MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

SISTEM DE REZERVĂRI ONLINE BAZAT PE TEHNOLOGII MICROSOFT

LUCRARE DE LICENŢĂ

Absolvent: **Sabina FILIP**

Coordonator ştiinţific: **Șef lucr. ing. Cosmina IVAN**

2014

MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

DECAN,
Prof. dr. ing. Liviu MICLEA

DIRECTOR DEPARTAMENT,
Prof. dr. ing. Rodica POTOLEA

Absolvent: **Sabina FILIP**

SISTEM DE REZERVĂRI ONLINE BAZAT PE TEHNOLOGII MICROSOFT

1. **Enunţul temei:** *Scopul acestui proiect este dezvoltarea unei aplicații web care permite utilizatorilor de Internet vizualizarea și rezervarea de servicii turistice. Obiectivul principal este facilitarea organizării unei călătorii prin efectuarea de rezervări online pentru unități de cazare și mijloace de transport.*

2. **Conţinutul lucrării:** *Introducere, Obiectivele proiectului, Studiu bibliografic, Analiză și fundamentare teoretică, Proiectare de detaliu și implementare, Testare și validare, Manual de utilizare, Concluzii, Anexe.*

3. **Locul documentării**: Universitatea Tehnică din Cluj-Napoca, Departamentul Calculatoare

4. **Consultanţi**:

5. **Data emiterii temei:** 1 noiembrie 2013

6. **Data predării:** 11 Septembrie 2014

Absolvent: _____

Coordonator ştiinţific: _____

MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

Declaraţie pe proprie răspundere privind autenticitatea lucrării de licenţă

Subsemnatul(a)_____
_____,
legitimat(ă) cu _____ seria _____ nr. _____
CNP _____, autorul lucrării
_____
_____
_____elaborată în vederea susţinerii examenului de finalizare a studiilor de licență la Facultatea de Automatică şi Calculatoare, Specializarea _____ din cadrul Universităţii Tehnice din Cluj-Napoca, sesiunea _____ a anului universitar _____, declar pe proprie răspundere, că această lucrare este rezultatul propriei activităţi intelectuale, pe baza cercetărilor mele şi pe baza informaţiilor obţinute din surse care au fost citate, în textul lucrării, şi în bibliografie.

Declar, că această lucrare nu conţine porţiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislaţiei române şi a convenţiilor internaţionale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în faţa unei alte comisii de examen de licenţă.

In cazul constatării ulterioare a unor declaraţii false, voi suporta sancţiunile administrative, respectiv, *anularea examenului de licenţă*.

Data

_____

Nume, Prenume

_____

Semnătura

Cuprins

Capitolul 1. Introducere – Contextul proiectului

1.1. Contextul proiectului

Scopul acestui proiect este dezvoltarea unei aplicații web care permite utilizatorilor de Internet rezervarea camerelor de hotel din diferite orașe ale lumii, precum și rezervarea unor locuri în mijloacele de transport puse la dispoziție: avion sau autocar. Astfel, sistemul reunește aspectele necesare organizării unui sejur într-o singură aplicație web, nemaifiind nevoie ca utilizatorii să acceseze mai multe site-uri.

De-a lungul timpului, sistemele de rezervări dezvoltate pentru anumite servicii cum ar fi cazarea la un hotel sau serviciile de transport au evoluat treptat. Cu ani în urmă era absolut necesară deplasarea la sediul unei companii de turism, urmată de o discuţie cu agentul responsabil pentru rezervări. Ulterior, astfel de operații s-au putut realiza şi prin intermediul unei discuţii telefonice. Cu toate acestea, aceste soluții nu îi ofereau clientului gradul de confort necesar zilelor noastre. Astfel, agențiile de turism și-au dezvoltat sisteme informatice performante, care să acopere nevoile utilizatorilor într-un mod cât mai simplu și ușor accesibil.

Dacă rezervările realizate personal prin deplasarea la sediul companiei necesită un consum de timp, energie şi uneori de bani, rezervările online ne oferă un confort sporit, prin simplul fapt că presupun doar folosirea unui sistem cu acces la Internet, lucru ce este atât de comun în momentul de faţă. De asemenea, sistemele de rezervări din ultima perioadă, care se ocupă în principal de rezervările de camere la hoteluri, oferă clienţilor facilităţi şi pentru transport, tocmai pentru a le uşura acestora cât mai mult sarcinile organizatorice din vacanţă, principala lor grijă trebuind să fie doar modul în care îşi vor savura timpul de relaxare din vacanță.

În momentul de faţă există numeroase agenţii de turism care oferă diferite servicii de acest gen, inclusiv circuite, activități și oferte speciale, cei responsabili de aceste agenţii de turism încercând să se gândească la toate aspectele necesare unei vacanțe perfecte. Clientului îi rămâne astfel doar să aleagă un pachet de servicii, care să fie cât mai apropiat de nevoile sale. Totuşi, oamenii sunt diferiţi fiecare în parte, iar un asemenea sistem care să le gândească pe toate în locul clientului ridică semne de întrebare relativ la compatibilitatea acestuia cu un caz particular. Există un număr considerabil de cazuri în care clienţii doresc să-şi alcătuiască un program personal în vacanţă, aceştia dorind în principal doar o cazare de calitate într-o zonă placută şi un transport confortabil până în aceea locaţie, de multe ori doar dus, astfel încât data întoarcerii sa fie stabilită de ei şi în funcţie de dispoziţia şi starea acestora de la un moment dat.

1.2. Conturarea domeniului

Turismul a fost strâns conectat cu progresul tehnologiilor din ultimii ani. Fondarea sistemului de rezervări (Computer Reservation System – CRS) și a sistemului de distribuție globala (Global Distribution System – GDS) a transformat dramatic practicile în domeniul turismului.

Industria turismului s-a concentrat la început pe sistemele computerizate (CRS, GDS) pentru a crește eficiența în procesarea de informații și managementul distribuției.

În ziua de azi, Internetul și tehnologia informației sunt relevante pe toate nivelurile operative, structurale, strategice și de marketing, pentru facilitarea interacțiunii globale dintre furnizori, intermediari și consumatorii de pe întreg globul.

Oferta în sectorul industriei turismului oferă oportunități pentru expansiune în toate sensurile: geografică, de marketing, operațională. Ca rezultat al dezvoltării practicilor folosind Internetul, piața turismului s-a dezvoltat considerabil. Una din cele mai importante schimbări a fost apariția companiilor aeriene care utilizează Internetul ca un mecanism principal de distribuție pentru vânzări directe de bilete. Acest lucru le-a oferit consumatorilor oportunitatea de a cumpăra bilete la prețuri reduse direct de la operatorul de transport, amenințând astfel un întreg sistem de distribuție. De asemenea, dezvoltarea unor mari agenții de turism cum ar fi Expedia, Travelocity, Lastminute, a creat adevărate "supermarketuri" pentru turism. Ele oferă soluții integrate pentru o întreagă gamă de servicii cum ar fi ghiduri turistice, prognoza meteo și asigurări. Prin adoptarea pachetelor dinamice (abilitatea de a personaliza o vacanță prin adăugarea mai multor componente individuale la un preț redus), aceste companii amenință rolul operatorilor turistici.

Turiștii apelează din ce în ce mai frecvent la site-uri de informații turistice și în special la site-uri ce permit efectuarea de rezervări pentru cazare sau servicii de transport. Astfel, aceștia au anumite așteptări de la un astfel de site: să fie ușor de folosit, să fie util, să aibă informații relevante, să fie sigur și rapid. În urma studiilor efectuate s-a demonstrat că un design "sărăcăcios" poate duce la pierderea a 50% din potențialul de vânzări. E-turismul implică următoarele aspecte pentru consumatorii finali:

- **E-information –** turiștilor le este acordată informație specializată prin e-brochures (broșuri electronice), ghiduri turistice audio, albume foto, imagini reale sau virtuale, clipuri video, bloguri sau chiar și prin accesarea comunităților virtuale cum ar fi *Virtual Tourist* sau *The guide of virtual cities*
- **E-booking –** această funcție este folosită de hoteluri, companii de transport aerian sau companii ce închiriază mașini oferind consumatorilor servicii prin care aceștia pot efectua rezervări
- **E-payment –** consumatorii pot folosi carduri de credit, cecuri electronice când au nevoie să plătească doar câțiva bani

Tehnologia informației oferă unelte strategice pentru organizațiile de turism pentru a îmbunătăți modul lor de operare. Astfel, locul lor pe piață va fi determinat în mare parte de tehnologiile și rețelele utilizate pentru a interacționa cu clienții.

1.3. Cerințele proiectului

Proiectul de față are drept scop dezvoltarea unui sistem care să poată fi folosit de orice persoană cu acces la Internet care dorește să profite de un serviciu turistic.

Pentru a folosi serviciile puse la dispoziție, un utilizator trebuie să intre pe site-ul aplicației. Aici va putea alege ce operație să facă: el trebuie să poată să vizualizeze informații despre hotelurile din baza de date a aplicației, să efectueze căutări astfel încât să găsească un hotel care să se potrivească cerințelor sale și bineînțeles să poată rezerva camere.

De asemenea, un utilizator trebuie să poată vizualiza informații și despre mijloacele de transport puse la dispoziție de către sistem: avion sau autocar. Astfel, el poate vedea rutele zborurilor, prețul lor, data plecării și a sosirii. Aceleași informații sunt disponibile și pentru cursele de autocar. Utilizatorul trebuie să poată alege să își rezerve locuri pentru un mijloc de transport.

O altă funcționalitate a sistemului este cea de oferte de vacanță. Sistemul trebuie să pună la dispoziția utilizatorului pachete de vacanță ce includ cazare pe un anumit număr de zile și mijloc de transport pentru o destinație turistică. Un utilizator trebuie să poată vedea toate aceste pachete, să poată căuta pachetul dorit în funcție de preferințele sale și să își facă o rezervare.

Pentru a putea efectua rezervări, un utilizator trebuie să fie înregistrat în sistem. Dacă utilizatorul nu are un cont, el își va crea unul, iar dacă are deja un cont trebuie să se logheze. Sistemul trebuie să permită o logare mai ușoară și cu contul de Facebook.

Un utilizator logat poate rezerva camere de hotel și bilete pentru mijloace de transport. După efectuarea unei rezervări, sistemul va trimite un mail de confirmare cu datele rezervării. De asemenea, o rezervare poate fi anulată – cu condiția ca data check-inului în cazul hotelurilor sau a plecării în cazul mijloacelor de transport să fie cu cel puțin o săptămână înaintea anulării.

Sistemul trebuie să prezinte și o hartă, pe care utilizatorul să poată vedea informații despre localizarea fiecărei unități de cazare. De asemenea, trebuie pus la dispoziție un serviciu de direcționare până la destinație.

Pentru managementul datelor, aplicația va pune la dispoziție funcționalități pentru administratorii sistemului. Aceştia se pot loga în aplicație și pot vizualiza, adăuga, modifica și sterge date din orice tabelă a bazei de date.

1.4. Rezumatul lucrării

În capitolul 2 se va discuta despre obiectivele care au dus la dezvoltarea acestui sistem. În capitolul 3 se va discuta despre eTurism și se va prezenta un studiu efectuat asupra unor sisteme care oferă servicii de turism online urmând ca la finalul capitolului să se prezinte o comparație între aceste sisteme. În capitolul 4 se vor prezenta tehnologiile folosite pentru implementarea sistemului de față, cerințele sistemului și cazurile de utilizare. În capitolul 5 se va prezenta în detaliu modul de implementare al aplicației având în vedere toate componentele sale și modul lor de comunicare. În capitolul 6 se vor prezenta diferite teste efectuate asupra aplicației, urmând ca în capitolul 7 să se prezinte informații referitoare la modul de utilizare al aplicației. Ultimul capitol va sublinia câteva concluzii precum și câteva direcții de dezvoltări ulterioare.

