

**Universitatea Tehnică a Moldovei**

**SISTEM DE REZERVARE A BILETELOR DE AVION „AirCornaby”**

**Reservation SYSTEM OF PLANE TICKETS „aIRCORNABY”**

|  |  |
| --- | --- |
| **Studenți:** | **gr. TI-231,**  **Ababii Ionel**  **gr. TI-233,**  **Cojocaru Cornelia** |
| **Coordonator:** | **Cebotari Daria,**  **asist.univ.** |

**Chişinău, 2024**

CUPRINS

[ABREVIERI 3](#_Toc183815508)

[INTRODUCERE 4](#_Toc183815509)

[1 ANALIZA DOMENIULUI DE STUDIU 5](#_Toc183815510)

[1.1 Importanța temei 6](#_Toc183815511)

[1.2 Sisteme similare cu proiectul realizat 7](#_Toc183815512)

[1.3 Scopul sistemului 8](#_Toc183815513)

[1.4 Definirea datelor problemei și stabilirea relației între date și soluția problemei 9](#_Toc183815514)

[2 ALGORITMUL SISTEMULUI 11](#_Toc183815515)

[2.2 Alegerea algoritmilor pentru rezolvarea problemei 13](#_Toc183815516)

[2.3 Argumentarea alegerii algoritmilor 15](#_Toc183815517)

[3 REALIZAREA SISTEMULUI 17](#_Toc183815518)

[3.1 Implementarea algoritmilor 18](#_Toc183815519)

[3.2 Interfața sistemului 19](#_Toc183815520)

[3.3 Analiza rezultatelor obținute 23](#_Toc183815521)

[CONCLUZII 25](#_Toc183815522)

[REFERINȚE BIBLIOGRAFICE 26](#_Toc183815523)

[ANEXA A 27](#_Toc183815524)

[ANEXA B 29](#_Toc183815525)

[ANEXA C 31](#_Toc183815526)

# **ABREVIERI**

**TIC** -Tehnologia informației și a Comunicațiilor;

**Online** - starea de conectare la o rețea;

**Device** - dispozitiv, aparat, instrument tehnic, utilizat în diferite domenii a științei și vieții de zi cu zi;

**Shopping online** - a face cumpărături pe platforme online;

**Low cost** - prețuri accesibile;

**Pop-up** - notificări care apar pe ecranul unui dispozitiv despre anumite evenimente;

**Feedback** - expunerea părerii asupra unui anumit subiect;

**Scam** - o încercare de fraudare a unei persoane, informațiilor personale sau bunurilor materiale;

**Social media** -rețelele de socializare (Instagram, Facebook etc.)

**Feed** - pagina unde e posibilă vizualizarea tuturor postărilor unui utilizator;

**Hardware** - compontentele fizice ale calculatorului;

**E-learning** - învățarea prin intermediul platformelor online;

**IT**- Informational Technology (tehnologia informațională);

**POO** - programarea orientată pe obiecte;

**MainMenu** - meniul principal;

**Cart** - coșul de cumpărături;

**CheckOut** - tranzacția de plată;

**Load** - încărcare (programe, site-uri etc.);

**Search** - a căuta;

**Select** - a selecta;

**Receipt** - bon de plată;

**Debugging** - depanarea erorilor;

**Submit** - a depune;

**Proceed to pay** - continuarea către plată.

# **INTRODUCERE**

În ziua de astăzi tehnologiile au devenit o necesitate de zi cu zi, sistemele digitale, jocurile, filmele, totul este digitalizat. Sistemele de rezervare a biletelor de avion joacă un rol important în facilitarea călătoriilor, siguranța și liniștea pasagerilor. Cererea pentru soluții rapide și eficiente este în continuă creștere, iar un sistem bine pus la punct nu doar va optimiza dar și transformă experiența utilizatorilor.

În plus, un sistem bine definit, cu o interfață plăcută, nu doar va reduce costurile, optimiza volumul de memorie, dar și va atrage o cantitate mai mare de clienți. Evoluția către soluțiile moderne, nu doar arată progresul tehnologic în țara noastră, dar se mai și reflectă în economia actuală.

Proiectul dat se axează pe dezvoltarea unei platforme intuitive, care să simplifice procesul de achiziție a biletelor de avion. Utilizatorii vor putea căuta zboruri, selecta opțiuni din cele valabile, după preț, și finaliza tranzacția de rezervare într-un mod sigur și eficient.

În Moldova, sistemele de rezervare existente se confruntă cu numeroase provocări, inclusiv actualizări ineficiente, date care nu sunt în pas cu noile tehnologii și interfețe prea complexe. Acestea evidențiază necesitatea unui sistem nou, care să rezolve aceste probleme. Unele din acestea sunt: autentificarea simplă a utilizatorilor, gestionarea zborurilor, criterii de căutare și filtrare prin algoritmi bine definiți, procesarea plăților și confirmarea rezervărilor prin expedierea biletelor pe poștă.

Prin aplicarea acestor modificări se rezultă o experiență optimizată, asigurând că fiecare pas, de la căutarea inițială a zborurilor până la finalizarea rezervării, este simplu și accesibil.

Implementarea acestui sistem nu doar că va îmbunătăți experiența de călătorie pentru utilizatori, și va susține și dezvoltarea digitală a companiilor aeriene din Moldova. Prin abordarea cerințelor simple, sistemul va deveni foarte eficient, capabil să satisfacă atât nevoile utilizatorilor, cât și ale companiilor avia.

Lucrarea este structurată în trei capitole, unde se va analiza concret modul de proiectare și de realizare a proiectului dat, fiecare capitol fiind dedicat unei etape importante în dezvoltarea platformei.

Primul dintre aceste capitole vine să prezinte fundamentele teoretice și necesitatea unui astfel de sistem în contextul actual, alături de cerințele și specificațiile funcționale, compararea și analiza sistemelor deja existente și funcționale, datele problemei și soluțiile acesteia.

Al doilea capitol se concentrează pe inima platformei, care conține algoritmii pe care ar putea fi bazată platforma biletelor de avion, compararea acestora și analiza avantajelor și dezavantajelor acestora, analiza empirică a algoritmilor pentru a explica alegerea celui mai efectiv.

În ultimul capitol se cuprinde partea de implementare a codului și testarea funcționalităților, interfața grafică a aplicației cu toate posibilitățile de alegere a opțiunilor, pentru a asigura că toate cerințele inițiale sunt îndeplinite, aplicația funcționează corespunzător și că utilizatorii beneficiază de o experiență complet optimizată.

# **1 ANALIZA DOMENIULUI DE STUDIU**

În afaceri, Tehnologia informației și a Comunicațiilor este adesea clasificată în două tipuri mari de produse: tehnologiile tradiționale bazate pe computer (lucruri care pot fi facute pe un computer personal) şi gama mai recentă și în creștere rapidă de tehnologii de comunicare digitală (care permit oamenilor și organizațiilor să comunice și să partajeze informații digital). [1] Sistemele de rezervare a biletelor de avion se referă la ambele.

TIC a devenit o parte fundamentală a vieții noastre de zi cu zi, influențând aproape fiecare aspect al societății moderne. De la comunicare și educație la transport, sănătate și divertisment, tehnologia ne-a transformat modul de viață, schimbând radical accesul la informații, oferind soluții mai rapide și mai eficiente. Această dezvoltare rapidă a tehnologiei informației a avut un impact pozitiv în multe domenii.

În domeniul comunicării, TIC a avut un efect complet radical. Azi, oamenii pot comunica, indiferent de distanțele fizice. Telefoanele inteligente, rețelele sociale și aplicațiile de mesagerie permit conexiunea cu cei dragi, conexiunea profesională și transmiterea informației într-un mod rapid și eficient. Videoconferințele și platformele online au făcut posibile întâlniri și evenimente virtuale, schimbând modul în care oamenii interacționează și colaborează, mai ales la nivel internațional.

În educație, tehnologia informației a schimbat modul în care elevii și studenții învață și accesează cunoștințele. Platformele de e-learning, accesibile de pe computere sau telefoane mobile, cum ar fi Moodle, permit oamenilor să studieze de oriunde și să își completeze studiile în ritmul propriu. [2] De asemenea, bibliotecile digitale, cursurile online și resursele educaționale gratuite oferă acces la o varietate largă de informații, făcând educația accesibilă pentru mai mulți oameni din întreaga lume.

Tehnologia informației a schimbat radical modul în care ne deplasăm și organizăm călătoriile. De la aplicațiile de navigație GPS, care ne ajută să găsim traseul cel mai rapid, la cele mai avansate device-uri.

În afaceri, tehnologia informației joacă un rol esențial în automatizarea și eficientizarea proceselor. Sistemele IT permit companiilor să gestioneze date, automatizeze anumite operații, cum ar fi înregistrarea prezenței angajaților, procesarea documentelor etc. [3] De asemenea, comerțul online a devenit o parte integrată a economiei globale, permițând oamenilor să facă cumpărături din confortul propriei case pe platformele de shopping online și companiilor să-și extindă piața la nivel global. Conform statisticilor globale de la GO-Globe, la finele anului 2024 se așteaptă un venit de 6.3 trilioane $ SUA de la cumpărăturile online globale. [4]

Unul dintre cele mai evidente exemple este digitalizarea proceselor din industria transportului aerian. Astăzi, căutarea și rezervarea unui zbor pot fi realizate rapid și eficient prin intermediul platformelor online, eliminând barierele fizice și oferind o experiență de călătorie mai accesibilă și mai comodă pentru utilizatori. Sunt disponibile nu doar biletele, dar și monitorizarea bagajelor, locurilor VIP, selectarea claselor de bussines și confort. Acest proces nu doar că simplifică planificarea călătoriilor, dar contribuie și la dezvoltarea economică, facilitând accesul mai ușor la servicii internaționale.

## **1.1 Importanța temei**

În contextul actual, necesitatea unor noi sisteme de rezervare a biletelor este din ce în ce mai evidentă, deoarece multe dintre sistemele existente sunt depășite și nu mai răspund nevoilor moderne ale utilizatorilor. Platformele vechi nu dispun de funcționalități avansate, cum ar fi actualizări în timp real, integrări cu alte servicii sau opțiuni de personalizare pentru utilizatori. Lipsa de adaptabilitate a acestor sisteme poate duce la experiențe frustrante pentru clienți, care întâmpină dificultăți în a găsi cele mai bune opțiuni de zbor sau întârzieri în confirmarea rezervărilor. Prin dezvoltarea unor platforme moderne, se poate răspunde mai bine cerințelor actuale, asigurând o interfață intuitivă, procese rapide și o gestionare eficientă a datelor, ceea ce contribuie la satisfacția clienților.

