini este stocată într-un fișier, și adăugarea lor se face în același fișier.

```
STA adauga_masina ( LISTA l)

  assert(l!=NULL);
  char zona[1000], *nume, *culoare, *carburant;
  int id, nrlocuri, km, pret;

  fflush(stdin);
  printf("Numele masinii este: ");
  gets(zona);

  nume = calloc(strlen(zona)+1, sizeof(char));
  strcpy(nume, zona);

  fflush(stdin);
  printf("Carburantul este: ");
  gets(zona);
  carburant = calloc(strlen(zona)+1, sizeof(char));
  strcpy(carburant, zona);

  fflush(stdin);
  printf("Culoarea este: ");
  gets(zona);
  culoare = calloc(strlen(zona)+1, sizeof(char));
  strcpy(culoare, zona);

  printf("Numarul de locuri: ");
  scanf("%d", &nrlocuri);
```

```
    printf("Numarul de locuri: ");
    scanf("%d", &nrlocuri);

    printf("Maxim km/h: ");
    scanf("%d", &km);

    printf("Pret autovechicul/zi: ");
    scanf("%d", &pret);

    id=l->nr+1;

    ins_la_urma(l,id,nume, carburant, culoare, nrlocuri,km, pret);
    adaugafisier(l);
    printf("\nMasina a fost inregistrata!\n");

    return l;
}
```

Fig.9 Funcția adauga_masina

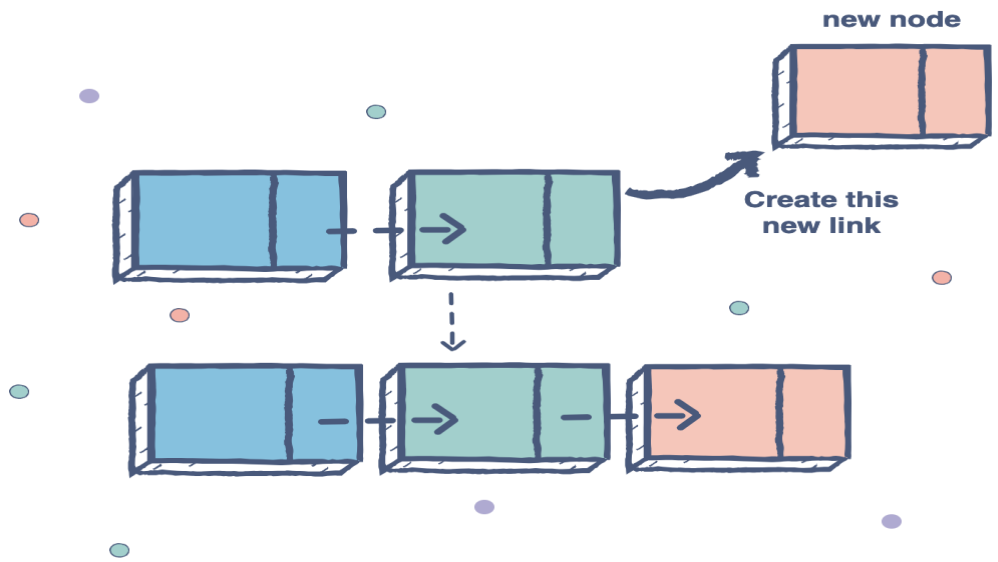

Fig.10 Reprezentare adăugare element în listă

8.Scrierea datelor în fișier (adaugafisier):

Această funcție are rolul de a scrie datele într-un fișier text.

```
void adaugafisier(LISTA l)
{
    FILE *fp=fopen("masini.txt","wt");
    assert(fp!=NULL);
    fprintf(fp,"%d\n",l->nr);
    MASINA p;
    for(p=l->primul; p!=NULL; p=p->urm)
    {
        fprintf(fp, "%d,", p->id);
        fprintf(fp, "%s,", p->nume);
        fprintf(fp, "%s,", p->carburant);
        fprintf(fp, "%s,", p->culoare);
        fprintf(fp, "%d,", p->locuri);
        fprintf(fp, "%d,", p->km);
        fprintf(fp, "%d\n",p->pret);

    }
    fclose(fp);
}
```

Fig.11 Funcția adaugafisier

9.Citirea datelor din fișier (citirefisier):