Capitolul 2. Obiectivele Proiectului

2.1. Obiective principale

Obiectivul principal este crearea unei aplicații web folosind tehnologii Microsoft care să faciliteze organizarea unei călătorii prin efectuarea de rezervări pentru unități de cazare și mijloace de transport. Pentru o funcționare cât mai precisă ar fi ideal ca un număr cât mai mare de unități de cazare să se introducă în sistem, pentru a oferi utilizatorului o experiență cât mai realistă și cu adevărat folositoare.

Sistemul trebuie să permită vizualizarea informațiilor din baza de date, relevante pentru client. Fiind o aplicație destinată turismului, un utilizator poate vizualiza informații despre unitățile de cazare, zborurile cu avionul, cursele de autocar sau pachetele de vacanță (incluzând cazarea și mijlocul de transport). De asemenea, el va putea căuta informațiile dorite folosind anumite filtre de căutare.

Sistemul va permite efectuarea de rezervări pentru oricare dintre serviciile oferite, cu condiția ca utilizatorul să fie înregistrat în baza de date. Dacă acesta nu este înregistrat, i se va permite să își creeze un cont sau să se logheze cu un cont de pe o rețea socială.

Există două tipuri de utilizatori: un utilizator simplu, ce dorește să vizualizeze site-ul și posibil să își facă o rezervare și administratorul de sistem care este responsabil cu managementul datelor. Acesta trebuie să poată vizualiza, adăuga și edita datele din orice tabel al bazei de date.

2.2. Obiective specifice

Sistemul trebuie să permită căutarea și vizualizarea informațiilor într-un mod eficient, în timp cât mai rapid. Un utilizator ce nu este decis ce serviciu dorește va putea vizualiza ofertele existente:

- **Hoteluri** – se vor putea vizualiza hoteluri în funcție de numele lor, de orașul în care se află, de numărul de stele; pentru fiecare hotel se vor putea vizualiza detalii cum ar fi descrierea lor, o galerie de poze, tarife pentru camere.
- **Mijloace de transport** (avion sau autocar) – se vor putea vizualiza zborurile și cursele de autocar folosind ca și filtru de căutare orașul destinație și cel de plecare, precum și data plecării. De asemenea, se vor putea căuta două tipuri de curse: doar dus, sau dus-întors. Dacă se vor căuta bilete dus-întors, atunci filtrarea se va face și după orașul de plecare și data la care se dorește întoarcerea.
- **Pachete de vacanță** – se vor putea vizualiza informații despre ofertele de vacanță puse la dispoziția clienților; acestea se vor putea căuta folosind data plecării și destinația ca și filtre.

Pentru toate aceste servicii se va pune la dispoziție și posibilitatea de a efectua rezervări. Un client va putea rezerva camere de hotel, locuri pe o cursă aeriană sau de autocar sau locuri pentru un pachet de vacanță:

- **Hoteluri** – un client își va putea selecta data de check-in și check-out dintr-un hotel și numărul de camere pe care dorește să îl rezerve. Pe o rezervare facută la un moment dat, un client poate alege maxim 6 camere – pentru fiecare din aceste camere va avea opțiunea de a-și alege tipul camerei pe care îl dorește. Sistemul

trebuie să verifice în baza de date pentru a vedea dacă mai sunt disponibile camere de tipul cerut și să afișeze un mesaj de informare a utilizatorului. După selectarea tuturor camerelor, sistemul trebuie să calculeze suma totală pe care clientul o are de plătit pentru rezervare și să o afișeze. Dacă clientul este mulțumit de ofertă, el va putea confirma rezervarea, moment în care datele vor trebui introduse în baza de date și trimise într-un mail de confirmare clientului.

- **Mijloace de transport** (avion sau autocar) – biletele de avion sau autocar se pot rezerva într-un număr maxim de 6. După alegerea cursei dorite și selectarea numărului de locuri, sistemul trebuie să verifice în baza de date dacă mai este disponibil numărul de locuri respectiv și să afișeze un mesaj corespunzător. În cazul în care clientul este mulțumit de alegerea sa, el va confirma rezervarea. Datele vor fi introduse în baza de date și îi va fi trimis un mail de confirmare.
- **Pachete de vacanță** – un pachet de vacanță conține o rezervare și la hotel și pe un mijloc de transport. În cazul în care un client va dori un astfel de pachet de vacanță, el va trebui să selecteze un număr de persoane (maxim 6) și să își aleagă tipul camerei dorit. În cazul în care nu mai există destule locuri în mijlocul de transport sau nu există camerele dorite în unitatea de cazare, clientul va fi informat printr-un mesaj corespunzător. Dacă clientul este mulțumit de ofertă, el va putea confirma rezervarea, moment în care datele vor trebui introduse în baza de date și trimise într-un mail de confirmare clientului.

Interfața utilizator trebuie să fie cât mai prietenoasă, ușor de înțeles și folosit, astfel încât să permită utilizarea sistemului de către orice persoane.

Este necesară și o interfață pentru administratorul sistemului, care nu trebuie să fie bogată în elemente grafice, ea putând avea strict elemente care ajută administratorul să facă operațiile sale specifice de inserare, modificare și ștergere ale înregistrărilor din baza de date. Un administrator trebuie să poată modifica orice tip de date din baza de date.

Accesul la serviciul de rezervări trebuie să se faca pe baza înregistrării în sistem. Astfel, doar utilizatorii autorizați pot rezerva camere de hotel și mijloace de transport. Unui utilizator neînregistrat i se va permite doar să vizualizeze oferta site-ului.

Pentru a se înregistra, utilizatorii trebuie să își introducă datele personale. Odata înregistrat, un utilizator va putea oricând să își acceseze contul prin logare. De asemenea, un utilizator se va putea loga și cu un cont de rețea socială (Facebook). Ambele metode de logare pot fi folosite în cazul în care un client dorește să efectueze o rezervare.

Un client trebuie să își poată accesa istoricul rezervărilor făcute. Astfel, dacă el este logat, va trebui să aibă acces la contul său. Acest cont va conține informații despre toate rezervările făcute, în ordinea în care au fost făcute. De asemenea, folosind acest istoric, clientul va trebui să poată să își anuleze anumite rezervări. O rezervare se poate anula cu cel puțin o săptămână înainte de data plecării. În cazul în care clientul își anulează o rezervare, se vor șterge datele din baza de date și i se va trimite și un mail de confirmare.

Utilizatorul trebuie să poată vizualiza pe o hartă poziționarea unităților de cazare. Astfel, pentru fiecare hotel, trebuie să se poată accesa harta care să îi arate unde anume se află hotelul respectiv. De asemenea, pentru cazul în care clientul dorește să ajungă acolo cu mașina, se va oferi direcționare pentru condus. Ruta se va considera ca începând din punctul în care se află în acel moment utilizatorul – se va folosi geolocația pentru a îi determina poziția la momentul accesării site-ului, iar de acolo se va afișa ruta până la

unitatea de cazare, împreună cu un set de direcții (străzile pe care trebuie să conducă) pentru a ajunge acolo.

2.3. Concluzii

Sistemul va facilita organizarea unei vacanțe pentru utilizatorii de Internet, oferind accesul la unități de cazare din diferite orașe și țări și mijloace de transport pentru a ajunge acolo. Sistemul va fi implementat folosind tehnologii Microsoft, el punând la dispoziție servicii la care clienții se vor putea conecta.

Utilizatorii site-ului vor putea să vizualizeze informații despre unitățile de cazare existente în baza de date a sistemului, despre cursele pe două tipuri de mijloace de transport oferite (avion și autocar) și despre pachetele de vacanță ce ușurează căutarea, ele oferind direct și cazare și un mijloc de a ajunge până acolo. Dacă unui utilizator îi va plăcea o ofertă, el va putea să își facă o rezervare. Pentru a efectua rezervări, respectivul utilizator va trebui să aibă un cont în baza de date a sistemului.

Capitolul 3. Studiu Bibliografic

3.1. Turism

Turismul este o activitate ce are impact asupra vieţii culturale, sociale şi economice. Are legătură cu numeroase domenii, printre care: ocuparea forţei de muncă, dezvoltarea regională, educaţie, mediul înconjurător, protecţia consumatorului, sănătate, noile tehnologii, transport, finanţe, sisteme de taxare şi cultură.

Turismul a devenit în zilele noastre o activitate la fel de importantă ca şi cele desfăşurate în alte sectoare ale economiei (industrie, agricultură, comerţ).

O definiţie a turismului în sens contextual este: "turismul este o industrie bazată pe servicii ce include un număr de elemente tangibile şi intangibile. Elementele tangibile sunt: transportul, mâncarea şi băutura, circuitele turistice, cazare şi suveniruri; în timp ce elementele intangibile implică educaţie, cultură, aventură sau pur şi simplu evadare din cotidian şi relaxare". De altfel, o altă definiţie a turismului include şi o definiţie a turistului : "Turismul poate fi definit ca actul călătoriei în scopul recreerii, precum şi serviciile aferente acestui act. Un turist este cineva care călătoreşte cel puţin 50 de mile de casă, conform Organizaţiei Mondiale a Turismului - World Tourism Organisation"[2].

Turismul ca și industrie internațională dar și ca cel mai mare furnizor de locuri de muncă de pe planetă se mândrește cu o gamă mai largă de părți interesate eterogene decât multe alte industrii. Creşterea energetică și dezvoltarea industriei sunt, probabil, reflectate de creşterea Tehnologiilor de Informație și Comunicare (TIC). Interacţiunea accelerată şi sinergică între tehnologie şi turism din ultima vreme a adus schimbări fundamentale în industrie şi asupra percepţiilor noastre față de natura acesteia. Conform [11] TIC joacă un rol critic pentru competitivitatea organizațiilor de turism şi a destinaţiilor, precum şi pentru întreaga industrie în ansamblu. Evoluţia motoarelor de căutare, capacitatea de încărcare şi de viteza de căutare au influenţat numărul de călători în întreaga lume care utilizează tehnologia pentru a-și planifica călătoriile. TIC au schimbat radical eficienţa şi eficacitatea organizaţiilor de turism, modul în care întreprinderile operează pe piața turistică, precum şi modul în care clienții interacţionează cu agențiile de turism, clasându-se astfel printre primele 5 motive ale folosirii Internetului de către oameni.

3.2. E-Turism

Conform [6], istoria practicilor turismului online începe în anii 1970 când marile companii aeriene își dezvoltă sisteme centralizate pentru distribuția globală a produselor lor. Următorul pas a fost conectarea agențiilor de turism și a altor furnizori de servicii turistice pentru a permite intermediarilor să acceseze informația într-un mod flexibil și personalizat. Evoluția tuturor acestor sisteme a venit cu implementarea unor modele de e-business în industria turismului cu o listă de lansări ulterioare de diferite site-uri prin intermediul cărora furnizorii tradiționali doreau să stabilească o legătură directă cu clienții lor.