Importanța acestei teme se reflectă în câteva aspecte cheie:

* **experiența utilizatorului;**
* **eficiența operațională**;
* **inovație și adaptabilitate.**

Datele statistice arată că, în 2018, aproximativ 82% din toate rezervările de bilete au fost efectuate pe platforme online pentru rezervarea zborurilor, subliniind importanța acestora în industria turismului. [5] Un sistem digital bine structurat permite operatorilor aerieni să gestioneze mai eficient cererea și să personalizeze ofertele, adaptându-se rapid la schimbările pieței. În acest mod, companiile avia nu doar că îmbunătățesc calitatea serviciilor, dar câștigă și încrederea clienților, care apreciază flexibilitatea și rapiditatea proceselor digitale.

În plus, digitalizarea rezervărilor contribuie la reducerea costurilor operaționale și creșterea profitabilității pentru companiile aeriene. În loc să depindă de centre de apel sau locații fizice, clienții pot face rezervările direct online, economisind timp și resurse. Un astfel de sistem devine esențial într-o eră în care oamenii caută simplitate și eficiență în orice activitate cotidiană, de la cumpărături până la călătorii.

Prin urmare, dezvoltarea continuă a sistemelor IT, precum platformele de rezervare de bilete, reprezintă nu doar o necesitate, ci și o oportunitate de a crea un ecosistem de servicii interconectate și accesibile.

Inovația tehnologică în acest domeniu nu îmbunătățește doar experiența călătorilor, dar și reputația companiilor implicate, evidențiind impactul pozitiv al IT-ului asupra vieții de zi cu zi.

Programarea orientată pe obiecte reprezintă un alt concept fundamental în dezvoltarea aplicațiilor complexe, cum sunt cele pentru rezervarea zborurilor. Principiile POO permit dezvoltatorilor să structureze codul în mod modular, prin utilizarea claselor și obiectelor, abstractizării situației pentru o dezvoltare în viitor. [6] Acest mod de organizare ajută la gestionarea și actualizarea ușoară a codului, la evitarea duplicării funcționalităților și la crearea unor interacțiuni clare între componentele sistemului. POO aduce avantaje importante în organizarea codului, reutilizarea acestuia și dezvoltarea pe termen lung a sistemului.

Astfel, în realizarea unei platforme complexe, cum este cea de rezervare a biletelor de avion, atât algoritmica cât și programarea orientată pe obiecte sunt necesare pentru a obține un sistem eficient, flexibil și ușor de întreținut. Aceste metode permit obținerea unei experiențe de utilizare îmbunătățite și permit extinderea funcționalităților în viitor, în conformitate cu nevoile pieței și cerințele tehnologice moderne.

### **1.2 Sisteme similare cu proiectul realizat**

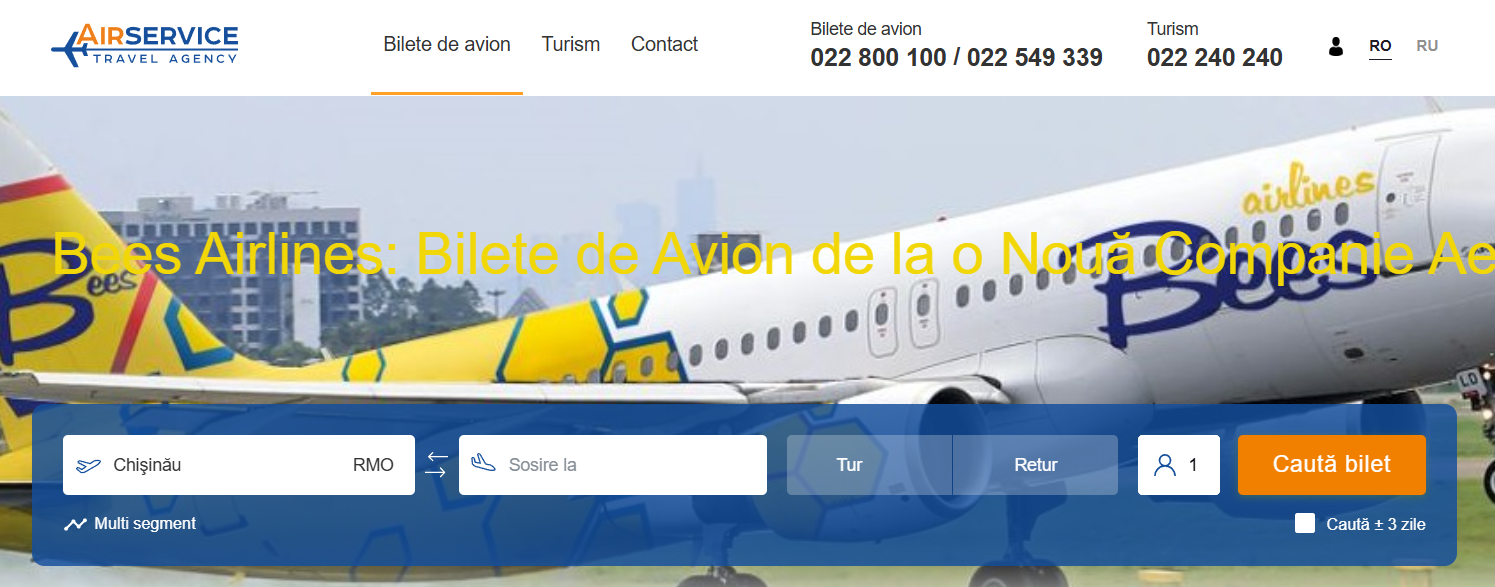
În cadrul analizei platformelor deja existente pentru rezervarea biletelor din Republica Moldova, s-au identificat trei cele mai populare platforme, și anume: “ZBOR.md”, “AIRSERVICE”, și “Avigo”. [7][8][9]

În figura 1.1 este reprezentată pagina de start a platformei online ZBOR.md, care are un design ușor de utilizat, oferind utilizatorilor acces rapid la informații despre zboruri și tarife. Platforma dispune de opțiuni de filtrare avansate, permițând clienților să selecteze criterii precum ora de plecare și compania aeriană preferată. De asemenea, ZBOR.md se remarcă prin suportul clienți disponibil non-stop, oferind asistență și soluționând eventualele probleme în timp scurt. Totuși, aceasta prezintă limitări la actualizarea în timp real a disponibilității biletelor. [7]



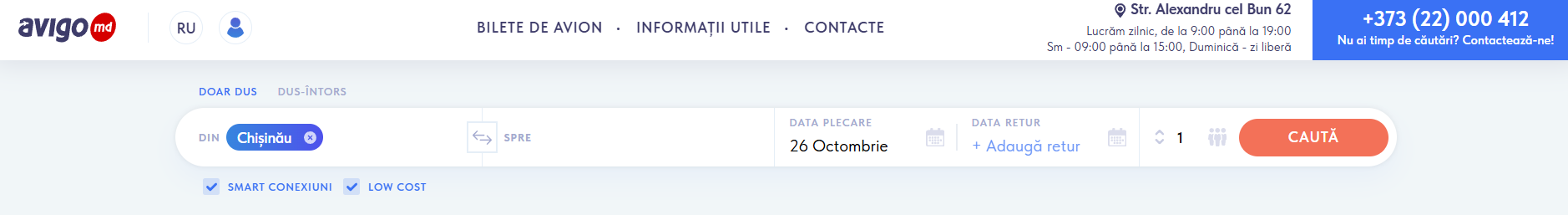
**Figura 1.1 – Pagina de start a platformei web ZBOR.md**

În figura 1.2 este reprezentată pagina de start a platformeionline AIRSERVICE, care are integrări avansate cu companii aeriene și opțiuni de plată diversificate, contribuind la o experiență de utilizare ușoară. AIRSERVICE pune accent pe securitatea plăților, utilizând tehnologii avansate pentru protejarea datelor și tranzacțiilor clienților. [8] În plus, platforma oferă reduceri speciale pentru membrii înregistrați și posibilitatea de a compara prețurile între companiile aeriene, optimizând astfel selecția de zboruri. Însă, lipsa unei interfețe mai simple poate îngreuna găsirea ofertelor personalizate, ceea ce face mai dificilă identificarea zborurilor potrivite pentru utilizatori.



**Figura 1.2 - Pagina de start a platformei web AIRSERVICE**

În figura 1.3 este reprezentată platforma online Avigo, care are funcții de căutare personalizate, permițând utilizatorilor să filtreze rezultatele după preferințe de călătorie și buget. [9] Platforma include o secțiune dedicată recenziilor utilizatorilor, oferind informații utile și ajutând clienții să ia decizii informate. Aceasta se mai evidențiază prin opțiunile sale low cost, și anume utilizatorii pot alege dacă doresc biletele sale la cel mai jos preț posibil. Cu toate acestea, platforma nu dispune de suport pentru notificări în timp real privind modificările de zbor.



**Figura 1.3 - Pagina de start a platformei web Avigo**

## **1.3 Scopul sistemului**

Este important de definit bine scopurile platformei de rezervare a biletelor de avion, deoarece acestea reprezintă fundația pentru un proiect complex. Prin definirea clară a scopurilor, se poate asigura că platforma răspunde nevoilor utilizatorilor și oferă soluții rapide și precise pentru achiziția de bilete și planificarea călătoriilor.

Scopul principal al sistemului de rezervare a biletelor de avion este de a crea o platformă accesibilă și eficientă, care să simplifice procesul de achiziție. Acesta va permite utilizatorilor să efectueze următoarele acțiuni: **căutarea rapidă**; prezentarea clară a prețurilor; asigurarea unui sistem de plată sigur.

Fiind o platformă simplă, este la fel una ușor de utilizat. Utilizatorii nu vor fi deranjați de mesaje pop-up, notificări stranii sau publicități enervante. De asemenea nu vor apărea mesaje de scam sau eschrocherii, cum se întâmplă adesea pe platformele online, în special cu noile tendințe de furturi a datelor bancare pe platformele de shopping online.