Această funcție a fost creată pentru a se putea citi date dintr-un fișier text. Fiecare informație este cosiderată scrisă pe o linie fiind delimitată de separatorul , . Pentru unele componete este alocat spațiul necesar.

```
LISTA citirefisier()
{
    LISTA l=newl();
    assert(l!=NULL);

    FILE *f = fopen("masini.txt", "r");
    if(f==NULL)
        printf("Eroare la deschiderea fisierului");
    else
    {
        char zona[100], *buff, *nume, *culoare, *carburant;
        int id, nrlocuri, km, pret,nr;

        fscanf(f, "%d", &nr);
        char c[5];
        fgets(c,5,f);//Trece pe urmatorul rand
        for(int i=0; i<nr; i++)
        {
            fflush(stdin);
            fgets(zona,100,f);

            buff=strtok(zona, ",");
            id=atoi(buff);

        buff=strtok(NULL, ",");
        nume=calloc(strlen(buff)+1, sizeof(char));
        assert(nume!=NULL);
        strcpy(nume, buff);

        buff=strtok(NULL, ",");
        carburant=calloc(strlen(buff)+1, sizeof(char));
        assert(carburant!=NULL);
        strcpy(carburant, buff);

        buff=strtok(NULL, ",");
        culoare=calloc(strlen(buff)+1, sizeof(char));
        assert(culoare!=NULL);
        strcpy(culoare, buff);

        buff=strtok(NULL, ",");
        nrlocuri=atoi(buff);

        buff=strtok(NULL, ",");
        km=atoi(buff);

        buff=strtok(NULL, ",");
        pret=atoi(buff);

        ins_la_urma(l,id,nume,carburant,culoare,nrlocuri,km,pret);

    }
}
return l;
```

Fig.12 Funcția citirefisier

10.Ștergerea unei mașini din listă (sterge_masina):

Această funcție a fost creată pentru a permite ștergerea unei mașini din cadrul listei.Inițial se cere id-ul autoturismului care se dorește a fi șters,după care se parcurge lista element cu element.

Se verifică dacă id-ul introdus este egal cu cel al mașinii din listă,după care se șterge autoturismul selectat.În cadrul acestei funcții sunt apelate si funcțiile remove_primul(), respectiv remove_ultimul().

```
LISTA sterge_masina(LISTA l)
{
    printf("\nDati ID-ul masinii pe care doriti sa o stergeti:");
    int ID;
    do{scanf("%d",&ID);
    }while(ID<1 || ID>l->nr);
    MASINA p;
    MASINA j;
    int ok=0;
    for(p=l->primul;p!=NULL;p=p->urm)
    {
        if(ok==1)
        {
            p->id=p->id -1;
        }
        else
        {
            if(p->id==ID)
            {
                ok=1;
                if(p==l->primul)
                {
                    remove_primul(l);
                }
                else
                {
                    if(p==l->ultimul)
                    {
                        remove_ultimul(l);
                    }
                    else
                    {
                        for(j=l->primul;j->urm!=p;j=j->urm);
                        j->urm=p->urm;
                        free(p);
                         l->nr--;
                    }
                }
            }
        }
    }
    adaugafisier(l);
    destroyl(l);
    return l;
}
```

Fig.13 Funcția sterge_masina

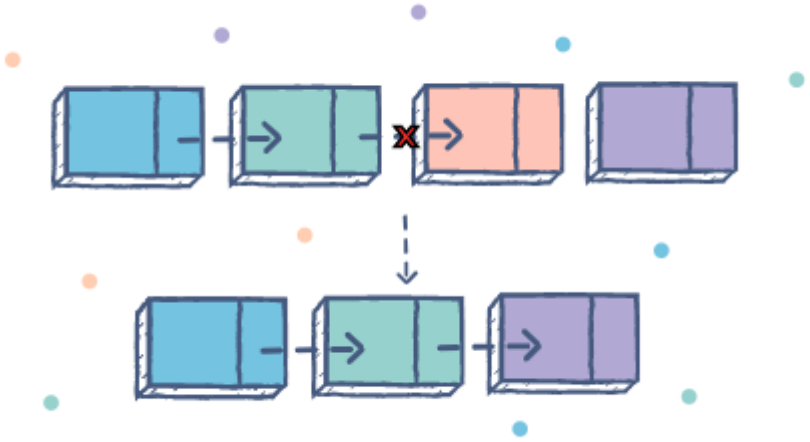

Fig.14 Reprezentare ștergerea unui element din listă

11.Îndepărtarea primului element din listă (remove_primul):

Rolul acestei funcții este cel de a șterge primul element din listă.Dacă lista are elemente , se decrementează numărul de mașini din listă după care al doilea element devine primul.Cel care ocupa poziția întâi este eliminat.

```
LISTA remove_primul(LISTA l)
{
    assert(l!=NULL);

    if(l->primul!=NULL)
    {
        l->nr--;
        MASINA p = l->primul->urm;
        free(l->primul);
        l->primul = p;
        if(l->primul == NULL )
            l->primul=NULL;
    }
    return l;
}
```

Fig.15 Funcția remove_primul

12. Îndepărtarea ultimului element din listă (remove_ultimul):