Anul 1996 a fost un an important pentru dezvoltarea serviciilor e-turism datorită pionierilor din acest domeniu, ce au lansat în acel an site-urile lor. Primul intermediar ce a apărut a fost Pegasus Systems care în colaborare cu un lanț hotelier a lansat *TravelWeb.com* – un portal disponibil pentru public în Martie 1996. Imediat după această

lansare, Sabre, o companie GDS cu un produs rezultat în urma colaborării American Airlines și IBM au lansat *Travelocity*. De asemenea, tot în 1996 Microsoft a lansat site-ul de rezervări *Expedia*.

La începutul anilor 1990 Priceline a lansat un nou model de business când au început să vândă bilete de avion iar în 1998, în colaborare cu Delta Airlines si-au diversificat oferta adăugând și posibilitatea de cazare la hoteluri și închiriere de mașini.

Un alt actor important a fost agentul *lastminute.com* care avea ca și obiectiv oferirea de bilete de avion și cazări în camere de hoteluri care nu s-ar fi vândut în alt caz. După șapte ani, acesta a fost cumpărat de Travelocity.

În anul 2000, *Trip Advisor*, cea mai mare comunitate pentru turiști a fost lansată, iar după 4 ani a fost achiziționată de *Expedia*. În ziua de azi este independent, cu o valoare mare de capital[2].

Diferitele portaluri și companii menționate anterior au fost asociate cu modele de e-business – ele prezintă diferite caracteristici și niveluri de inovație. Aceste caracteristici sunt ceea ce le fac diferite de alte companii.

Impactul TIC asupra industriei turismului sugerează că e-turismul reflectă digitizarea tuturor proceselor din industria turismului, călătoriilor, ospitalieră și de catering. Se include astfel principiul de e-commerce pentru a maximiza eficiența și eficacitatea organizațiilor de turism. De asemenea, e-turismul revoluționează toate procesele de business, întregul lanț de valori și relațiile strategice dintre organizațiile de turism și stakeholderi (părți interesate). Conform [2], e-turismul determină nivelul de competitivitate al organizațiilor profitând de rețele intranet pentru a reorganiza procesele interne, de rețele extranet pentru a face tranzacții cu partenerii și Internetul pentru a interacționa cu toți stakeholderii și clienții.

TIC nu numai că oferă posibilitatea de identificare, personalizare și cumpărare a produselor turistice dar oferă de asemenea suport pentru globalizarea industriei turismului prin instrumente eficiente acordate agenților de turism pentru ca aceștia să își dezvolte și distribuie oferta global.

Conceptul de e-turism include toate funcțiile de business (de exemplu e-commerce, e-marketing, e-finance, e-accounting - contabilitate, eHRM – managementul resurselor umane, eR&D – cercetare și dezvoltare) precum și e-strategy, e-planning și e-management pentru toate sectoarele din industria turismului. Astfel, e-turismul cuprinde 3 discipline distincte: management, sisteme informaționale și managementul lor și turismul[9].

Capabilitățile electronice în e-turism sunt un subset al strategiei e-business de ansamblu a unei companii. Acestea oferă posibilitatea de a lega sistemele de procesare a datelor interne și externe într-un mod eficient și flexibil. Sistemele electronice folosite în turism sunt împărțite în două categorii:

- Sisteme "front-office" – sunt folosite pentru a procesa datele și pentru a oferi rapoarte scrise sau vizuale. Acestea înregistrează turiștii în sistem și se ocupă de cazări, de vânzarea produselor și gestionează venitul.
- Sisteme pentru rezervări – pot fi operative atât pentru turiști cât și pentru agenții prin oferirea unui pachet de funcții de vânzare și informare. Structura lor modulară le permite să se conecteze la alte sectoare cum ar fi contabilitatea sau sectorul financiar. Printre alte funcționalități, aceste sisteme permit trasportul și primirea datelor de la GDS-uri.

Turiștii apelează din ce în ce mai frecvent la site-uri de informații turistice și în special la site-uri ce permit efectuarea de rezervări pentru cazare sau servicii de transport. Astfel, aceştia au anumite aşteptări de la un astfel de site: să fie uşor de utilizat, să fie util, să aibă informații relevante, să fie sigur și rapid. Potrivit [6], în urma studiilor efectuate s-a demonstrat că un design "sărăcăcios" poate duce la pierderea a 50% din potențialul de vânzări.

Conform www.internetworldstats.com, în 2012 populația surferilor de web din lume era 2.4 miliarde. De asemenea, potrivit ITB World travel trends report, internetul s-a stabilit în mod clar ca fiind modul principal de a profita de servicii de turism cu 54% dintre rezervări, cu mult peste cifra agenţiilor de turism care au ajuns la 24%[6]. Majoritatea oamenilor acceptă faptul că influența Internetului asupra comportamentului consumatorului crește pe zi ce trece și că gestionează și va gestiona modelele de marketing.

O arie importantă ce va trebui dezvoltată în viitor este cea de oferire de servicii turistice pe Internet oamenilor cu dizabilități (nevăzători sau cu deficiențe de auz). În momentul de față nu există pe piață astfel de produse. Industria ospitalieră trebuie să fie conștientă că segmentul de piață format din oamenii cu dizabilități sau oameni vârstnici este în creștere. Astfel, tehnologiile care vin în sprijinul lor – browsere vocale ce oferă asistență pentru accesul informație online – trebuie să fie în continuă dezvoltare.

3.3. Sisteme similare

3.3.1. Eximtur

Fondată în anul 1993 la Cluj Napoca, Compania EXIMTUR a cunoscut o dezvoltare continuă, extinzându-şi reţeaua de agenţii proprii în oraşe importante din România şi devenind una dintre cele mai importante companii de turism din România. Această firma și-a expus serviciile și in mediul online prin intermediul web site-ului *www.eximtur.ro*. Din punct de vedere al serviciilor oferite pentru clienții din mediul online, aici se regăsesc:

- Rezervare bilete de avion
- Rezervare hoteluri în România și în străinătate
- Vacanțe în România și în străinătate

Aplicația oferă detalii pentru fiecare dintre cele menționate mai sus, galerie de imagini (unde este cazul), vizualizare locații pe hartă, căutare într-o anumită perioadă, rezervare, opinii.

Fiecare firmă de acest tip încearcă să adauge ceva diferit față de restul. Astfel această aplicație deține o funcționalitate interesantă și anume: o hartă a lumii pe care se poate naviga cu ușurință cu ajutorul mouse-ului, alegând continentul, țara și apoi orașul sau regiunea, iar sistemul marchează pe hartă toate excursiile disponibile în acea regiune.

Sistemul nu oferă posibilitatea de a face plăți online, astfel pentru rezervare, se alege destinatia dorită, se completează un formular și se trimite acel formular la un operator al firmei, care contactează ulterior clientul. Ca și un atu al acestei aplicații s-a remarcat timpul mic de afișare al rezultatelor căutate.

3.3.2. Vivolis

Vivolis.ro este prima platformă online full mobile din România dedicată rezervărilor de servicii turistice, pentru bilete de avion, hoteluri şi pachete super city break, oferind o platformă intuitivă, utilizatorul având astfel acces la funcţiile platformei indiferent de tipul dispozitivului de pe care este accesată – desktop, laptop, tabletă sau telefon mobil. Conceptul full mobile înglobează mai multe elemente, de la accesul utilizatorului la o platformă online modulară – adaptabilă la ecranul dispozitivului de pe care este accesată – la modalităţi de căutare flexibile pentru găsirea serviciilor turistice dorite, pana la accesul la servicii de asistenţă inovative şi o gamă largă de metode de plată.

Platforma oferă mai multe tipuri de căutare, utilizatorii putând cauta şi rezerva cea mai bună ofertă pentru vacanţa lor. Astfel, prin Weekend Search, utilizatorul poate efectua o căutare simultană pentru toate weekendurile dintr-o lună calendaristică. Acest tip de căutare este disponibil momentan doar pentru hoteluri.

Pentru cei care au diverse activităţi preferate, *vivolis.ro* pune la dispoziţie căutarea în funcţie de acestea. Astfel, prin selectarea datelor de călătorie şi a activităţii preferate, platforma va genera o hartă cu hoteluri disponibile în mai multe zone ale lumii unde utilizatorii pot practica activitatea preferată. De asemenea, utilizatorul poate opta pentru căutarea în funcţie de obiectivele turistice lângă care doreşte să se cazeze sau în funcţie de bugetul pe care-l are la dispoziţie.

3.3.3. Traveo

Traveo.ro este o altă platformă de turism pentru clienţii din mediul online. Serviciile oferite sunt:

- City Break
- GetAway
- Avion
- Hotel

Site-ul prezintă funcţionalitatea GetAway – caută zboruri ce au ca loc de plecare unul din aeroporturile din România şi destinaţia fiind unul din cele 7 continente. Se poate seta când se doreşte plecarea (peste o săptămână, peste două săptămâni, peste o lună) şi bugetul (până la 1000 euro). Sistemul afişează în urma unei astfel de căutări harta cu continenul ales, iar pe hartă sunt marcate ofertele GetAway (oraşul destinaţie al zborului).

Funcţionalitatea City Break oferă posibilitatea rezervării de camere de hotel împreună cu bilete de avion, în timp ce Avion şi Hotel efectuează rezervări doar pentru bilete de avion, respectiv camere de hotel.

Sistemul oferă posibilitatea plăţii online. După efectuarea plăţii, o factură este eliberată şi trimisă la adresa de mail a clientului.

Pagina web	**eximtur.ro**	**vivolis.ro**	**traveo.ro**
Rezervare bilete avion	DA	DA	DA
Rezervare camere hotel	DA	DA	DA
Rezervare vacanțe	DA	DA	DA
Rezervare bilete croaziere	NU	DA	NU
Rezervare circuite	NU	DA	NU
Rezervare activități	NU	DA	NU
Plată online	NU	DA	DA

Tabelul 3.1 - Comparație între sisteme similare

După cum se observă din tabelul de comparație între sisteme, vivolis.ro este sistemul ce oferă cele mai multe funcționalități utilizatorilor: se pot rezerva vacanțe, croaziere, circuite, activități, camere de hotel, iar plata se poate face online. Traveo.ro si eximtur.ro nu oferă rezervări pentru circuite, croaziere și activități, dar traveo.ro oferă posibilitatea plății online în timp ce eximtur.ro vor contacta ulterior clienții pentru a discuta modul de plată.

Se observă că niciunul dintre aceste site-uri nu oferă posibilitatea călătoriilor cu autocarul. Toate pun la dispoziție doar bilete de avion.

Toate site-urile au o interfață prietenoasă și hărți interactive, astfel încât utilizatorul să aibă o experiență cât mai plăcută.

Capitolul 4. Analiză şi Fundamentare Teoretică

4.1. Tehnologii utilizate

Proiectul de față este o aplicație dezvoltată folosind tehnologii Microsoft, ce rulează pe sistemul de operare Windows. Sistemul a fost dezvoltat în mediul de programare Visual Studio 2012, iar limbajul de programare folosit este C#.

Partea de server se bazează pe librării de clase: Entity Framework 6.1.0 – soluția de acces la date recomandată de Microsoft și WCF Service - un API din cadrul .NET Framework folosit pentru construcția aplicațiilor cu arhitectură orientată pe servicii. Partea de client este implementată cu ajutorul Framework-ului ASP .NET MVC.

Baza de date a sistemului a fost realizată folosind Microsoft SQL Server.

4.1.1. Framework-ul .NET

Framework-ul .NET este un framework software dezvoltat de Microsoft care rulează pe Microsoft Windows. Este o tehnologie folosită de aplicații web și servicii web. ASP.NET este succesorul lui ASP (Active Server Pages) și beneficiază de puterea platformei de dezvoltare .NET, și de setul de instrumente oferite de mediul de dezvoltarea al aplicației „Visual Studio .NET”.