Obiectivele unei platforme de rezervare sunt orientate spre oferirea unei experiențe sigure, rapide și accesibile pentru utilizatori, eliminând astfel barierele tradiționale legate de achiziționarea biletelor de avion. Obiectivele bine stabilite contribuie la îmbunătățirea fiecărui aspect al platformei, de la căutare și filtrare până la confirmarea rezervărilor, asigurând în același timp satisfacția clienților. Obiectivele specifice ale sistemului includ:

* **interfața prietenoasă;**
* **căutarea avansată;**
* **sistem de actualizare în timp real;**
* **nivel înalt de securitate;**
* **timp scurt de căutare;**
* **feedback in caz de necesitate.**

Cerințele sistemului sunt clasificate în funcționale și nefuncționale, fiecare cu un rol aparte în integritatea și funcționalitatea proiectului. Cerințele funcționale reprezintă: **autentificarea utilizatorilor prin c**rearea unui cont de utilizator și gestionarea sesiunilor; **gestionarea zborurilor disponibile**; **căutarea opțiunilor de zbor; procesarea plăților; confirmarea rezervărilor prin g**enerarea și trimiterea de confirmări prin e-mail. Cerințele nefuncționale pe de altă parte sunt că sistemul trebuie să fie capabil: să gestioneze un număr crescut de utilizatori fără a compromite performanța; să răspundă rapid la cererile utilizatorilor, asigurându-se timpi scurți de încărcare a paginilor; să utilizeze algoritmi cât mai eficienți; să nu dea greș în momentele excepționale.

Platformele de rezervare urmăresc și îmbunătățirea proceselor interne ale companiilor aeriene, facilitând gestionarea eficientă a resurselor și anticipând cerințele pieței. Prin dezvoltarea unui astfel de sistem companiile nu doar ca își pot personaliza designul cum doresc dar să își și aleagă parametrii care îi doresc, începând de la filtre până la statistici. Astfel companiile de avioane vor putea soluționa mai multe probleme printr-o singură metodă.

## **1.4 Definirea datelor problemei și stabilirea relației între date și soluția problemei**

Un sistem bine definit este reputația unei companii față de publicul său. Utilizarea acestuia oferă un feedback despre calitatea acestuia, iar un feedback negativ nu doar că distruge reputația, dar poate să și falimenteze compania.

Pentru a dezvolta un sistem de rezervare a biletelor de avion eficient și accesibil, este esențială definirea clara a datelor problemei, obiectivele urmărite și modul în care acestea vor fi abordate, mereu lăsând loc pentru noi idei sau îmbunătățiri pe viitor. Datele de intrare pentru un sistem de rezervare simplu includ: orașul de pornire; destinația; data de plecare; numărul disponibil de locuri; costul unui bilet; distanța parcursă de avion; durata zborului; numărul zborului.

Pe lângă datele esențiale, sistemul mai include printarea informației personale create pentru fiecare cumpărător sau utilizator a platformei date.

Modul de abordare a problemei sistemului include o interfață ușor de utilizat, care va permite accesarea și vizualizarea datelor în timp real, conform unor metode utilizate, facilitând utilizatorilor alegerea celor mai potrivite zboruri în funcție de nevoile lor, de obicei de prețul unei călătorii. În faza de dezvoltare a platformei, algoritmi de căutare avansată sunt implementați pentru a le oferi utilizatorilor rezultate cât mai relevante, în timp ce securitatea și performanța vor fi asigurate prin optimizarea infrastructurii sistemului.

De asemenea, este creată o bază de date cu toți parametrii de intrare a zborurilor. Atât timp cât acest sistem se bazează pe zborurile între companiile aflate în capitalele Europene, în baza de date vor fi cât mai multe tipuri de zboruri între acestea. În baza de date există mai multe zboruri la preț redus față de alte companii aeriene pentru a asigura cât mai mulți clienți cu bilete pentru călătoriile sale.

Procesul de rezervare este structurat cu pași simpli și clari, de la căutarea zborurilor disponibile până la selectarea numărului de bilete și finalizarea plății, cu printarea bonurilor de plată cu informații actualizate, care includ atât timpul și data achitării cât și informațiile personale a cumpăratorilor, pe lângă însuși informația despre bilete. Datele de intrare, introduse de utilizatori vor fi procesate pentru a afișa opțiunile de zbor relevante pentru fiecare căutare, în dependență de prețul mai accesibil, aranjate de la cele mai ieftine zboruri la cele mai scumpe. Sistemul stochează toate datele despre rezervările făcute și le adaugă în contul utilizatorului logat, într-o bază de date stabilită. Într-un final are loc confirmarea și gestionarea rezervărilor.

În general, sistemele de rezervare a biletelor de avion au o structură foarte asemănătoare, deși algoritmii de căutare diferă, rezultatele pot fi numite asemănătoare. Acest sistem se evidențiază prin simplitate și unicitate. Oamenii apreciază minimalismul, în special în secolul XXI, dând o valoare nu doar funcțională dar și estetică platformei.

Sistemul oferă o soluție completă de rezervare care răspunde tuturor cerințelor definite și îndeplininește obiectivele propuse. Rezultatul este o platformă accesibilă și sigură, cu funcționalități adaptate utilizatorilor și un nivel înalt de performanță. Aceasta facilitează atât procesul de rezervare a biletelor, cât și gestionarea eficientă a resurselor interne ale companiilor aeriene, contribuind astfel la o experiență mai bună și la satisfacția clienților, precum și la competitivitatea și adaptabilitatea companiilor în fața cerințelor pieței moderne.

# **2 ALGORITMUL SISTEMULUI**

Un algoritm este un set de pași clar definiți și logici folosiți pentru a rezolva o problemă sau pentru a realiza o anumită sarcină. În esență, algoritmii reprezintă un mod de a organiza și a executa instrucțiuni astfel încât să se obțină un rezultat dorit. În matematică, informatică și alte domenii, algoritmii sunt folosiți pentru a procesa date, a efectua calcule și a rezolva probleme complexe.

Algoritmii sunt esențiali în foarte multe domenii ale vieții moderne, de la tehnologie la economie și chiar sănătate. De exemplu, algoritmii sunt folosiți în motoarele de căutare, cum ar fi Google, pentru a găsi cele mai relevante informații pentru utilizatori într-o fracțiune de secundă, deși acestea caută prin miliarde de piese de informație, până găsește acel cuvânt sau frază. [10] În social media, algoritmii sunt cei care decid ce postări vedem în feed-ul nostru, bazându-se pe interesele și interacțiunile noastre anterioare.

În economie, algoritmii sunt utilizați pentru a analiza datele pieței și a prezice schimbările financiare, fiind folosiți în sistemele de tranzacționare automată. În medicină, algoritmii de inteligență artificială ajută la diagnosticarea bolilor prin analiza imaginilor medicale și prin identificarea de date complexe în datele pacienților. De asemenea, algoritmii sunt aplicați în transport, cum ar fi în aplicațiile de navigație, pentru a găsi rute optime și pentru a evita traficul.

În programare, algoritmii sunt fundația oricărui program sau aplicație. Fie că este vorba despre căutarea unui element într-o listă, sortarea unor date sau generarea unor trasee optime, algoritmii oferă soluții eficiente pentru aceste sarcini. Un algoritm bine structurat permite rezolvarea unei probleme într-un timp mai scurt și cu un consum redus de resurse, ceea ce este crucial pentru performanța programelor, în special în cazul celor complexe sau care rulează pe dispozitive cu resurse limitate.

Pentru a înțelege importanța algoritmilor și a programării, trebuie de înțeles aplicarea lor în practică prin utilizarea computerelor. Și anume, algoritmilor le este deseori necesar un hardware specific pentru a efectua anumite operațiuni. Deși scrisul propriu-zis nu este solicitant, anumite operații necesită cipuri specializate și o tehnică suplimentară.

Eficiența algoritmilor în programare nu poate fi subestimată, deoarece aceștia determină eficiența și performanța unui software. Un algoritm eficient poate face diferența între un program rapid și unul lent, între o aplicație ușor de utilizat și una frustrant de înceată. Pentru un programator, înțelegerea algoritmilor este esențială pentru a putea crea soluții eficiente și simple. În plus, algoritmii stau la baza învățării automate, a criptografiei și a multor alte tehnologii moderne, motiv pentru care programatorii și specialiștii IT trebuie să aibă cunoștințe solide în acest domeniu. Pentru ca algoritmul să rezolve problema, trebuie ca el să se definească la „intrare”, în timp ce la „ieșire” se găsește [rezolvarea problemei](https://www.link-academy.com/gandirea-critica-si-rezolvarea-problemelor). Ceea ce este important să se menționeze este că algoritmul trebuie să aibă o procedură corectă. Deci, fiecare segment al problemei stabilite (la intrare) trebuie să aibă definită o soluție (la ieșire). Dacă este clar stabilită și nu are nici o greșeală, se consideră că procedura este corectă. [11]

**2.1 Algoritmul sistemului**

Algoritmul Dijkstra este o soluție fundamentală pentru rezolvarea problemelor de rute în rețele, fiind util pentru identificarea drumului optim într-o rețea de zboruri, între orașe conectate prin diferite rute aeriene. În contextul unui sistem de rezervare a biletelor de avion, acesta poate fi aplicat pentru a determina cele mai rapide și eficiente rute între destinații, mai ales atunci când nu sunt disponibile zboruri directe și sunt necesare zboruri de legătură. Acesta nu doar că avansează alți algoritmi, dar și reduce timpul de căutare, un punct important pentru atragerea clienților.

Algoritmului Dijkstra în sistemul de rezervare reprezintă motorul de cătare a acestuia. Obiectivul principal este acela de a furniza utilizatorilor o rută optimizată către destinația dorită. Algoritmul Dijkstra poate fi utilizat pentru a calcula drumul cel mai scurt între două aeroporturi, având ca variabile factori precum durata totală a călătoriei, costul biletelor, numărul de escale sau distanța. Astfel, utilizatorii beneficiază de informații clare și precise, iar platforma le poate oferi cel mai avantajos traseu în raport cu criteriile setate.

Problema este structurată într-o rețea de noduri și arce. Fiecare aeroport este reprezentat de un nod, iar fiecare zbor dintre două aeroporturi este un arc între noduri. Arcul are un cost asociat, care poate fi definit prin durata zborului, costul biletului sau distanța parcursă, în dependență de implementarea acestuia sau dorinței utilizatorului. Algoritmul Dijkstra va parcurge această rețea pentru a identifica drumul cu cel mai mic cost de la nodul sursă (aeroportul de plecare) la nodul de destinație (aeroportul de sosire), conform prețului asociat.

Pașii principali în aplicarea algoritmului Dijkstra sunt: definirea nodurilor și a arcelor, adică a orașelor și drumurilor; inițializarea algoritmului cu punctul de pornire și destinație; iterarea și actualizarea distanțelor și determinarea traseului minim.