Rolul acestei funcții este cel de a șterge ultimul element din listă.În cazul în care aceasta nu este deja goală se parcurge lista până la ultimul element care va fi șters utilizând *free(l->ultimul).*Elementul următor va fi NULL.

```
LISTA remove_ultimul(LISTA l)
{
    assert(l!=NULL);
    if(isEmptyl(l))
        return l;
    l->nr--;
    if(l->primul == l->ultimul)
    {
        free(l->primul);
        l->primul = l->ultimul = NULL;
    }
    else
    {
        MASINA p;
        for(p=l->primul; p->urm!=l->ultimul; p = p->urm) ;
        free(l->ultimul );
        p->urm = NULL;
        l->ultimul = p;
    }
    return l;
}
```

Fig.16 Funcția remove_ultimul

Manualul de utilizare al aplicației

Această aplicație a fost concepută pentru a demonstra practicabilitatealitatea tipului de TDA Listă înlănțuită, și funcționează astfel:

La deschiderea programului utilizatorul întâlnește următorul meniu:

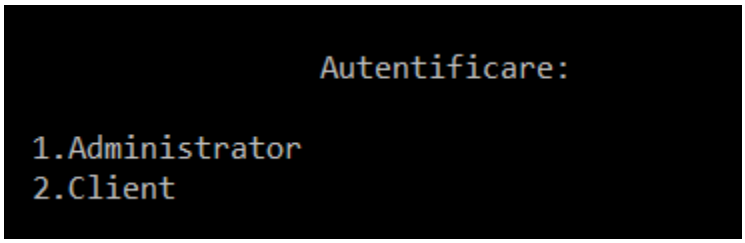

Fig.17 Meniu principal

În cazul în care utilizatorul este administratorul aplicației, acesta va selecta opțiunea ‚1',în cadrul căreia îi va fi solicitată o parolă de acces:

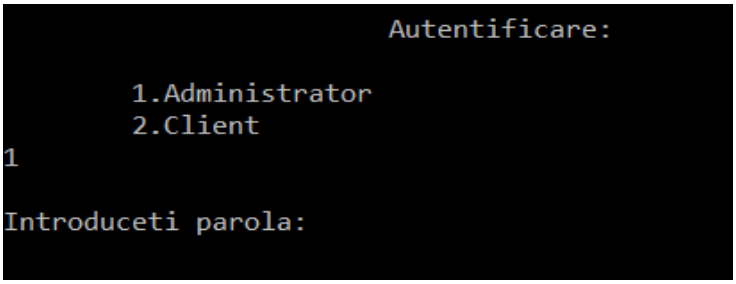

Fig.18

Dacă parola introdusă este cea corectă ,administratorul va putea vizualiza meniul aferent optiunii ‚1':

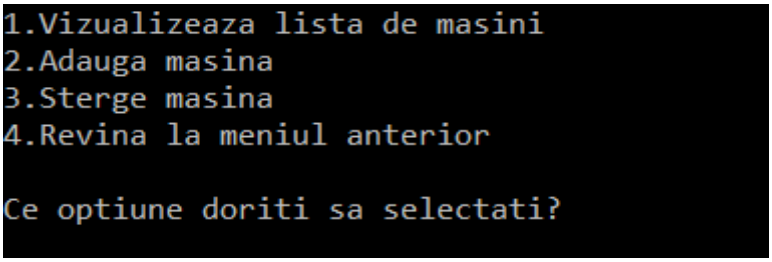

Fig.19

Dacă va fi selectată optiunea 1, administratorul va putea vizualiza lista de mașini.Pentru a putea adăuga o mașină în listă va fi selectată opțiunea 2 ,iar pentru a șterge o mașină din cadrul listei va fi aleasă opțiunea 3.

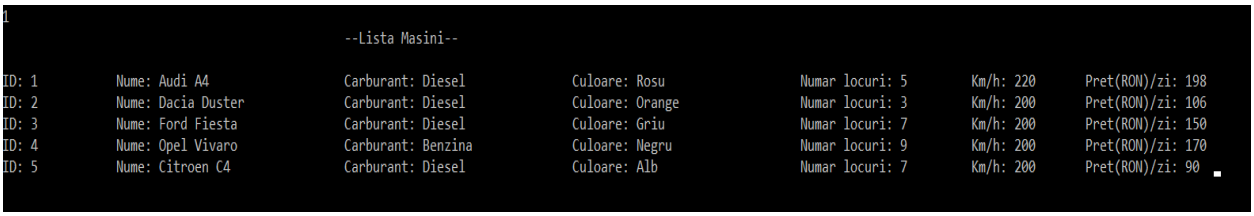

Fig.20 Lista aferentă primei opțiuni

În cazul în care utilizatorul este client, va selecta opțiunea 2 din meniul principal și astfel va avea posibilitatea de a închiria , sau nu, un autoturism din lista de mașini afișată.