Avantajul .NET este faptul că include o librarie extinsă și furnizează interoperabilitate între limbaje (fiecare limbaj poate folosi cod scris în alte limbaje). Programele scrise pentru acest framework se execută într-un mediu software cunoscut sub numele de Common Language Runtime (CLR), o mașină virtuală (application virtual machine) care asigură anumite servicii, cum ar fi securitate, management-ul memoriei și tratarea excepțiilor. Libraria de clase și CLR-ul formează împreună framework-ul .NET.

4.1.2. ADO .NET Entity Framework

.NET Framework are propria tehnologie de acces la date, numită ADO.NET, ce constă într-un set de clase prin intermediul cărora aplicaţiile .NET se pot conecta la bazele de date sau surse de date (baze de date aflate pe servere sau fişiere locale), pot executa comenzi (în regim normal sau tranzacţional) sau gestiona şi procesa date în mod deconectat (local în aplicaţie).

Arhitectura ADO.NET este structurată pe mai multe nivele pentru a integra diferitele tipuri de baze şi surse de date. În acest scop se foloseşte modelul furnizorului de date (data provider). Un data provider reprezintă un set clase ADO.NET ce permit accesul la un anumit tip de bază de date, conectarea la o bază de date, execuţia de comenzi SQL şi preluarea rezultatelor comenzilor. Cu alte cuvinte, un provider oferă o interfaţă între sursa de date şi aplicaţia client[13].

Deşi nu aparţine de un anumit provider, un obiect DataSet reprezintă o colecţie de tabele şi relaţii păstrate în memorie sub forma unui cache şi asupra cărora se pot executa operaţii (inserare, ştergere şi actualizare) fără a fi necesară o conexiune permanentă cu sursa de date, modificările fiind efectuate printr-un obiect DataAdapter.

ADO.NET nu oferă furnizori de date generici. Fiecare furnizor de date este caracterizat de implementarea proprie a setului de obiecte (Connection, Command, DataAdapter şi DataReader), optimizate pentru tipul de RBDMS (Relational DataBase Management System) pe care îl reprezintă. În cadrul .NET sunt oferiţi furnizori pentru următoarele tipuri de baze de date: SQL Server, Oracle, OLE DB şi ODBC[7].

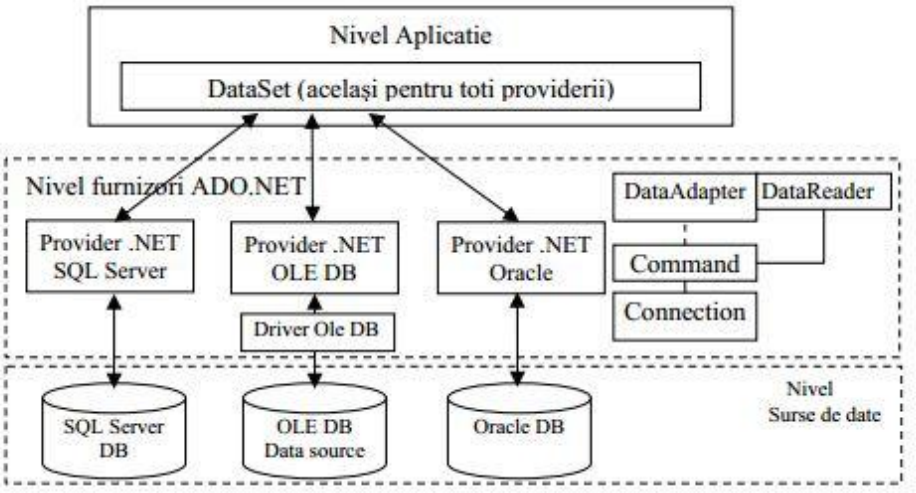

Figura 4.1 - Model simplificat al arhitecturii ADO.NET

Entity Framework este soluţia ORM (Object Relational Mapping) oferită de Microsoft. La baza Entity Framework-ului stă Entity Data Model (EDM). Acesta se vrea a fi limbajul comun între structurile de date şi modelele de prezentare. EDM-ul defineşte un limbaj menit să descrie datele fără să trebuiască să descrie modul de stocare a acestuia. Deasupra EDM-ului lucrează un set de servicii care permit manipularea datelor, spre beneficiul întregii aplicaţii.

Conform [10], Microsoft a dorit implementarea unui framework ideal pentru dezvoltarea aplicațiilor de business, un framework care să permită programatorilor să descrie logica de business și domeniul problemei pe care îl modelează fără a avea dificultăți provenite de la infrastructura ce suportă programul. De aceea, s-a ales să se folosească Entity Framework, pentru a obţine un mediu în care nivelul de prezentare este independent de nivelul de date, ceea ce oferă flexibilitate în exprimare aplicaţiei. Pentru a asigura flexibilitate la nivelul structurilor de date EDM-ul oferă o gamă variată de mapări disponibile standard.

4.1.3. ASP .NET MVC

Framework-ul ASP. NET MVC este o alternativă pentru ASP .NET ce implementează șablonul arhitectural Model-View-Controller (MVC). Acesta separă o aplicație în trei componente principale: modelul (Model), vizualizarea (View) și controller-ul (Controller). Framework-ul ASP.NET MVC este integrat cu trăsăturile existente ale ASP.NET[10].

- Models - Obiectele model sunt părți ale aplicației ce implementează logica pentru domeniul de date al aplicației. Adesea, obiectele model regăsesc și memorează starea modelului într-o bază de date.
- Views - View-urile sunt componente ce afișează elementele din interfața cu utilizatorul (UI). În mod obișnuit acestea conțin textbox-uri, combobox-uri, listview-uri, etc. Logica UI aparține view-ului.
- Controllers - Controller-ele sunt elemente ce manipulează inputul de la utilizator, interacționează cu modelul și selectează un view pentru a afișa rezultatele. Interacțiunea cu utilizatorul este gestionată de controller.

Cel mai mare beneficiu, adus de ASP .NET MVC, în comparație cu predecesorul ASP .NET Web Forms este separarea conceptelor. Astfel, există o posibilitate mai mică pentru supra-complicarea codului. De asemenea, permite reutilizabilitate (un controller poate fi folosit pentru mai multe view-uri, nu depinde de unul singur) si testabilitate (un controller fiind o clasa separată se poate testa). Framework-ul are o performanță mai bună, suport deplin pentru HTML și permite programare paralelă (un dezvoltator poate lucra la un controller, unul poate lucra la view-uri, acestea fiind slab cuplate).

4.1.4. Windows Communication Foundation

Windows Communication Foundation (sau WCF) este un API din cadrul .NET Framework folosit pentru construcția aplicațiilor cu arhitectură orientată pe servicii (tip de arhitectură software care presupune distribuirea funcționalității aplicației în unități mai mici, distincte - numite servicii - care pot fi distribuite într-o rețea și pot fi utilizate împreună pentru a crea aplicații).

WCF a fost proiectat în acord cu principiile SOA pentru implementarea programării distribuite, unde serviciile sunt consumate de către consumatori. Clienții pot consuma mai multe servicii, și serviciile pot fi, la rândul lor, consumate de mai mulți clienți. Serviciile sunt slab conectate între ele. De-obicei, serviciile oferă o interfață WSDL, pe care orice client WCF o poate folosi, pentru a consuma serviciul, indiferent de platforma pe care este găzduit serviciul. WCF implementează mai multe standarde avansate pentru servicii web, cum ar fi WS-Addressing, WS-ReliableMessaging și WS-Security. Odată cu lansarea .NET Framework versiunea 4.0, WCF oferă și servicii pentru utilizarea familiei de formate RSS (Really Simple Syndication)[14].

Un serviciu WCF este compus din 3 părți — o clasă *Serviciu* care implementează serviciul care va fi oferit, un mediu gazdă, pentru a găzdui serviciul, și unul sau mai multe *endpoints* la care clienții se vor conecta. Toată comunicația cu serviciul WCF se derulează prin intermediul endpoint-urilor. Endpoint-urile specifică un *Contract* care definește care metode din clasa Serviciu vor fi accesibile prin intermediul endpoint-ului; fiecare endpoint poate expune un set diferit de metode. În concluzie, endpoint-urile definesc o *legătură* care specifică cum va comunica un client cu serviciul și *adresa* unde este găzduit *endpoint*-ul. Pentru găzduirea serviciului WCF se poate folosi IIS, sau se poate autogăzdui în orice proces, folosind clasa ServiceHost, care este furnizată de către WCF. Un serviciu WCF auto-găzduit poate fi pus la dispoziție de către o aplicație-consolă, o aplicație Windows Forms, sau un serviciu Windows.

Un client WCF se conectează la un serviciu WCF folosind un *endpoint*. Fiecare serviciu își expune contractele prin intermediul unuia sau mai multor *endpoint*-uri. Un *endpoint* are o adresă, care este un URL specificând unde poate fi accesat *endpoint*-ul, și parametrii de conexiune, care specifică transferul de date.

Prescurtarea "ABC" poate fi folosită pentru memorarea noțiunilor Address / Binding / Contract (Adresa / Conexiune / Contract). Conexiunea specifică ce protocoale de comunicare sunt folosite pentru accesarea serviciului, dacă anumite mecanisme de securitate sunt necesare, și alte informații similare. WCF include conexiuni predefinite pentru cele mai comune protocoale de comunicare, cum ar fi SOAP peste HTTP, SOAP peste TCP, și SOAP peste Message Queues, etc. Interacțiunea dintre *endpoint*-ul WCF și client este realizată folosind formatul SOAP[15].

Când un client dorește să acceseze serviciul prin intermediul unui endpoint, nu îi este suficientă doar cunoașterea contractului, ci mai este necesară și acceptarea tipului de legătură, specificată de endpoint. Deci atât clientul, cât și serverul trebuie să aibă endpoint-uri compatibile.

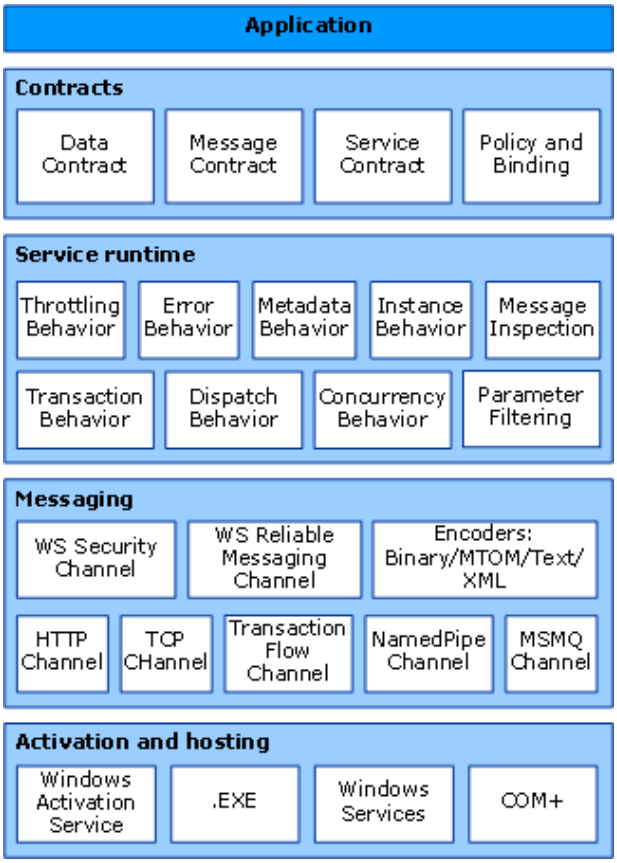

Figura 4.2 - Arhitectura WCF

Prin folosirea acestui framework se obține un grad de flexibilitate și portabilitate mai mare decât prin folosirea serviciilor web tradiționale.