De exemplu, în sistemul de rezervare, dacă un utilizator dorește să călătorească de la Paris la Budapesta, dar nu există zbor direct, algoritmul Dijkstra poate identifica alte rute disponibile cu zboruri intermediare, cum ar fi ruta Paris – Viena – Budapesta ca fiind cea mai rapidă sau cea mai avantajoasă ca preț, în funcție de criteriile de optimizare. Astfel, sistemul poate calcula și recomanda ruta optimă, incluzând durata totală și costul fiecărui zbor de legătură.

Ca rezultat al aplicării algoritmului în sistem la finalul procesării, algoritmul Dijkstra furnizează utilizatorului o rută optimizată, adaptată criteriilor selectate, care poate include timpi de tranzit minimi și trasee cost-eficiente. Astfel, sistemul de rezervare devine capabil să ofere o experiență predictibilă și eficientă, completată de informații detaliate despre fiecare legătură aeriană. Pentru clienții care au des zboruri, interfața platformei de unde pot procura biletele de avion este cel mai important punct, deoarece interfața trebuie să fie fluidă și intuitivă de utilizat, pentru a nu ajunge în numeroase meniuri greu de înțeles, și mai rău, de a nu alege corect destinația dorită și data potrivită. Astfel că acest algoritm este perfect pentru a preveni astfel de greșeli ce se întâlnesc destul de des în viața cotidiană.

### **2.2 Alegerea algoritmilor pentru rezolvarea problemei**

Problema sistemului de rezervare a biletelor de avion dat poate fi rezolvată prin mai mulți algoritmi, unde algoritmul Dijkstra este ideal pentru găsirea celui mai scurt drum între orașe, fiind foarte eficient pentru grafuri mari și sparse. Algoritmul Bellman-Ford este util pentru detectarea ciclurilor negative și gestionarea grafurilor cu ponderi negative, deși este mai lent. Algoritmul Floyd-Warshall găsește cele mai scurte căi între toate perechile de orașe, dar este mai puțin eficient pentru grafuri mari. [12] Fiecare algoritm are avantaje specifice, adaptate diferitelor nevoi ale sistemului de rezervare.

Conform Tabelului 1,putem deosebi plusurile și minusurile fiecărui algoritm întrebuințat mai sus, remarcând caracteristicile fiecărui algoritm, fiind incluse caracteristici precum scop, avantaje și complexitate.

**Tabelul 2.1 - Caracterisitcile algoritmilor de cautare a drumului minim**

|  |  |  |  |
| --- | --- | --- | --- |
| **Caracteristici**  **Algoritm** | **Scop** | **Avantaje** | **Complexitate** |
| **Bellman-Ford** | Găsește cele mai scurte căi de la un nod sursă la toate celelalte noduri dintr-un grafic ponderat, inclusiv grafice cu greutăți negative. | Poate detecta cicluri negative și funcționează bine cu grafice dens utilizate. | , unde N este numărul de noduri și M este numărul de arce. |
| **Floyd-Warshall** | Găsește toate perechile de cele mai scurte căi între noduri într-un grafic. | Ușor de implementat și poate gestiona grafice cu greutăți negative (dar nu cicluri negative). | , unde N este numărul de noduri și M este numărul de arce, ceea ce poate fi mai puțin eficient pentru grafice mari. |
| **Dijkstra** | Găsește cea mai scurtă cale de la un nod sursă la toate celelalte noduri, dar funcționează doar cu grafice cu greutăți non-negative. | Foarte eficient pentru grafice mari, mai ales când este combinat cu o structură de date adecvată (cum ar fi un heap). | , unde N este numărul de noduri și M este numărul de arce, utilizând o coadă de priorități. |

Algoritmul Dijkstra este esențial pentru determinarea celor mai scurte drumuri într-un graf cu ponderi pozitive, fiind utilizat frecvent în aplicații de navigație și sisteme de rezervare. Acesta funcționează prin explorarea nodurilor cele mai apropiate de sursă, actualizând continuu distanțele minime cunoscute. Un avantaj major al algoritmului Dijkstra este capacitatea sa de a oferi soluții rapide și eficiente pentru grafuri mari și sparse, datorită utilizării unei cozi de priorități.

Algoritmul Bellman-Ford, pe de altă parte, este cunoscut pentru abilitatea sa de a gestiona grafuri cu ponderi negative, fiind capabil să detecteze cicluri negative. Acesta funcționează prin relaxarea tuturor muchiilor de (N-1) ori, unde (N) este numărul de noduri, asigurându-se că toate distanțele minime sunt corect calculate. Un avantaj al algoritmului Bellman-Ford este eficiența sa în fața grafurilor cu ponderi negative, ceea ce îl face util în anumite aplicații financiare și de rețea.

Algoritmul Floyd-Warshall este utilizat pentru a găsi cele mai scurte drumuri între toate perechile de noduri într-un graf. Acesta funcționează prin actualizarea unei matrice de distanțe, comparând toate perechile de drumuri posibile și actualizând distanțele minime. Un avantaj al algoritmului Floyd-Warshall este simplitatea implementării sale și capacitatea de a gestiona grafuri cu ponderi negative, deși nu poate gestiona cicluri negative.

Totodată, se mai folostește un program pentru a compara algoritmii și eficiența acestora, regăsit în Anexa A. Din rezultatele codului din Anexa A, se obține analiza empirică a algoritmilor plasată în tabelul 2.2, bazat pe timpul de execuție și numărul total de operații executate, bazată pe 44 de noduri și 200 de arce.

**Tabelul 2.2 - Rezultalele analizei empirice**

|  |  |  |
| --- | --- | --- |
| **Caracteristici**  **Algoritmul** | **Timpul de execuție** | **Numărul de operații** |
| **Dijkstra** | 0.000454700 sec | 4066 |
| **Bellman-Ford** | 0.000169500 sec | 8674 |
| **Floyd-Warshall** | 0.003224000 sec | 95822 |

Conform Tabelului 2.2, se observă cum algoritmul Floyd-Warshall este eficient pentru calcularea distanțelor minime între toate perechile de noduri. Totuși, în cazul unei rețele mari de zboruri, acest algoritm devine rapid ineficient. Timpul de execuție și numărul de operații cresc rapid pe măsură ce dimensiunea grafului se mărește, ceea ce poate duce la probleme de performanță. De exemplu, în cazul a 44 de aeroporturi, utilizarea acestui algoritm va genera un număr semnificativ de operații, ceea ce afectează viteza de procesare și memoria ocupată.

În contrast, algoritmul Bellman-Ford este capabil să gestioneze grafuri cu arce negative. Deși oferă un timp de execuție acceptabil pentru grafuri mai mici, devine ineficient pe măsură ce numărul de zboruri crește. Numărul de operații efectuate de Bellman-Ford crește semnificativ cu fiecare arc adăugat, ceea ce poate duce la întârzieri considerabile. Această limitare îl face mai puțin potrivit pentru rețelele de zboruri în expansiune, unde adăugarea constantă de zboruri poate exagera problemele de performanță.

Pe de altă parte, algoritmul **Dijkstra** se dovedește a fi o soluție mai eficientă în contextul rețelelor de zboruri. Deși poate necesita un timp de execuție mai lung în anumite situații, numărul de operații efectuate este semnificativ mai mic comparativ cu cele două algoritmi anterior menționați. Dijkstra se concentrează pe găsirea celui mai scurt drum de la un nod sursă la toate celelalte noduri, minimizând astfel resursele consumate. Această eficiență îl face ideal pentru gestionarea rețelelor de zboruri, unde fiecare zbor reprezintă o operațiune costisitoare.

Comparând cei trei algoritmi, este evident că fiecare are propriile sale puncte forte și slabe. Floyd-Warshall excelează în furnizarea unei soluții complete pentru distanțele între toate perechile, dar suferă din cauza complexității sale ridicate. Bellman-Ford este util în cazul grafurilor cu arce negative, dar numărul său crescut de operații îl face ineficient în scenarii cu multe zboruri. Dijkstra, deși poate necesita mai mult timp pentru a rula în anumite condiții, minimizând numărul de operații, își dovedește superioritatea în rețele mari, făcându-l alegerea preferată.

Alegerea algoritmului potrivit pentru o rețea extinsă de zboruri este importantă pentru asigurarea unei performanțe optime. Această analiză subliniază importanța alegerii corecte a algoritmului în funcție de specificul problemei și de cerințele sistemului, asigurând astfel o performanță eficientă și stabilă în creștere a sistemului.

## **2.3 Argumentarea alegerii algoritmilor**

Algoritmul Dijkstra este unul dintre cei mai cunoscuți algoritmi pentru găsirea celui mai scurt drum într-un graf cu ponderi pozitive. În comparație cu algoritmii Bellman-Ford și Floyd-Warshall, Dijkstra oferă anumite avantaje distincte, care îl fac preferabil în multe scenarii. În această analiză, vom compara în detaliu algoritmul Dijkstra cu Bellman-Ford și Floyd-Warshall, evidențiind punctele forte și limitările fiecăruia.

Unul dintre principalele avantaje ale algoritmului Dijkstra este eficiența sa. Implementarea simplă a algoritmului are o complexitate relative joasă. Aceasta îl face foarte eficient pentru grafuri sparse, adică grafuri cu relativ puține muchii în comparație cu numărul de noduri.

Performanța algoritmului Dijkstra este de asemenea remarcabilă. Acesta este excelent pentru găsirea celui mai scurt drum de la un nod sursă la toate celelalte noduri din graf. Această caracteristică este esențială în sistemele de rezervare a biletelor de avion, unde trebuie să determini rapid și eficient rutele optime pentru utilizatori.

Diversivitatea algoritmului Dijkstra este un alt punct forte. Acesta funcționează bine pentru grafuri mari cu multe noduri și muchii, utilizând eficient memoria, mai ales în implementările optimizate. În schimb, algoritmul Floyd-Warshall necesită mai multă memorie, deoarece utilizează o matrice de distanțe între toate perechile de noduri, ceea ce poate deveni greu pentru grafuri mari. Aceasta înseamnă că Dijkstra este mai potrivit pentru aplicațiile unde resursele de memorie sunt limitate.