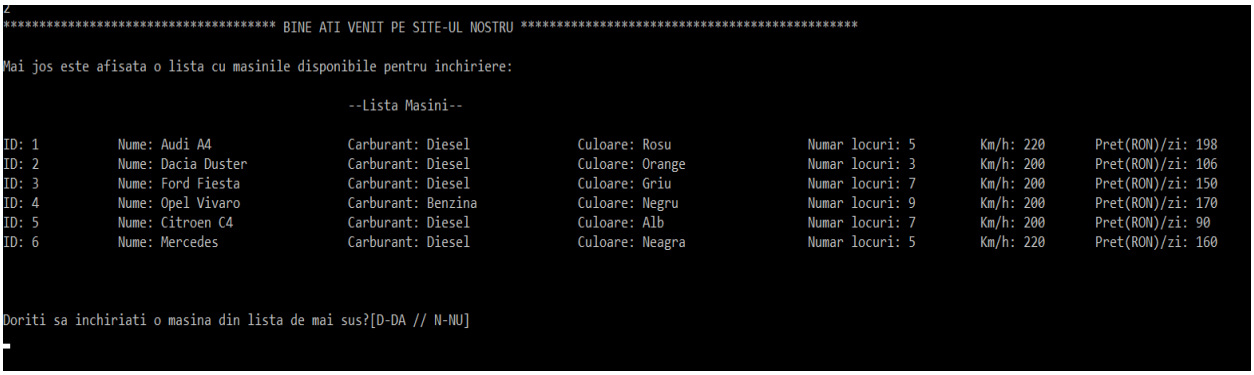

Fig.21 Meniu aferent opțiunii 2

Dacă clientul dorește să închirieze o mașină din listă, va fi nevoit sa introducă numele acesteia, urmând să îi fie preluate datele personale.După inserarea datelor, cererea clientului va fi înregistrată.

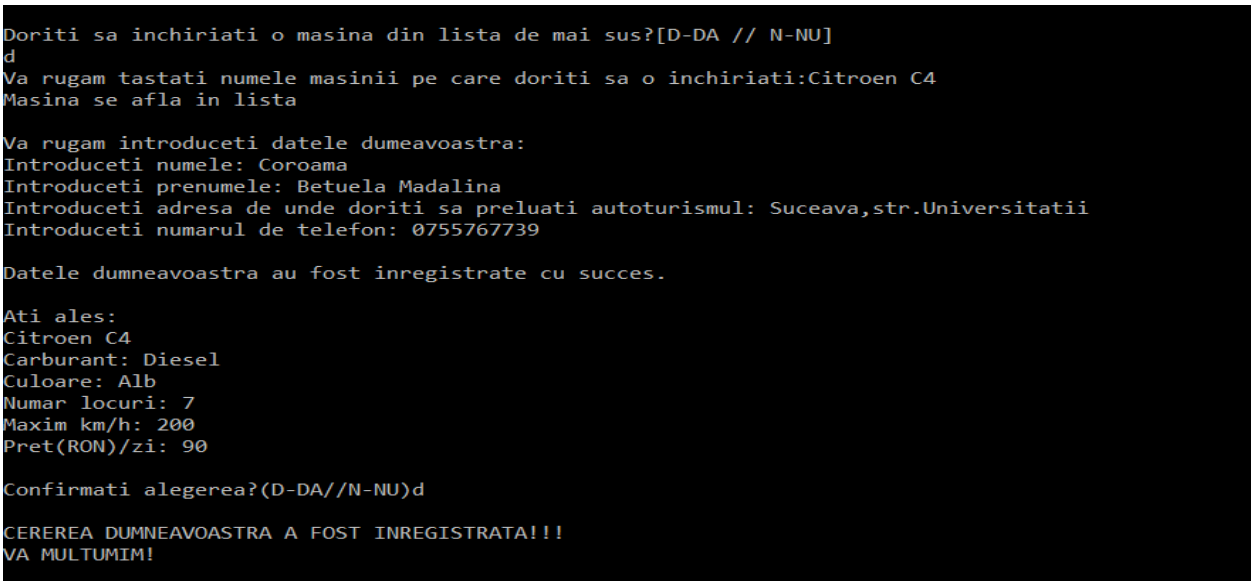

Fig.22 Opțiunea client

Referințe bibliografice

http://eed.usv.ro/~ionelar/teaching.php

https://www.educative.io/edpresso/what-is-a-singly-linked-list

https://www.geeksforgeeks.org/linked-list-set-1-introduction/

https://www.programiz.com/dsa/linked-list