Serviciile web ASP .NET au fost dezvoltate pentru construirea aplicațiilor ce trimit și primesc mesaje folosind SOAP (Simple Object Access Protocol) și protocolul HTTP. Structura unui mesaj poate fi descrisă printr-un fișier XML. Este furnizat un tool pentru facilitarea serializării mesajelor către și de la obiecte ale framework-ului .NET. Tehnologia poate genera in mod automat metadate pentru a descrie serviciile web în WSDL (Web Services Description Language). De asemenea, se mai furnizează un tool pentru generarea clienților pentru serviciile web folosind WSDL.

S-a ales să se folosească WCF și nu servicii web pentru facilitarea transmiterii de mesaje ale aplicațiilor framework-ului .NET cu alte entități software. SOAP este folosit în mod implicit; cu toate acestea, mesajele pot avea orice format și transmise folosind orice protocol de transport. Structura mesajelor poate fi definită printr-un XML și există o serie de opțiuni pentru serializarea mesajelor. WCF poate genera automat metadate pentru descrierea aplicațiilor folosind tehnologia din WSDL și pune la dispoziție un tool pentru generarea clienților. WCF este mai flexibil deoarece serviciile sale pot fi găzduite în diferite tipuri de aplicații: IIS,WAS, Self-hosting, Managed Windows Service.

4.1.5. Microsoft SQL Server

Microsoft SQL Server este un sistem de gestionare de baze de date relaționale (RDBMS) produs de Microsoft. Limbajul principal de interogare este T-SQL.

O bază de date tipică are două componente: fișierele care conţin baza de date fizică şi softul Database Management System (DBMS) pe care aplicaţiile îl folosesc pentru a accesa date. DBMS este responsabil pentru aplicarea structurii bazei de date cuprinzând:

- Întreţinerea relaţiilor între datele din baza de date
- Asigurarea că toate datele sunt stocate corect și că nu sunt violate regulile care definesc relaţiile dintre date
- Recuperarea tuturor datelor până la un punct în cazul căderii sistemului

Una dintre principalele caracteristici a aplicaţiilor de baze de date constă în faptul că accentul este pus pe operaţiile de memorare si regăsire. Principala operaţie ce apare în orice aplicaţie de baze de date este aceea de regăsire a datelor. Alături de operaţiile de regăsire mai apar de asemenea și operații de memorare, folosite la introducerea de noi date în baza de date, operaţii de ștergere și actualizare a unor date deja existente în baza de date[8].

Această metodă de centralizare a datelor într-o bază de date, conferă o serie de avantaje considerabile precum:

- Reducerea redundanței - modul de centralizare a datelor este ușor de urmărit și structurat astfel încât diferite aplicații ce folosesc în comun aceleași date, să utilizeze în comun un singur fișier pentru stocarea acestora, în scopul folosirii unui spațiu de memorie minimal
- Evitarea inconsistenței datelor – faptul că ar putea exista mai multe copii ale aceleeași date în cadrul unei baze de date, va îngreuna procedeul de actualizare, și totodată permite crearea unor situații în care apar valori

diferite pentru aceeași dată, în situația unor actualizări incomplete, ceea ce va atrage după sine evident o inconsistență a bazei de date

- Partajarea datelor – buna structurare a bazei de date ne permite utilizarea în comun a datelor, de către mai multe aplicații, dar și dezvoltarea unora noi, folosind datele deja existente
- Menținerea integrității datelor – se referă la corectitudinea datelor, ce reiese atât din consistența acestora, dar și din existența funcțiilor de validare care să asigure introducerea în baza de date a unor date corecte.[8]

S-a folosit Microsoft SQL Server deoarece Entity Framework are suport integrat pentru acest sistem de gestiune a bazelor de date. Comunicarea între EF și SQL Server este ușor de configurat.

4.2. Cerințe

Analiza și definirea cerințelor stabilește funcțiile, constrângerile, obiectivele sistemului prin consultarea cu utilizatorii. Acestea trebuie scrise în așa fel încât să fie înțelese atât de utilizatori cât și de dezvoltatorii sistemului software. O cerință poate varia de la o frază abstractă care descrie un serviciu sau o constrângere de sistem până la o specificare matematică detaliată. Implementarea oricărei cerințe trebuie să contribuie la rezolvarea problemei utilizatorului.

În secțiunea cerințe funcționale se descriu serviciile furnizate către utilizator sau către alte sisteme. Cerințele funcționale pot să descrie: ce intrări trebuie să accepte sistemul, ce ieșiri trebuie să producă sistemul, ce date (care pot fi utilizate de alte sisteme) trebuie să fie stocate de sistem, ce calcule trebuie să efectueze sistemul.

În secțiunea cerințe non-funcționale se descriu constrângerile sub care trebuie să opereze sistemul sau standardele ce trebuie să fie satisfăcute de sistemul software.

4.2.1. Cerințe funcționale

Cerințele funcționale descriu funcționalitatea sistemului și a serviciilor oferite. O funcţie este descrisă ca un set de intrări şi ieşiri. Cerinţele funcţionale pot fi calcule, detalii tehnice, manipulări şi procesări de date, dar şi alte funcţionalităţi specifice care definesc ce trebuie să îndeplinească un sistem, care este scopul său.

În tabelul 4.3 este prezentată descrierea cerințelor funcționale:

Cerință	Descriere
CF 1	**Autentificare utilizatori**
CF 1.1	Logare utilizator
CF 1.2	Logare utilizator folosind rețele sociale (Facebook)
CF 1.3	Înregistrare utilizatori
CF 2	**Căutare și localizare unități de cazare**
CF 2.1	Căutarea unităților de cazare (hoteluri) folosind diferite criterii – căutare după țară, oraș, nume hotel, număr de stele
CF 2.2	Localizarea pe hartă a unităților de cazare, calcularea și afișarea rutei spre acestea
CF 3	**Căutare pachete de vacanțe**

CF 3.1	Căutarea pachetelor de vacanțe folosind ca și criteriu orașul destinație și cel de plecare
CF 4	**Efectuare de rezervări**
CF 4.1	Rezervarea unei camere de hotel
CF 4.2	Rezervarea de locuri pe un mijloc de transport – la alegere, avion sau autocar
CF 4.3	Rezervarea unui pachet de vacanță
CF 4.4	Confirmarea prin mail la efectuarea unei rezervări
CF 4.5	Anularea rezervării
CF 4.6	Vizualizarea istoricului de rezervări al unui client
CF 5	**Managementul datelor**
CF 5.1	Managementul hotelurilor făcut de către administratorii sistemului (operații CRUD pe baza de date pentru hoteluri)
CF 5.2	Managementul zborurilor cu avionul făcut de către administratorii sistemului (operații CRUD pe baza de date pentru zboruri)
CF 5.3	Managementul curselor de autocar făcut de către administratorii sistemului (operații CRUD pe baza de date pentru curse)
CF 5.4	Managementul pachetelor de vacanțe făcut de către administratorii sistemului (operații CRUD pe baza de date pentru vacanțe)

4.2.2. Cerințe non-funcționale

Cerințele non-funcționale sunt atașate celor funcționale, ele descriind constrangeri sub care trebuie sa opereze sistemul sau standarde ce trebuie sa fie satisfacute de sistemul software. Constrângerile sunt proprietăţi ale componentelor de stocare de date, reprezentări ale sistemului, etc. Cerinţele non-funcţionale pot fi mai critice decât cele funcţionale deoarece dacă nu sunt îndeplinite, sistemul este inutilizabil.

În general utilizatorii au așteptări mari în privința performanței sistemului pe care îl folosesc – acesta trebuie să aibă un timp foarte mic de răspuns, disponibilitate, capacitate și precizie. De asemenea, trebuie să fie ușor de folosit, astfel încât să nu pună utilizatorul în dificultate atunci când acesta dorește să efectueze o operație, să confere un grad de securitate și să se comporte bine în situații neașteptate (de exemplu un crash).

În tabelul 4.4 este prezentată descrierea cerințelor non-funcționale:

CNF 1 – **Fiabilitate**	Sistemul software să funcţioneze fără erori, dacă există bug-uri cunoscute acestea vor fi documentate
CNF 2 – **Robustețe**	Sistemul să se comporte bine și în situații neprevăzute – în cazul datelor invalide să continue să funcționeze
CNF 3 – **Uzabilitate**	Sistemul să aibă o interfață prietenoasă și ușor de folosit, un utilizator care va folosi pentru prima dată site-ul să înțeleagă ce trebuie să facă pentru a efectua operațiile dorite
CNF 4 – **Posibilitate de reutilizare**	Proiectul să fie realizat astfel încât în interiorul oricărui layer să existe componente reutilizabile (de exemplu repository, business library, WCF host, componente MVC)
CNF 5 –	Sistemul să fie ușor de întreținut – să existe joburi SQL care să

Posibilitate de întreținere	întreține baza de date (backup, restore)
CNF 6 - **Securitate**	Sistemul să implementeze un prim nivel de securitate – accesarea să se facă pe bază de nume de utilizator şi parolă
CNF 7 - **Recuperare**	Sistemul trebuie să aibă funcţie de restore pentru refacerea bazei de date în urma unui backup

4.3. Cazuri de utilizare

Sistemul permite efectuarea de operații de către două tipuri de utilizatori: utilizator simplu, ce vizualizează site-ul și face rezervări şi un administrator al sistemului care face operații CRUD (creare, citire, update și ștergere) pe tabelele din baza de date.