În ceea ce privește aplicațiile, algoritmul Dijkstra este ideal pentru sistemele de rezervare a biletelor de avion, unde este necesar să găsești cel mai scurt sau cel mai ieftin drum între orașe. De asemenea, este utilizat frecvent în rețelele de transport pentru optimizarea rutelor și în aplicațiile de navigație GPS pentru calcularea rutelor optime. Algoritmul Bellman-Ford, pe de altă parte, este utilizat în probleme unde ciclurile negative sunt posibile și trebuie detectate, dar acest lucru nu este relevant în contextul rezervării biletelor de avion, unde costurile sunt întotdeauna pozitive. Algoritmul Floyd-Warshall este utilizat în probleme unde este necesar să se cunoască distanțele între toate perechile de noduri, dar acest lucru este rar necesar în sistemele de rezervare a biletelor de avion. Cu atât mai mult algoritmul Dijkstra găsește rute trecând prin mai multe orașe, nu doar printr-un singur aeroport, demonstrând adaptibilitateasa la condițiile în care nu există zboruri directe în orice direcție.

Algoritmul Dijkstra demonstrează un avantaj semnificativ prin simplitatea implementării sale în comparație cu alte metode de găsire a celui mai scurt drum. Fiind ușor de adaptat pentru diverse tipuri de grafuri, acesta permite inginerilor să construiască soluții eficiente fără a introduce o complexitate inutilă. Această accesibilitate îl face ideal pentru utilizarea în proiecte variate, de la aplicații mici la sisteme complexe.

Un alt aspect notabil al succesului algoritmului Dijkstra este abilitatea sa de a gestiona rapid modificările într-un graf. În scenariile în care conexiunile dintre noduri se schimbă frecvent, cum ar fi într-o rețea de transport, algoritmul poate recalcula rutele optime fără a necesita refacerea întregului proces de la zero. Această flexibilitate îl face de neînlocuit în medii dinamice.

Dijkstra excelează și prin compatibilitatea cu structuri de date moderne, precum cozi de priorități și arbori Fibonacci, ceea ce permite optimizări suplimentare. Aceste implementări mai avansate pot reduce și mai mult timpul de execuție, crescând astfel eficiența în rezolvarea problemelor complexe. Acest lucru demonstrează adaptabilitatea algoritmului la tehnologii noi. Pentru a asigura un viitor dezvoltat aplicației și proprietatea de a fi disponibil la modificări punctul dat reprezintă un plus foarte mare pentru dezvoltatori și companiile aeriene. Nu e nevoie de un extra cost pentru câteva schimbări în implementarea noilor propuneri și idei.

Algoritmul este adesea preferat datorită stabilității sale în aplicații practice. Spre deosebire de alte metode care pot deveni impracticabile în anumite cazuri, Dijkstra funcționează în mod excelent atât pentru grafuri mari, cât și pentru cele mici. Această consistență în performanță contribuie la reputația sa ca soluție de încredere pentru problemele legate de grafuri. [13]

Pe lângă performanțele sale remarcabile, algoritmul Dijkstra este apreciat și pentru excelența sa în gestionarea erorilor. Chiar și în cazul grafurilor incomplete sau al datelor parțiale, acesta poate furniza soluții valide și utile, adaptându-se rapid la informațiile disponibile. Această caracteristică îl face o alegere de încredere în aplicații critice, cum ar fi navigația GPS sau sistemele de management al rețelelor. Ca și în cazul schimbărilor frecvente a datelor biletelor, unde pot apărea pe neașteptate anumite erori, acesta nu va da greș și strica tot programul, dar va continua căutarea drumurilor trecând peste erori sau informații incomplete.

Deci algoritmul Dijkstra oferă avantaje semnificative în ceea ce privește eficiența, performanța și implementarea, făcându-l o alegere excelentă pentru multe aplicații, inclusiv sistemele de rezervare a biletelor de avion.

# **3 REALIZAREA SISTEMULUI**

C++ este un limbaj ideal pentru dezvoltarea unui sistem de rezervare a biletelor de avion datorită performanței sale ridicate și eficienței în gestionarea resurselor. Fiind un limbaj de programare de nivel mediu, oferă acces direct la hardware și memorie, permițând optimizarea fină a codului pentru a asigura timpi de răspuns rapizi și utilizarea eficientă a resurselor. Suportul pentru programarea orientată pe obiecte facilitează modelarea complexă a datelor și a comportamentelor, esențială pentru gestionarea rezervărilor și a rutelor de zbor. [14]

Biblioteca Standard Template Library oferă structuri de date și algoritmi eficienți, care simplifică dezvoltarea și îmbunătățesc performanța aplicației. Suportul pentru programarea concurentă prin intermediul thread-urilor permite dezvoltarea de aplicații multi-threaded, esențiale pentru gestionarea simultană a multiplelor cereri de rezervare. [15]

C++ combină puterea și flexibilitatea necesare pentru a dezvolta un sistem de rezervare a biletelor de avion performant, fiind totodată potrivit pentru o gamă largă de alte aplicații critice. Aceste caracteristici fac o alegere excelentă pentru dezvoltatorii care doresc să creeze aplicații robuste și eficiente. Flexibilitatea și puterea sa îl fac potrivit pentru proiecte de orice dimensiune.

Tot el este un limbaj excelent pentru dezvoltarea interfețelor grafice datorită flexibilității și performanței sale. Utilizarea bibliotecii MyForms permite dezvoltatorilor să creeze aplicații GUI complexe și atractive. MyForms oferă un set complet de instrumente pentru dezvoltarea interfețelor grafice, inclusiv suport pentru evenimente, widget-uri personalizabile și integrare multiplatformă, ceea ce înseamnă că aplicațiile pot rula pe Windows, macOS și Linux fără modificări semnificative ale codului.

MyForms facilitează crearea de interfețe utilizator prietenoase și responsive, oferind o gamă largă de controale și funcționalități. Aceasta permite dezvoltatorilor să creeze interfețe modulare și reutilizabile, permițând organizarea și gestionarea eficientă a codului. Suportul pentru programarea orientată pe obiecte în C++ facilitează dezvoltarea de aplicații GUI eficiente.

Limbajul dat permite, de asemenea, integrarea ușoară cu alte limbaje și tehnologii, ceea ce este esențial pentru dezvoltarea de aplicații moderne. Performanța ridicată este importantă pentru aplicațiile GUI care necesită răspunsuri rapide și eficiente la interacțiunile utilizatorilor. MyForms, combinat cu puterea și flexibilitatea C++, asigură dezvoltarea de aplicații GUI de înaltă calitate.

Librăriile acestuia oferă un set mare de funcționalități predefinite care simplifică dezvoltarea de aplicații complexe și eficiente. În proiectul dat, pentru realizarea cu succes a acestuia, s-au utilizat mai multe librării, și anume: <iostream>; <string>; <fstream>; <sstream>; <unordered\_map>; <unordered\_set>; <algorithm>; <iomanip>; <vector>; <ctime>; <limits>; <numeric>.

Acest program C++ este conceput pentru a gestiona biletele de călătorie între diverse orașe, utilizând o serie de librării standard pentru a facilita manipularea datelor, citirea și scrierea fișierelor, precum și implementarea algoritmilor de căutare și sortare. Librăria <iostream> este esențială pentru input și output standard, permițând interacțiunea cu utilizatorul prin intermediul consolei. Pentru manipularea stringurilor, programul se bazează pe <string>, care oferă funcționalități extinse pentru gestionarea textului.

Citirea și scrierea fișierelor sunt realizate cu ajutorul librăriei <fstream>, care permite deschiderea, citirea și scrierea datelor în fișiere, facilitând astfel persistența datelor. Manipularea stringurilor ca fluxuri de date este posibilă datorită librăriei <sstream>, care permite conversia și procesarea eficientă a datelor textuale. Pentru stocarea perechilor de orașe și a distanțelor dintre ele, programul utilizează <unordered\_map>, o structură de date eficientă pentru asocieri cheie-valoare. În plus, <unordered\_set> este folosită pentru a păstra o listă de elemente unice, cum ar fi orașele vizitate, asigurând astfel că nu există duplicări.

Algoritmii de sortare și căutare sunt implementați cu ajutorul librăriei <algorithm>, care oferă funcții standardizate pentru aceste operațiuni, asigurând performanță și eficiență. Pentru formatarea output-ului, în special afișarea prețurilor cu un anumit număr de zecimale, programul utilizează <iomanip>, care permite controlul precis asupra modului în care sunt afișate datele numerice. Stocarea biletelor de călătorie este realizată cu ajutorul <vector>, o structură de date dinamică ce permite gestionarea eficientă a colecțiilor de elemente.

Manipularea datelor și a timpului este facilitată de librăria <ctime>, care oferă funcții pentru obținerea și manipularea timpului calendaristic și a timpului de procesare. Pentru a obține limitele numerice necesare în algoritmul Dijkstra, programul se bazează pe <limits>, care furnizează valorile maxime și minime pentru tipurile de date numerice. În final, <numeric> este utilizată pentru a efectua operații numerice, cum ar fi calcularea prețului total al biletelor, asigurând astfel corectitudinea și eficiența calculelor.

## **3.1 Implementarea algoritmilor**

Implementarea sistemului include algoritmul Dijkstra de căutare a drumului pentru un graf. Funcția responsabilă de această implementare se regăsește în anexa B, și anume funcția FlightData e responsabilă de căutarea tuturor rutelor de la orașul de pornire la cel destinație.

Parametrii de intrare a funcției date sunt departureCity și destinationCity, orașul de pornire și respective destinație. Pe baza acestora se aplică algoritmul. Căutarea are loc pintr-un fișier CSV care stochează 44 de capitale a Europei și 200 de rute posibile între acestea.

Aceasta folosește trei vectori: paths, pathPrices; pathAvailability. Fiecare vector în urma trecerii grafului prin algoritmul Djikstra stochează respectiv însuși ruta, prețul total al rutei cu toate biletele incluse în ea și numărul de bilete respective. În cazul în care ruta include trecerea prin mai multe orașe și sunt disponibile un număr diferit de bilete în vectorul pathAvailability se stocheză cel mai mic număr disponibil de bilete pentru întreaga rută. Algoritmul selectează în mod repetat nodul cu cea mai mică distanță nevizitat, actualizând distanțele vecinilor săi.

Pentru a inițializa algoritmul, se creează o listă de distanțe minime pentru fiecare nod din rețea, setând valoarea nodului de plecare la zero și celelalte noduri la infinit. O coadă de priorități este utilizată pentru a selecta nodul cu cea mai mică distanță estimată de la sursă la fiecare pas al algoritmului. Această structură eficientizează procesul de identificare a drumului optim.

La fiecare iterare, algoritmul extrage nodul cu distanța minimă din coada de priorități și analizează vecinii acestuia. Pentru fiecare vecin, se calculează costul rutei trecând prin nodul curent. Dacă acest cost este mai mic decât valoarea curentă stocată pentru nodul vecin, distanța minimă este actualizată, iar vecinul este adăugat în coadă pentru procesare ulterioară. Acest proces se repetă până când nodul de destinație este procesat sau coada devine goală.