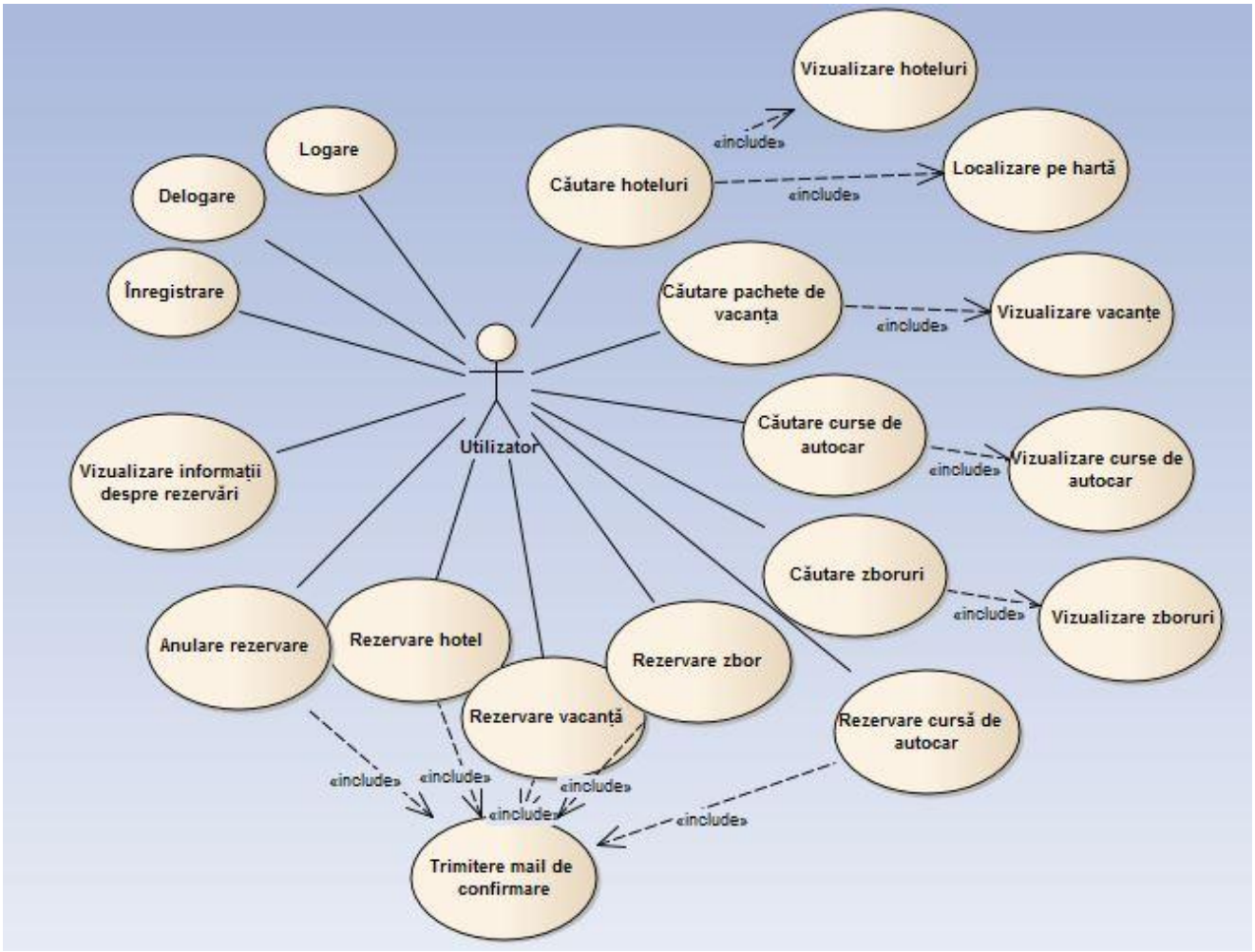

Figura 4.5 – Diagrama cazurilor de utilizare pentru un utilizator simplu al sistemului

Utilizatorul accesează site-ul, după care are posibilitatea efectuării următoarelor operații:

- Înregistrare/Autentificare
- Căutare hoteluri
- Căutare mijloc de transport
- Căutare pachet de vacanță
- Vizualizare pe hartă a unităților de cazare disponibile și a rutelor până la acestea

Utilizatorul nu poate face o rezervare dacă nu este logat în sistem. Utilizatorul autentificat beneficiază de o serie de facilități suplimentare, și anume:

- Rezervare cameră hotel și/sau mijloc de transport
- Rezervare pachet de vacanță
- Vizualizare rezervări efectuate
- Anulare rezervare

Caz de utilizare CU1: înregistrare utilizator nou
Descriere: un utilizator al sistemului dorește să se înregistreze pentru a putea beneficia de servicii
Actor primar: utilizator simplu
Precondiții: utilizatorul nu este înregistrat în baza de date a sistemului
Postcondiții: utilizatorul e înregistrat și poate efectua rezervări
Scenariu de succes: 1. utilizatorul selectează opțiunea **"register"** din meniu
2. utilizatorul introduce datele în pagina încărcată în browser
3. sistemul accesează baza de date și introduce corect informațiile noului utilizator
4. sistemul afișează un mesaj de bun venit
Scenariu de eșec: 3a. datele nu sunt introduse in baza de date
4a. sistemul afișează un mesaj de eroare și reafișează pagina de introducere a datelor

Caz de utilizare CU2: login utilizator
Descriere: un utilizator al sistemului ce are deja cont dorește să se logheze pentru a putea beneficia de servicii
Actor primar: utilizator simplu
Precondiții: utilizatorul este înregistrat în baza de date a sistemului
Postcondiții: utilizatorul e logat și poate efectua rezervări
Scenariu de succes: 1. utilizatorul selectează opțiunea **"login"** din meniu
2. utilizatorul introduce datele în pagina încărcată în browser
3. sistemul accesează baza de date și introduce corect informațiile noului utilizator
4. sistemul afișează un mesaj de bun venit
Scenariu de eșec: 3a. datele nu sunt introduse in baza de date
4a. sistemul afișează un mesaj de eroare și reafișează pagina de introducere a datelor

Caz de utilizare CU3: căutare hotel
Descriere: un utilizator al sistemului dorește să vadă unitățile de cazare disponibile; el poate aplica mai multe filtre pentru a face o căutare: poate căuta un hotel după numele său, după oraș, țară, număr de stele
Actor primar: utilizator obișnuit
Precondiții: nu există
Postcondiții: afișarea tuturor rezultatelor care corespund filtrului setat de utilizator
Scenariu de succes: 1a. utilizatorul completează câmpurile folosite pentru efectuarea căutării
1b. utilizatorul nu completează niciun câmp
2a. sistemul afișează pagina cu hotelurile care se potrivesc filtrului de căutare
2b. sistemul afișează pagina cu toate hotelurile din baza de date
Scenariu de eșec: 2c. se afișează un mesaj de informare a utilizatorului, cum că pagina cerută nu poate fi afișată

Caz de utilizare CU4: căutare pachet de vacanță
Descriere: un utilizator al sistemului dorește să vadă pachetele de vacanțe disponibile – cazare la hotel cu mijloc de transport asigurat; el poate folosi ca și filtru de căutare destinația dorită (orașul) și locul de plecare
Actor primar: utilizator obișnuit
Precondiții: nu există
Postcondiții: afișarea tuturor rezultatelor care corespund filtrului setat de utilizator
Scenariu de succes: 1a. utilizatorul completează câmpurile folosite pentru efectuarea căutării
1b. utilizatorul nu completează niciun câmp
2a. sistemul afișează pagina cu pachetele care se potrivesc filtrului de căutare
2b. sistemul afișează pagina cu toate pachetele din baza de date
Scenariu de eșec: 2c. se afișează un mesaj de informare a utilizatorului, cum că pagina cerută nu poate fi afișată

Caz de utilizare CU5: căutare mijloc de transport
Descriere: un utilizator al sistemului dorește să vadă mijloacele de transport disponibile; el poate efectua căutări pentru zboruri și curse de autocar introducând ruta dorită
Actor primar: utilizator obișnuit
Precondiții: utilizatorul e înregistrat în sistem
Postcondiții: afișarea tuturor rezultatelor care corespund filtrului setat de utilizator
Scenariu de succes: 1. utilizatorul alege una din opțiunile **"flights"** sau **"bus rides"**
2. utilizatorul introduce datele în pagina încărcată în browser – loc de plecare, destinație, data

3. sistemul afișează pagina cu mijloace de transport care se potrivesc filtrului de căutare

Scenariu de eșec: 3a. sitemul afișează un mesaj de informare a utilizatorului, cum că pagina cerută nu poate fi afișată

Caz de utilizare CU6: rezervare cameră de hotel

Descriere: după selectarea unei unități de cazare, utilizatorul poate rezerva una sau mai multe camere la unitatea respectivă

Actor primar: utilizator obișnuit

Precondiții: utilizatorul e înregistrat în sistem

Postcondiții: rezervarea este corect salvată în baza de date

Scenariu de succes: 1. utilizatorul alege opțiunea **"book now"** ce apare în dreptul unui hotel

2. utilizatorul introduce datele în pagina încărcată în browser – tipul camerei, perioada

3. sistemul accesează baza de date și introduce corect informațiile noii rezervări

4. sistemul afișează un mesaj de succes și trimite un mail de confirmare utilizatorului

Scenariu de eșec: 3a. datele nu sunt introduse in baza de date

3b. rezervarea nu poate fi făcută deoarece camera nu este disponibilă

4a. se afișează un mesaj de informare a utilizatorului, cum că rezervarea sa nu poate fi făcută

Caz de utilizare CU7: rezervare pachet de vacanță

Descriere: după selectarea unei oferte, utilizatorul poate rezerva locuri pentru una sau mai multe persoane în unitatea de cazare și mijlocul de transport

Actor primar: utilizator obișnuit

Precondiții: utilizatorul e înregistrat în sistem

Postcondiții: rezervarea este corect salvată în baza de date

Scenariu de succes: 1. utilizatorul alege opțiunea **"book now"** ce apare în dreptul unei oferte

2. utilizatorul introduce datele în pagina încărcată în browser – număr persoane, tipul camerei

3. sistemul accesează baza de date și introduce corect informațiile noii rezervări

4. sistemul afișează un mesaj de succes și trimite un mail de confirmare utilizatorului

Scenariu de eșec: 3a. datele nu sunt introduse in baza de date

3b. rezervarea nu poate fi făcută deoarece camera nu este disponibilă

4a. se afișează un mesaj de informare a utilizatorului, cum că rezervarea sa nu poate fi făcută

Caz de utilizare CU8: rezervare mijloc de transport
Descriere: pentru a ajunge la destinația dorită, utilizatorul are opțiunea de a rezerva un mijloc de transport – poate lua bilete de autocar sau avion
Actor primar: utilizator obișnuit
Precondiții: utilizatorul e înregistrat în sistem
Postcondiții: rezervarea este corect salvată în baza de date
Scenariu de succes: 1. utilizatorul alege opțiunea **"book ticket"**
2. utilizatorul introduce datele în pagina încărcată în browser – loc de plecare, destinație, număr de persoane, data
3. sistemul accesează baza de date și introduce corect informațiile noii rezervări
4. sistemul afișează un mesaj de succes și trimite un mail de confirmare utilizatorului
Scenariu de eșec: 3a. datele nu sunt introduse in baza de date
3b. rezervarea nu poate fi făcută deoarece nu mai sunt locuri disponibile
4a. se afișează un mesaj de informare a utilizatorului, cum că rezervarea sa nu poate fi făcută

Caz de utilizare CU9: vizualizare rezervări (history)
Descriere: un utilizator dorește să vadă toate rezervările pe care le-a făcut pe site
Actor primar: utilizator obișnuit
Precondiții: utilizatorul e înregistrat în sistem
Postcondiții: afișarea tuturor rezultatelor care corespund informației cerute de utilizator
Scenariu de succes: 1. utilizatorul alege opțiunea **"View booking history"**
2. sistemul afișează pagina cu toate rezervările făcute de utilizator
Scenariu de eșec: 2a. se afișează un mesaj de informare a utilizatorului, cum că pagina cerută nu poate fi afișată

Caz de utilizare CU10: anulare rezervare
Descriere: este posibil ca în urma efectuării unei rezervări, utilizatorul să dorească să o anuleze
Actor primar: utilizator obișnuit
Precondiții: utilizatorul a făcut o rezervare (de hotel, mijloc de transport sau pachet de vacanță), mai sunt cel puțin 7 zile până la data de început a rezervării sale
Postcondiții: rezervarea este ștearsă din baza de date
Scenariu de succes: 1. utilizatorul alege opțiunea "**cancel booking"** din meniul de vizualizare al propriilor rezervări
2. sistemul accesează baza de date și șterge informațiile rezervării
3. sistemul afișează un mesaj de succes și trimite un mail de confirmare utilizatorului
Scenariu de eșec: 2a. datele nu sunt șterse din baza de date
2b. rezervarea nu poate fi anulată deoarece a trecut termenul maxim pentru anulare

3a. se afișează un mesaj de informare a utilizatorului, cum că rezervarea sa nu a fost anulată

Caz de utilizare CU11: vizualizare rută spre unități de cazare
Descriere: utilizatorul poate vizualiza distanța până la unitățile de cazare; pentru această operație este pusă la dispoziție o hartă
Actor primar: utilizator obișnuit
Precondiții: utilizatorul să fie înregistrat în sistem
Postcondiții: sistemul să afișeze pe hartă ruta corectă
Scenariu de succes: 1. utilizatorul alege opțiunea **"view route"** aflată în dreptul unei unități de cazare
2. sistemul afișează o hartă pe care este marcat drumul de la poziția curentă a utilizatorului până la destinație
Scenariu de eșec: 2a. se afișează un mesaj de informare a utilizatorului, cum că pagina cerută nu poate fi afișată

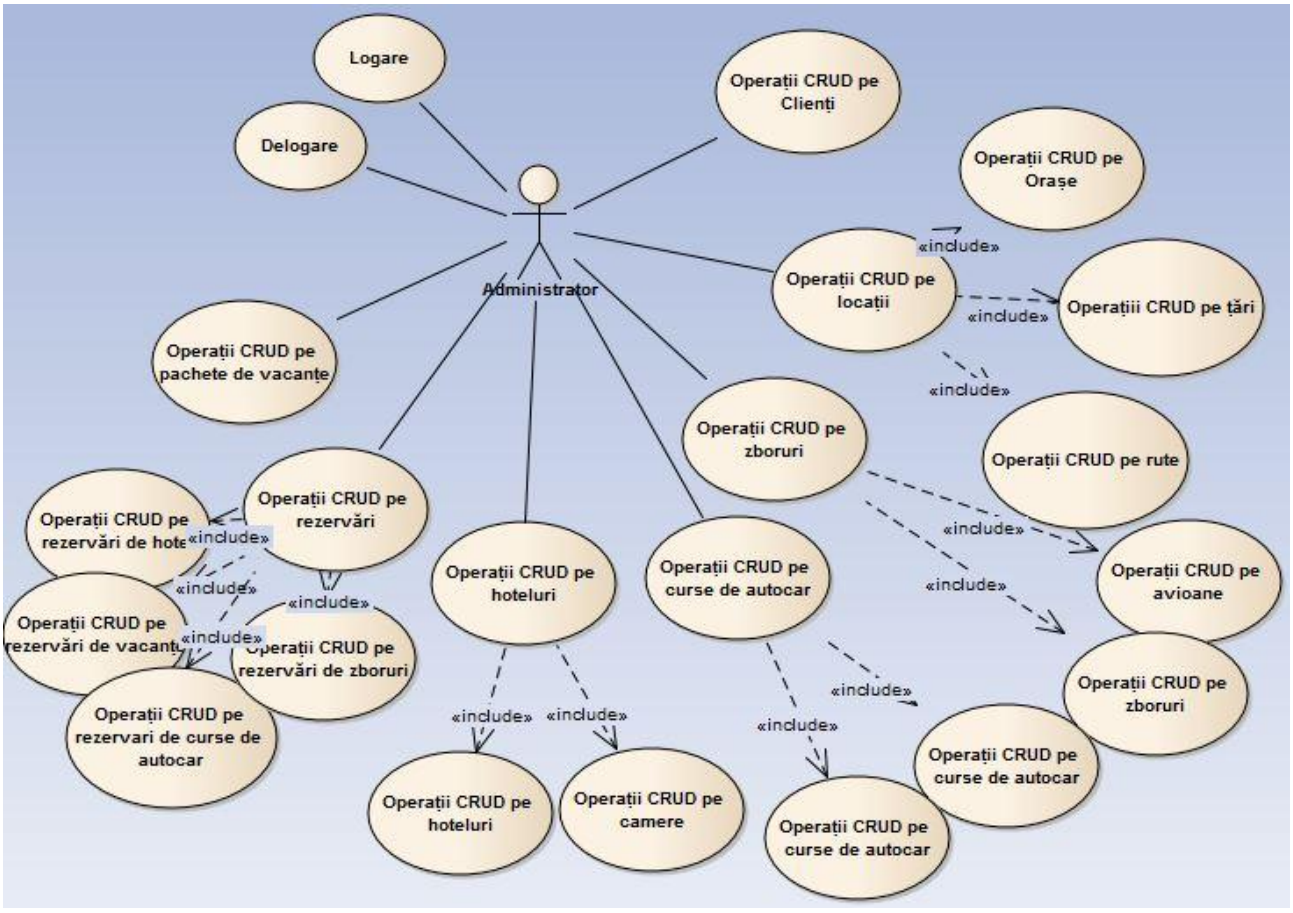

Figura 4.6 – Diagrama cazurilor de utilizare pentru un administrator al sistemului

Administratorul sistemului are acces la toate tabelele bazei de date. El poate să facă orice operații pe acestea.

Caz de utilizare: vizualizare utilizatori
Descriere: administratorul dorește să vadă utilizatorii sistemului
Actor primar: administrator sistem
Precondiții: administratorul e logat în sistem
Postcondiții: afișarea tuturor utilizatorilor existenți în baza de date
Scenariu de succes: 1. administratorul selectează opțiunea **"clients"** din meniul principal
2. sistemul afișează pagina cu toți utilizatorii din baza de date
Scenariu de eșec: 2a. se afișează un mesaj de informare a utilizatorului, cum că pagina cerută nu poate fi afișată

Caz de utilizare: adăugare utilizator
Descriere: administratorul dorește să introducă un nou utilizator în baza de date
Actor primar: administrator sistem
Precondiții: administratorul e logat în sistem; utilizatorul nu este înregistrat în baza de date a sistemului
Postcondiții: utilizatorul e înregistrat și poate efectua rezervări
Scenariu de succes: 1. administratorul selectează opțiunea **"add client"** din meniul **"clients"**
2. administratorul introduce datele în pagina încărcată în browser
3. sistemul accesează baza de date și introduce corect informațiile noului utilizator
4. sistemul afișează pagina de vizualizare a tuturor utilizatorilor
Scenariu de eșec: 3a. datele nu sunt introduse in baza de date
4a. sistemul afișează un mesaj de eroare și reafișează pagina de introducere a datelor

Caz de utilizare: editare utilizator
Descriere: administratorul dorește să editeze un utilizator din baza de date
Actor primar: administrator sistem
Precondiții: administratorul e logat în sistem; utilizatorul e înregistrat în baza de date a sistemului
Postcondiții: datele utilizatorului sunt corect modificate în baza de date
Scenariu de succes: 1. administratorul selectează opțiunea **"edit"** din dreptul unui client
2. administratorul introduce datele în pagina încărcată în browser
3. sistemul accesează baza de date și introduce corect informațiile noului utilizator
4. sistemul afișează pagina de vizualizare a tuturor utilizatorilor
Scenariu de eșec: 3a. datele nu sunt introduse in baza de date
4a. sistemul afișează un mesaj de eroare și reafișează pagina de introducere a datelor

Caz de utilizare: ștergere utilizator
Descriere: administratorul dorește să șteargă un utilizator din baza de date
Actor primar: administrator sistem
Precondiții: administratorul e logat în sistem; utilizatorul e înregistrat în baza de date a sistemului
Postcondiții: datele utilizatorului sunt corect modificate în baza de date
Scenariu de succes: 1. administratorul selectează opțiunea **"delete"** din dreptul unui client
2. sistemul accesează baza de date și șterge informațiile corespunzătoare
3. sistemul afișează pagina de vizualizare a tuturor utilizatorilor
Scenariu de eșec: 2a. datele nu sunt șterse din baza de date
3a. sistemul afișează un mesaj de eroare și reafișează pagina de vizualizare a utilizatorilor

Aceste cazuri de utilizare reprezintă operațiile CRUD pe tabela de clienți din baza de date. Celelalte operații efectuate de administrator respectă același flow – el putând face operații CRUD pe toate tabelele. Restul cazurilor de utlizare sunt similare celor prezentate.

Capitolul 5. Proiectare de Detaliu si Implementare

În acest capitol va fi descris în detaliu modul de implementare a sistemului. Vor fi detaliate modul de proiectare şi implementare a elementelor de bază din cadrul sistemului, cum ar fi: realizarea arhitecturii conceptuale, a diagramelor de deployment şi de secvență, structura bazei de date, identificarea şi descrierea componentelor şi a proceselor implicate în sistem.

5.1. Arhitectura conceptuală

Pentru implementarea acestui sistem s-a ales o arhitectură pe niveluri (layered). Nivelurile comunică unele cu altele, unidirecțional. Layerul i stie doar de i-1 nu și de i+1 (sau oricare altul).

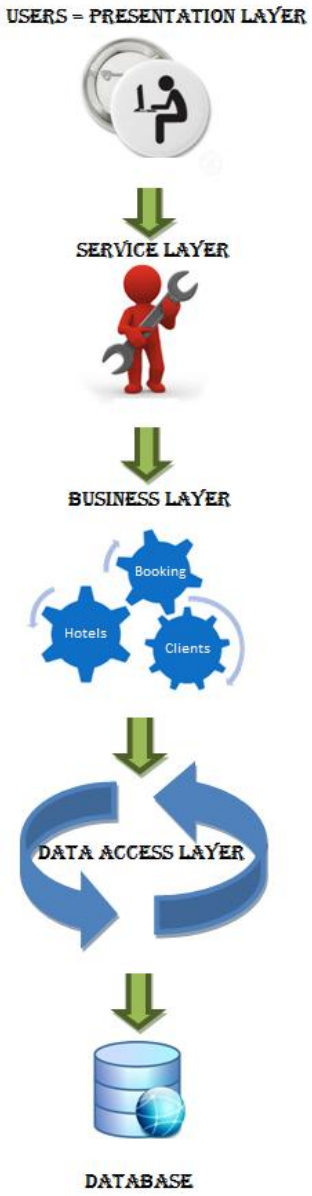

Figura 5.1 - Arhitectura unei aplicații pe niveluri

Arhitectura sistemului prezintă 4 niveluri:

1. **Nivelul de prezentare** – conține funcționalitatea necesară utilizatorilor - prin intermediul acestui layer se face managementul interacțiunilor utilizatorilor cu sistemul; cuprinde în general, în mare parte, apeluri la servicii – pentru a comunica cu logica de business prin intermediul nivelului de servicii
2. **Nivelul de servicii** – este alcătuit din contracte de serviciu (Service Contract) și tipuri de mesaje pentru comunicarea cu logica de business (pentru a separa nivelul de business ca fiind un nivel independent)
3. **Nivelul de business** – acest nivel implementează funcționalitatea de bază a sistemului, el încapsulând logica de business relevantă
4. **Nivelul de acces la date** – acest nivel comunică cu baza de date pentru a prelua și salva anumite date din tabele folosind contextele proprii ale bazelor de date

Folosirea unei astfel de arhitecturi are anumite avantaje, cum ar fi: un layer poate fi înțeles ca un întreg fără a ști prea multe informații despre celelalte layere, impactul modificărilor aduse aplicației este micșorat (fiecare layer are propriile responsabilități iar o modificare a funcționalității implică modificarea componentelor – o componentă poate fi modificată fară a avea impact asupra altora), sistemul este mai ușor de menținut și testat și de asemenea prezintă componente ce pot fi refolosite.

5.2. Implementare

Pentru implementarea fiecărui layer al sistemului a fost creat un proiect separat. Toate proiectele sunt reunite în aceeași soluție. În continuare, se va descrie fiecare proiect ce alcătuiește sistemul.