După ce algoritmul finalizează procesarea, drumul rezultat este tradus într-o listă de zboruri consecutive, fiecare având detalii despre ora de plecare, durata zborului și costul. Aceste informații sunt utilizate pentru a genera o listă completă și optimă cu toate rutele posibile. Rutele sunt afișate la utilizator și acesta poate să aleagă ruta dorită. Un lucru important aici este că pentru stocarea biletelor adăugate în coș se utilizează un vector definit global pentru a nu utiliza și transmite datele de la o funcție la alta.

O altă parte importantă a sitemului dat o reprezintă achiziția propriu-zisă a biletelor din coș. Conform anexei C aceasta conține mai mulți pași pentru verificarea datelor. Se verifică dacă numele utilizatorului este introdus corect și lungimea numărului cardului să fie de exact 8 cifre cu ajutorul parametrului cardNo.length. De asemenea după o achitare reușită prin crearea unui nou file txt se printează biletele cumpărate de utilizator, cu informațiile personale securizate.

Un alt aspect esențial al implementării este flexibilitatea. Sistemul poate fi extins pentru a integra informații suplimentare, cum ar fi timpii de tranzit în aeroporturi sau restricțiile specifice ale zborurilor de legătură. Este nevoie doar de modificat fișierul cu zboruri pentru a extinde sistemul, și nu este nevoie de schimbarea codului.

Pentru eficiență algoritmul este implementat astfel încât să funcționeze rapid pe rețele complexe, cu sute sau mii de noduri și arce. În cazul unei rețele aeriene, unde nu fiecare aeroport este conectat direct cu altele, listele de adiacență sunt preferate datorită eficienței în utilizarea memoriei.

## **3.2 Interfața sistemului**

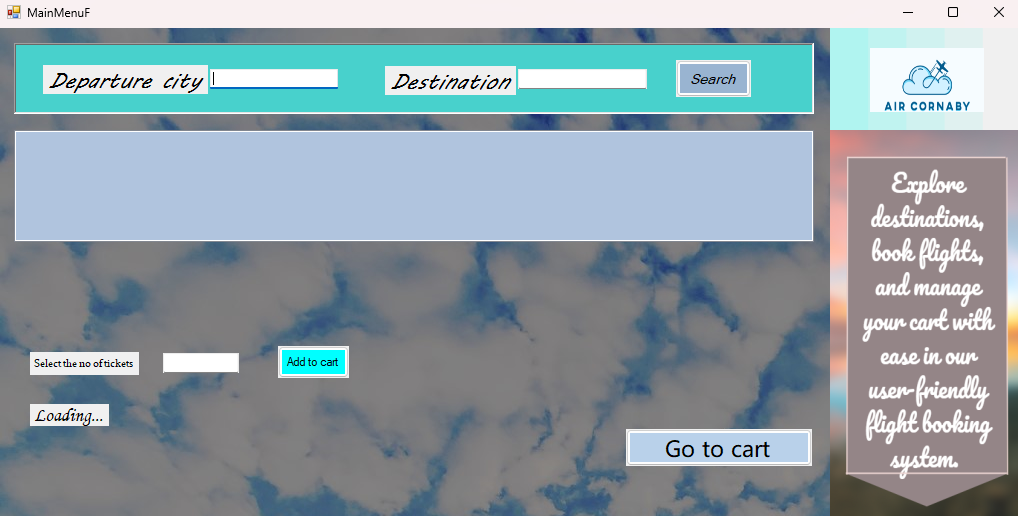
Interfața sistemului este structurată sub forma unei aplicații compuse din patru forme principale, fiecare având un rol bine definit pentru a ghida utilizatorul în procesul de navigare și utilizare a funcționalităților. Prima formă reprezintă pagina de autentificare și este punctul de intrare în sistem. Aceasta permite utilizatorului să introducă informații esențiale, cum ar fi numele, prenumele și un ID unic, pentru a se conecta la platformă.

Fereastra de start reprezentată în figura 3.1 include butoane intuitive, cum ar fi Login, care joacă rolul principal în declanșarea procedurii de autentificare. De asemenea, interfața oferă câmpuri bine structurate pentru introducerea datelor și eventuale mesaje de eroare sau confirmare, asigurând o experiență simplă și eficientă pentru utilizator.



**Figura 3.1 – Fereastra de start a platformei**

A doua formă a interfeței, conform imaginei 3.2, reprezintă pagina dedicată căutării biletelor, unde utilizatorii pot explora opțiunile de zbor disponibile între diverse destinații. Această pagină include două textbox-uri, unul pentru introducerea orașului de plecare și celălalt pentru destinație. După completarea acestor informații, utilizatorul poate apăsa butonul Search, care inițiază comunicarea cu codul din spatele aplicației. Acesta rulează algoritmul Dijkstra pentru a identifica drumurile minime între orașele selectate.

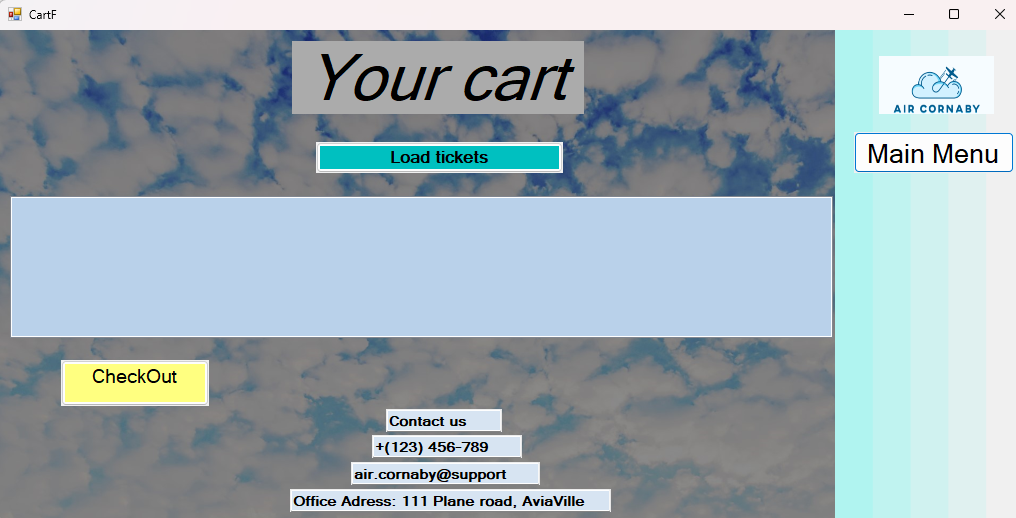


**Figura 3.2 – Fereastra MainMenu a platformei**

Rezultatele căutării sunt afișate într-un listbox, oferind o listă detaliată a rutelor disponibile, împreună cu informații precum durata zborurilor și costurile aferente. Sub această listă, interfața include un buton dedicat selectării numărului dorit de bilete și un alt buton, Select, care permite adăugarea biletelor alese în coșul de cumpărături. Aceste opțiuni permit utilizatorului să personalizeze achiziția conform nevoilor sale.

În partea inferioară a paginii se află un label informativ care afișează un mesaj dinamic despre numărul de bilete selectate, oferind claritate și confirmare imediată utilizatorului. Această organizare facilitează o interacțiune eficientă și intuitivă, ghidând utilizatorul pas cu pas prin procesul de căutare și selecție a biletelor, asigurând totodată o experiență prietenoasă și fără erori.

În figura 3.3 este prezentată pagina dedicată coșului de cumpărături. Aceasta are un design simplu și clar, menit să faciliteze gestionarea biletelor selectate de utilizator înainte de finalizarea achiziției.



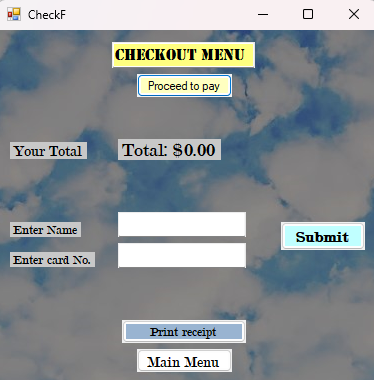
**Figura 3.3 – Fereastra Cart a platformei**

În partea de sus a paginii se află titlul „Your cart”, scris cu un font mare și vizibil, care indică funcționalitatea principală a acestei secțiuni. Sub titlu se găsește un buton albastru etichetat „Load tickets”, destinat încărcării biletelor în coș, astfel încât utilizatorul să poată verifica detaliile acestora.

Zona centrală a paginii este rezervată pentru afișarea biletelor, fiind reprezentată de un dreptunghi albastru deschis. Această secțiune este spațioasă și permite listarea mai multor bilete selectate, ceea ce o face ușor de utilizat pentru gestionarea datelor. În partea stângă jos este poziționat butonul „CheckOut”, evidențiat prin culoarea galbenă, care permite utilizatorului să finalizeze achiziția biletelor. De asemenea, în partea de jos a paginii sunt incluse informații de contact, cum ar fi numărul de telefon, adresa de e-mail și locația fizică a companiei. În colțul din dreapta sus este plasat un logo al companiei, împreună cu un buton „Main Menu” care facilitează navigarea înapoi către meniul principal.

Funcționalitatea acestei pagini este bine definită și intuitivă, permițând utilizatorului să își gestioneze eficient selecția de bilete. Prin utilizarea butonului „Load tickets,” utilizatorul poate încărca și vizualiza lista biletelor selectate, având posibilitatea de a le verifica înainte de a trece la procesul de plată. De asemenea, butonul „CheckOut” asigură o tranziție rapidă către pasul final, oferind o experiență lină și fără complicații. Această combinație de funcții facilitează un proces simplu și eficient pentru utilizatori, asigurând un grad ridicat de accesibilitate și satisfacție.

În figura 3.4 este prezentată pagina „Checkout Menu” din proiectul de rezervare a biletelor, realizată pentru a simula procesul final de plată.



**Figura 3.4 – Fereastra CheckOut a platformei**

Titlul „CHECKOUT MENU” este evidențiat, indicând utilizatorului că aceasta este secțiunea pentru finalizarea achiziției. Sub titlu se află butonul „Proceed to pay,” care permite inițierea procesului de plată și arată scopul formei date.

Pe partea centrală a paginii se află afișajul „Total: $0.00,” care informează utilizatorul despre suma totală de plată. Lângă acesta sunt plasate câmpurile de introducere a datelor utilizatorului – numele și numărul cardului – însoțite de etichetele „Enter Name” și „Enter card No.” Acestea sunt urmate de un buton „Submit,” prin care utilizatorul poate confirma datele introduse.

În partea de jos a paginii sunt două butoane suplimentare. Butonul „Print receipt” permite utilizatorului să imprime un bon pentru tranzacția efectuată, iar „Main Menu” oferă posibilitatea de a reveni la meniul principal, asigurând o navigare ușoară.

Această pagină este concepută cu o funcționalitate clară și intuitivă, oferind utilizatorului toate opțiunile necesare pentru finalizarea tranzacției într-un mod rapid și eficient.

Pagina „Checkout Menu” este dedicată finalizării procesului de plată în cadrul aplicației de rezervare a biletelor. Aceasta permite utilizatorului să introducă informațiile esențiale pentru efectuarea plății, cum ar fi numele și numărul cardului, prin intermediul unor câmpuri de text intuitive. După completarea datelor, utilizatorul poate confirma tranzacția prin apăsarea butonului „Submit.”

Totalul de plată este afișat în partea centrală a paginii, sub forma „Total: $0.00,” oferind utilizatorului claritate asupra sumei finale. Butonul „Proceed to pay” din partea superioară permite inițierea procesului de plată, oferind o tranziție rapidă către etapa următoare.

De asemenea, pagina include funcționalități suplimentare, cum ar fi posibilitatea de a imprima bonul prin butonul „Print receipt” sau de a reveni la meniul principal cu butonul „Main Menu.” Aceste opțiuni asigură o experiență completă și simplificată pentru utilizator, adresând toate nevoile esențiale legate de plată.

Aceste pagini formează împreună interfața sistemului de rezervare a biletelor, care este structurat pentru a ghida utilizatorul de la selecția biletelor până la finalizarea plății. Funcționalitățile sunt clare și intuitive, oferind utilizatorilor opțiuni de a adăuga bilete în coș, de a verifica totalul de plată și de a introduce informațiile necesare pentru efectuarea plății. Butoanele „Proceed to pay” și „Submit” sunt esențiale pentru avansarea procesului, iar opțiunile suplimentare, cum ar fi „Print receipt” și „Main Menu,” asigură o experiență completă și ușor de navigat. Sistemul este conceput pentru a facilita tranzacțiile într-un mod eficient și prietenos. Toate butoanele sunt simplu definite pentru a nu încărca utilizatorii cu informație suplimentară și a minimiza costul memoriei sistemului. Butoanele principale, care pornesc însuși procesele sunt evidențiate pentru a nu creea impresie de haos dar și de a nu duce în eroare utilizatorii.

## **3.3 Analiza rezultatelor obținute**

Proiectul realizat este un sistem de gestionare a biletelor de zbor, având ca scop simplificarea procesului de rezervare, gestionarea datelor clienților și generarea de chitanțe. Acest sistem a fost construit folosind limbajul C++ și Windows Forms pentru interfața grafică, oferind o soluție funcțională și accesibilă utilizatorilor.

Un rezultat pozitiv al proiectului este funcționalitatea completă atinsă. Utilizatorii pot interacționa cu interfața pentru a-și introduce datele, a rezerva bilete și a primi chitanțe detaliate. Datele sunt procesate rapid, iar aplicația este concepută să fie ușor de utilizat chiar și pentru persoane fără experiență tehnică. Astfel, scopurile inițiale ale proiectului au fost atinse cu succes, oferind o soluție completă și practică.

Implementarea sistemului s-a realizat într-un mod organizat și eficient, respectând principiile unei structuri modulare și ușor de întreținut. Codul a fost structurat pe clase și fișiere separate, ceea ce a permis gestionarea clară a responsabilităților fiecărei componente. Clasele Tickets, Cart și Client au fost gândite pentru a oferi o funcționalitate clară și pentru a simplifica manipularea datelor în cadrul aplicației. De asemenea, integrarea funcțiilor de procesare a fișierelor CSV a fost bine realizată, oferind un mod rapid și sigur de a salva și accesa informațiile utilizatorilor.

Interfața grafică a fost concepută pentru a fi intuitivă și ușor de folosit, utilizând Windows Forms pentru a crea un mediu prietenos cu utilizatorul. Butoanele, câmpurile de text și etichetele au fost poziționate logic, iar mesajele afișate utilizatorilor sunt clare și informative. De exemplu, utilizatorii primesc mesaje de confirmare sau notificare în cazul unor erori, ceea ce contribuie la o experiență pozitivă. De asemenea, integrarea între interfața grafică și logica din spate a fost realizată fără probleme majore, demonstrând o înțelegere solidă a principiilor de dezvoltare software.

Sistemul a fost testat în mai multe scenarii, iar rezultatele au fost satisfăcătoare. Funcționalitățile principale, cum ar fi adăugarea clienților noi, verificarea celor existenți, rezervarea biletelor și generarea chitanțelor, au fost implementate corect și răspund cerințelor proiectului. Fiecare componentă funcționează conform așteptărilor, iar erorile întâlnite au fost gestionate cu succes. Performanța sistemului este optimă pentru volumul de date utilizat, iar timpii de răspuns sunt rapizi, ceea ce asigură o utilizare fără întârzieri semnificative.

Un alt aspect pozitiv al implementării este flexibilitatea sistemului. Codul scris permite adăugarea ușoară a unor noi funcționalități în viitor, cum ar fi integrarea unei baze de date sau a unor tehnologii de autentificare. Modularitatea designului și separarea clară a responsabilităților fac ca întreținerea și extinderea aplicației să fie simplificate, ceea ce reprezintă un avantaj major pentru viitor. Astfel, se poate afirma că sistemul este nu doar funcțional, ci și bine pregătit pentru adaptările viitoare, ceea ce îi asigură relevanța pe termen lung.

Utilitatea algoritmului a fost evidentă mai ales în cazul gestionării unor grafuri complexe, unde fiecare nod reprezenta o locație, iar fiecare muchie era un zbor disponibil cu un cost asociat. Prin aplicarea Dijkstra, sistemul a putut oferi utilizatorului cea mai bună opțiune de zbor, luând în calcul constrângeri precum bugetul și preferințele de itinerariu. De asemenea, performanța algoritmului a fost remarcabilă, reușind să proceseze datele în timp real, ceea ce a contribuit la o experiență fluidă și satisfăcătoare pentru utilizatori.

Proiectul are potențialul de a fi extins și îmbunătățit, iar prin adaptările necesare, poate deveni o platformă utilă și apreciată. De asemenea, prin utilizarea unor tehnologii mai avansate, sistemul poate câștiga în popularitate și eficiență, devenind o soluție de încredere pentru utilizatori.

# **CONCLUZII**

Proiectul realizat în cadrul acestei lucrări a urmărit dezvoltarea unui sistem software pentru gestionarea biletelor de avion, capabil să răspundă nevoilor utilizatorilor prin funcționalități intuitive și eficiente. Scopul principal al proiectului a fost atins, iar soluția creată permite utilizatorilor să efectueze rezervări, să primească chitanțe detaliate și să acceseze informații rapid și fără dificultăți. Interfața grafică a fost concepută astfel încât să fie accesibilă chiar și pentru cei fără experiență tehnică, asigurând o utilizare simplă și plăcută.

Dezvoltarea proiectului a implicat mai multe etape care au consolidat atât cunoștințele tehnice, cât și cele practice. Procesul de implementare a fost unul bine structurat, punând accent pe separarea responsabilităților în cadrul aplicației. Clasele care stau la baza sistemului, precum Tickets, Cart și Client, au fost create astfel încât să simplifice gestionarea datelor și să permită funcționarea clară a fiecărei componente. Testările efectuate au confirmat că principalele funcționalități, cum ar fi rezervarea biletelor și generarea chitanțelor, funcționează conform cerințelor și oferă utilizatorilor o experiență eficientă.

Pe parcursul proiectului, dificultățile întâmpinate au oferit ocazii valoroase de învățare. Gestionarea fișierelor CSV a reprezentat o provocare din cauza limitărilor acestora în cazul unui volum mare de date sau al cerințelor de securitate. De asemenea, integrarea limbajului C++ cu Windows Forms a necesitat atenție deosebită, întrucât a fost important să se îmbine corect logica de fundal cu interfața utilizatorului. În ciuda acestor provocări, fiecare problemă a fost rezolvată pas cu pas, ceea ce a dus la o mai bună înțelegere a procesului de dezvoltare software.

Un element important al proiectului a fost utilizarea algoritmului Dijkstra pentru optimizarea rutelor de zbor. Acest algoritm a permis utilizatorilor să selecteze cu ușurință cele mai avantajoase opțiuni, atât din perspectiva timpului, cât și a costurilor. Rezultatele oferite de algoritm au fost rapide și precise, chiar și în cazul unor grafuri complexe. Această componentă a îmbunătățit considerabil funcționalitatea aplicației, oferind o soluție modernă și practică pentru nevoile utilizatorilor.

De asemenea, proiectul a scos în evidență câteva aspecte care pot fi îmbunătățite în viitor. De exemplu, înlocuirea fișierelor CSV cu o bază de date ar putea oferi o gestionare mai sigură și mai rapidă a informațiilor. Interfața grafică, deși funcțională, poate fi modernizată pentru a aduce un aspect mai atrăgător și pentru a îmbunătăți experiența utilizatorului. În plus, includerea unor funcționalități suplimentare, precum notificările ar face aplicația mai completă și mai competitivă.

Deci, în urma implementării acestui proiect, s-a demonstrat că un sistem bine planificat și organizat poate oferi soluții utile și practice. Codul scris este ușor de întreținut și e adaptabil, fiind pregătit pentru extinderea ulterioară. Aplicația are un potențial ridicat de dezvoltare și poate deveni, cu unele îmbunătățiri, un instrument de încredere pentru utilizatori. Experiența acumulată în acest proces reprezintă nu doar o etapă semnificativă de învățare, ci și o bază solidă pentru proiecte viitoare.

# **REFERINȚE BIBLIOGRAFICE**

[1] *What Is ICT?* <https://www.tutor2u.net/business/reference/what-is-ict> . Accessed 28 Nov. 2024.

[2] *Acasă | Moodle.org*. <https://moodle.org/>. Accessed 28 Nov. 2024.

[3] “Role of Information Technology in Business Management.” *Informative Blogs for Engineering Management Students - Sandip University*, 4 Feb. 2023, <https://www.sandipuniversity.edu.in/blog/role-of-information-technology-in-business-management/>.

[4] *The E-Commerce Boom: How Online Shopping Is Changing the World*. 26 July 2024, <https://www.go-globe.com/e-commerce-and-online-shopping-stats/>.

[5] “Over 70+ Online Travel Booking Statistics 2024.” *Condor Ferries*, <https://www.condorferries.co.uk/online-travel-booking-statistics>. Accessed 28 Nov. 2024.

[6] “Principiile POO.” *CodeGym*, 1689955871326, <https://codegym.cc/ro/groups/posts/ro.155.principiile-poo>.

[7] “Bilete de avion ieftine şi zboruri low-cost - cumpără online.” *Zbor.md*, <https://www.zbor.md>. Accessed 28 Nov. 2024.

[8] *Bilete de Avion Ieftine În Orice Direcție - Airservice.Md*. <https://airservice.md>. Accessed 28 Nov. 2024.

[9] *Avigo*. <https://avigo.md/>.

[10] “Google Algorithms Explained: Everything You Need to Know.” *WhatIs*, <https://www.techtarget.com/whatis/feature/Google-algorithms-explained-Everything-you-need-to-know>. Accessed 28 Nov. 2024.

[11] *Ce Este Algoritmul? | LINK Academy*. <https://www.link-academy.com/ce-este-algoritmul>. Accessed 28 Nov. 2024.

[12] Stefan, Munteanu. *Proiectarea Algoritmilor 2009-2010*. <https://lbi.ro/~cristiana/clasa%20XI%20/Grafuri/grafuri%20orientate/drumuri_minime_1.pdf>.

[13] “Find Shortest Paths from Source to All Vertices Using Dijkstra’s Algorithm.” *GeeksforGeeks*, 25 Nov. 2012, <https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/>.

[14] *C++ Introduction*. <https://www.w3schools.com/cpp/cpp_intro.asp>. Accessed 28 Nov. 2024.

[15] “Multithreading in C++.” *GeeksforGeeks*, 8 Jan. 2018, <https://www.geeksforgeeks.org/multithreading-in-cpp/>.

# **ANEXA A**

**Analiza empirică a algoritmilor**

const int INF = numeric\_limits<int>::max();

const int N = 44;

const int E = 200;

struct Arc {

int u, v, cost;

};

int graph[N][N];

vector<Arc> edges;

void GenerateGraph() {

srand(time(0));

for (int i = 0; i < N; ++i)

for (int j = 0; j < N; ++j)

graph[i][j] = (i == j) ? 0 : INF;

for (int i = 0; i < E; ++i) {

int u = rand() % N;

int v = rand() % N;

int cost = rand() % 100 + 1;

if (u != v) {

graph[u][v] = cost;

edges.push\_back({ u, v, cost });

}

}

}

void Dijkstra(int src, int& operations) {

vector<int> dist(N, INF);

vector<bool> visited(N, false);

dist[src] = 0;

operations += 2;

for (int i = 0; i < N; ++i) {

int u = -1;

for (int j = 0; j < N; ++j) {

operations++;

if (!visited[j] && (u == -1 || dist[j] < dist[u]))

u = j;

}

visited[u] = true;

for (int v = 0; v < N; ++v) {

operations++;

if (graph[u][v] != INF && dist[u] + graph[u][v] < dist[v]) {

dist[v] = dist[u] + graph[u][v];

operations += 3;

}

}

}

}

void BellmanFord(int src, int& operations) {

vector<int> dist(N, INF);

dist[src] = 0;

operations++;

for (int i = 0; i < N - 1; ++i) {

for (const auto& arc : edges) {

operations++;

if (dist[arc.u] != INF && dist[arc.u] + arc.cost < dist[arc.v]) {

dist[arc.v] = dist[arc.u] + arc.cost;

operations += 2;

}

}

}

}

void FloydWarshall(int& operations) {

int dist[N][N];

for (int i = 0; i < N; ++i)

for (int j = 0; j < N; ++j) {

dist[i][j] = graph[i][j];

operations++;

}

for (int k = 0; k < N; ++k)

for (int i = 0; i < N; ++i)

for (int j = 0; j < N; ++j) {

operations++;

if (dist[i][k] != INF && dist[k][j] != INF && dist[i][k] + dist[k][j] < dist[i][j]) {

dist[i][j] = dist[i][k] + dist[k][j];

operations += 2;

}

}

}

void Timp(const string& metoda, int& operations) {

auto start = chrono::high\_resolution\_clock::now();

if (metoda == "Dijkstra") {

Dijkstra(0, operations);

}

else if (metoda == "Bellman-Ford") {

BellmanFord(0, operations);

}

else if (metoda == "Floyd-Warshall") {

FloydWarshall(operations);

}

auto end = chrono::high\_resolution\_clock::now();

chrono::duration<double> durata = (end - start);

cout << "Metoda " << metoda << endl;

cout << fixed << setprecision(9) << "Timpul de executie: " << durata.count() << " sec" << endl;

cout << "Totalul de operatii executate: " << operations << endl << endl;

}

int main() {

GenerateGraph();

int operationsDijkstra = 0, operationsBellmanFord = 0, operationsFloydWarshall = 0;

Timp("Dijkstra", operationsDijkstra);

Timp("Bellman-Ford", operationsBellmanFord);

Timp("Floyd-Warshall", operationsFloydWarshall);

return 0;

}

# **ANEXA B**

**Implementarea algoritmului Dijkstra**

vector<std::tuple<std::vector<std::string>, double, int>> Tickets::FlightData(const std::string& departureCity, const std::string& destinationCity) {

cout << "\n--- Search Flights ---\n";

if (availableTickets.empty()) {

cout << "Loading available flights...\n";

Tickets::loadTickets();

if (availableTickets.empty()) {

cout << "No flights available.\n";

return {}; //

}

}

vector<vector<string>> paths;

vector<double> pathPrices;

vector<int> pathAvailability;

priority\_queue<pair<double, vector<string>>, vector<pair<double, vector<string>>>, greater<>> pq;

pq.push({ 0.0, {departureCity} });

unordered\_map<string, double> minCost;

unordered\_map<string, int> maxTickets;

minCost[departureCity] = 0.0;

maxTickets[departureCity] = INT\_MAX;

while (!pq.empty()) {

auto top = pq.top();

pq.pop();

double currentPrice = top.first;

vector<string> currentPath = top.second;

string currentCity = currentPath.back();

if (currentCity == destinationCity) {

paths.push\_back(currentPath);

pathPrices.push\_back(currentPrice);

pathAvailability.push\_back(maxTickets[currentCity]);

continue;

}

for (const auto& ticket : availableTickets) {

if (ticket.departure == currentCity) {

double newPrice = currentPrice + ticket.price;

int availableSeats = ticket.seats;

int newMinTickets = min(maxTickets[currentCity], availableSeats);

if (minCost.find(ticket.destination) == minCost.end() || newPrice < minCost[ticket.destination]) {

minCost[ticket.destination] = newPrice;

maxTickets[ticket.destination] = newMinTickets;

vector<string> newPath = currentPath;

newPath.push\_back(ticket.destination);

pq.push({ newPrice, newPath });

}

}

}

}

if (paths.empty()) {

cout << "No paths available from " << departureCity << " to " << destinationCity << ".\n";

return {};

}

vector<size\_t> indices(paths.size());

iota(indices.begin(), indices.end(), 0);

sort(indices.begin(), indices.end(), [&](size\_t i, size\_t j) { return pathPrices[i] < pathPrices[j]; });

vector<std::tuple<std::vector<std::string>, double, int>> availableRoutes;

for (size\_t rank = 0; rank < indices.size(); ++rank) {

size\_t i = indices[rank];

vector<string> selectedPath = paths[i];

double selectedPrice = pathPrices[i];

int availableTickets = pathAvailability[i];

availableRoutes.push\_back({ selectedPath, selectedPrice, availableTickets });

}

return availableRoutes;

}

# **ANEXA C**

**Metoda de cumpărare a biletelor**

System::Void CheckF::button2\_Click(System::Object^ sender, System::EventArgs^ e) {

std::string userName = msclr::interop::marshal\_as<std::string>(this->textBox2->Text);

std::string cardNo = msclr::interop::marshal\_as<std::string>(this->textBox3->Text);

if (userName.empty()) {

MessageBox::Show("Please enter a valid name.", "Error", MessageBoxButtons::OK, MessageBoxIcon::Error);

return;

}

if (cardNo.length() != 8 || !std::all\_of(cardNo.begin(), cardNo.end(), ::isdigit)) {

MessageBox::Show("Please enter a valid 8-digit card number.", "Error", MessageBoxButtons::OK, MessageBoxIcon::Error);

return;

}

std::time\_t currentTime = std::time(nullptr);

std::tm\* localTime = std::localtime(&currentTime);

std::ostringstream dateTimeStream;

dateTimeStream << (localTime->tm\_year + 1900) << "-"

<< (localTime->tm\_mon + 1) << "-"

<< localTime->tm\_mday << " "

<< localTime->tm\_hour << ":"

<< localTime->tm\_min << ":"

<< localTime->tm\_sec;

std::string timestamp = dateTimeStream.str();

std::string fileName = userName + "TicketReceipt.txt";

std::ofstream file(fileName);

if (!file.is\_open()) {

MessageBox::Show("Failed to create the file.", "Error", MessageBoxButtons::OK, MessageBoxIcon::Error);

return;

}

file << "Receipt Date and Time: " << timestamp << "\n";

file << "Hello " << userName << ",\n";

file << "Card Number (last 4 digits): \*\*\*\*" << cardNo.substr(cardNo.length() - 4) << "\n";

file << "\nTicket Information:\n";

const auto& cartItems = sharedCart.getCartItems();

if (cartItems.empty()) {

file << "No tickets in cart.\n";

}

else {

for (const auto& ticket : cartItems) {

file << "From: " << ticket.getDeparture()

<< " To: " << ticket.getDestination()

<< " Quantity: " << ticket.getQuantity()

<< " Price: $" << ticket.getPrice()

<< " Flight No: " << ticket.getFlightNo() << "\n";

}

}

double totalAmount = 0.0;

for (const auto& ticket : cartItems) {

totalAmount += ticket.getPrice() \* ticket.getQuantity();

}

file << "\nTotal: $" << totalAmount << "\n";

file << "\nSuccesful Payment! Thank you for staying with us" << "\n";

file.close();

MessageBox::Show("Ticket information saved to TicketReceipt.txt", "Success", MessageBoxButtons::OK, MessageBoxIcon::Information);

}