5.2.1. Layerul de acces la date

Layerul de acces la date este folosit pentru comunicarea cu baza de date pentru a încărca/salva date. Framework-ul .NET oferă ADO .NET Entitiy Data Model (EDM) – un mod ușor de a crea fișiere edmx care descriu schema bazei de date și definesc mapările dintre EDM și baza de date. EDM constituie fundamentul pentru Entity Framework. Este reprezentat de un fișier XML, iar articolole descrise de EDM se numesc entități. Există 3 abordări pentru a folosi Entity Framework:

- model first - permite crearea unui nou model folosind Entity Framework Designer, apoi generarea unei baze de date folosind modelul
- database first – se creează modelul folosind o bază de date existentă
- code first – se creează prima dată clasele, iar apoi se generează baza de date din acele clase; nu se folosește Entity Designer

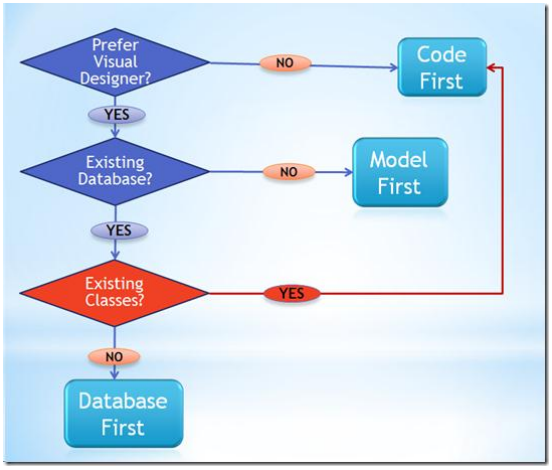

Figura 5.2 – Arbore decizional pentru alegerea abordării Entity Framework

S-a ales să se construiască vizual baza de date (abordarea model first). Asfel, Entity Framework a pus la dispoziție un mediu pentru creare de entități cu atribute specifice și asocieri între acestea.

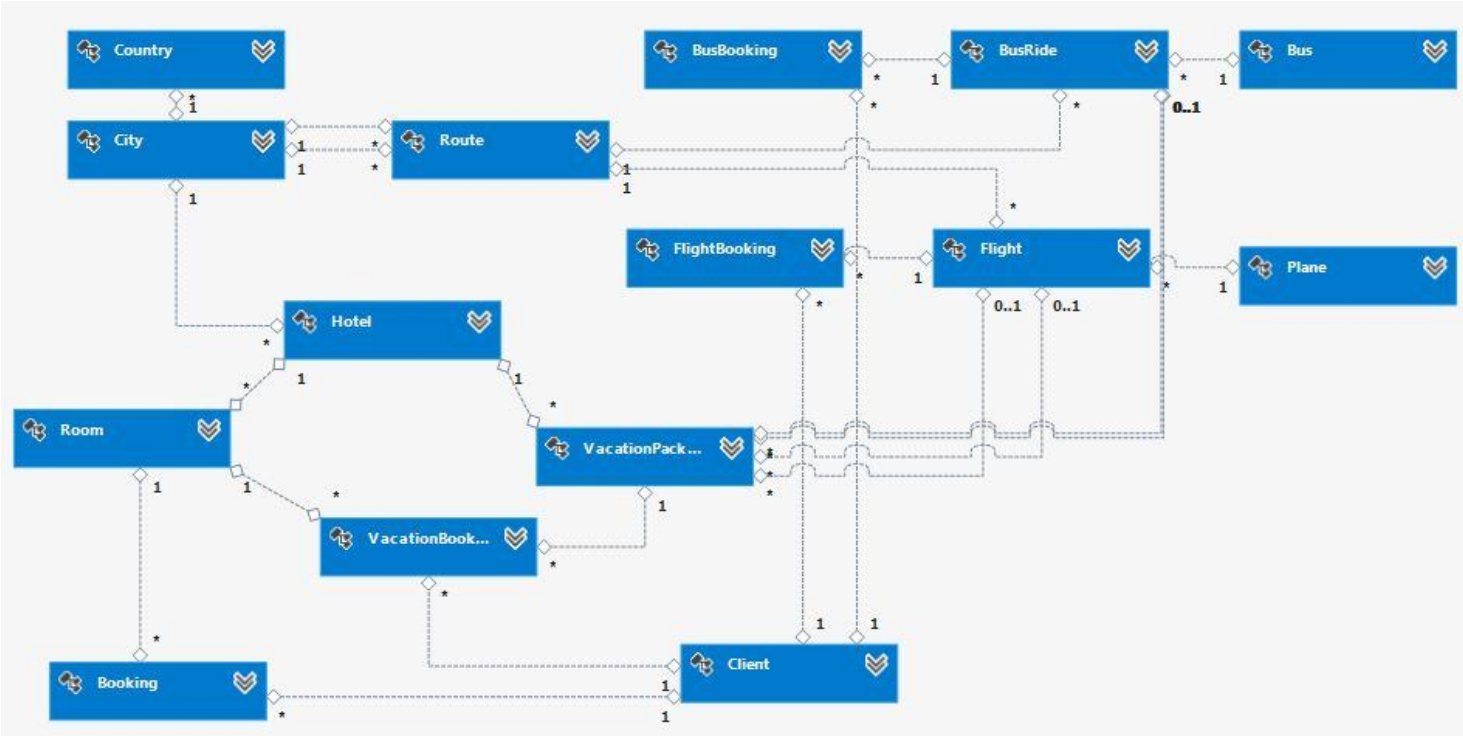

Figura 5.3 – Modelul Entity Framework

După introducerea tuturor entităților (fiecare entitate corespunde unei tabele din baza de date) se generează fișierul cu script SQL care descrie modelul creat. Pentru ca scriptul să se execute cu succes, adică să creeze baza de date în mod corect, folosind Microsoft SQL Server, trebuie setați anumiți parametri de conexiune. Acei parametri vor fi apoi salvați în fișierul XML de descriere a configurației aplicației (App.config), în tag-ul „connectionStrings”. Parametrii includ printre alte informații și numele serverului de baze de date, userul și parola pentru logare, numele bazei de date care urmează a fi creată. După efectuarea acestor setări, se poate executa scriptul și astfel se creează o nouă bază de date în SQL Server. Ca rezultat al tuturor acestor operații va fi creat codul Entity Framework și clasele database context corespunzătoare modelului definit.

Pentru a modifica datele din baza de date, în acest layer este creat un pachet *Repository* ce conține operații de acces la baza de date. Astfel, au fost create clase cu metode de create, read, update, delete pentru toate clasele din model. Metodele instanțiază contextul bazei de date, efectuează operațiile dorite, salvează modificările, iar apoi fac dispose pe instanța creată. Acest lucru asigură faptul că un singur context este folosit de fiecare dată, astfel încât scrierile în baza de date să se facă în mod corect. Folosirea unor contexte diferite ar duce la multiplicarea numărului de scrieri în baza de date sau a nescrierii lor; de exemplu dacă s-ar folosi două instanțe de context, s-ar scrie datele de două ori in baza de date.

5.2.2. Layerul de business

Layerul de business conține logica aplicației și se află între layerul de acces la baza de date și cel de prezentare.

Un lucru ce trebuie luat în considerare la dezvoltarea unei astfel de aplicații este comunicarea dintre nivele. De exemplu, layerul de servicii nu știe de layerul de entități – el nu poate lucra cu obiectele sale, respectiv cu clasele create în urma design-ului modelului. De aceea, fiecare layer trebuie să aibă propriile clase model. Aceste clase sunt asemănătoare cu entitățile, având aceleași proprietăți.

Proiectul ce constituie acest layer conține un pachet cu modele de business. Dat fiind faptul că fiecare layer are propriile modele și poate să lucreze doar cu modelele sale și modelele layerului inferior, a fost necesară implementarea unor clase de conversie – presupunând că avem layerul i și i-1, o clasă de conversie de la nivelul i va avea metode pentru conversia unui obiect model de nivel i la obiect model de nivel i-1 și invers (obiect de la nivelul i-1 la nivelul i). Astfel, în proiect a fost creat un pachet ce conține aceste clase pentru conversia obiectelor. O clasă de conversie conține două metode – una pentru conversia modelului de business la entitate și una pentru conversia entităților la modele de business.

Un alt pachet existent în acest proiect este cel de logică. Pentru a nu avea pentru fiecare entitate o clasă de logică de business, s-au grupat logic operațiile. Astfel, există o clasă pentru operații pe clienți, una pentru locații (operații pe țări, orașe și rute), hoteluri (operații pe hoteluri și camerele lor), pachete de vacanțe, zboruri (operații pe avioane și zboruri), curse de autocar (operații pe autocare și cursele lor), rezervări (rezervări de camere de hotel, pachete de vacanțe, bilete de avion sau bilete de autocar). Acest pachet conține clase care implementează metode ce apelează Repository-ul din layerul de acces la date.

5.2.3. Layerul de servicii

Layerul de servicii are două componente: librăria de servicii WCF și aplicația pentru hosting. Acest nivel expune funcționalități sub forma unui serviciu WCF care poate fi accesat de către orice client, indiferent de platformă.

WCF este un framework folosit pentru dezvoltarea aplicațiilor orientate pe servicii. Cu ajutorul WCF se pot construi sisteme sigure, ce pot fi integrate cu alte platforme. El cuprinde .NET Remoting, tranzacții distribuite, cozi de mesaje și servicii web într-un singur model de programare pentru sisteme distribuite.

Un serviciu WCF este expus ca o colecție de endpoints. Un endpoint reprezintă un punct în care mesajele se trimit sau se recepționează (sau ambele). Comunicarea dintre două endpoints se face folosind SOAP (WCF în mod implicit face ca endpoints să fie disponibile doar clienților SOAP). SOAP este un protocol folosit în scenariile client-server pentru schimbul de informații. Clientul invocă un serviciu care este pe server, iar serviciul web SOAP returnează date clientului.

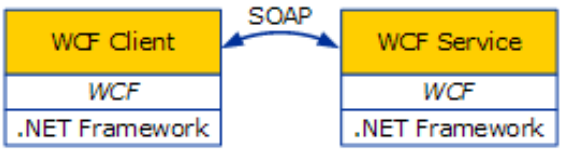

Figura 5.4 – Schimbul de informații între client și serviciu

Prescurtarea "ABC" poate fi folosită pentru memorarea noțiunilor Address / Binding / Contract (Adresa / Conexiune / Contract). Adresa reprezintă locația ce definește unde se vor trimite mesajele. Bindingul e o specificare a mecanismului de comunicare ce descrie cum ar trebui trimise mesajele. Aici sunt specificate ce protocoale de comunicare sunt folosite pentru accesarea serviciului, dacă anumite mecanisme de securitate sunt necesare, și alte informații similare. WCF include bindinguri predefinite pentru cele mai comune protocoale de comunicare, cum ar fi SOAP peste HTTP, SOAP peste TCP, și SOAP peste Message Queues, etc. Contractul este o definiție a unui set de mesaje ce pot fi trimise sau recepționate la acea locație – o înțelegere între două sau mai multe părți asupra a ce se va comunica.

```
<services>
      <service name="WcfServiceLibrary.WCFService">
        <endpoint address="" binding="basicHttpBinding"
contract="WcfServiceLibrary.IWCFService">
          <identity>
            <dns value="localhost" />
          </identity>
        </endpoint>
        <endpoint address="mex" binding="mexHttpBinding"
contract="IMetadataExchange" />
        <host>
          <baseAddresses>
            <add
baseAddress="http://localhost:8733/Design_Time_Addresses/WcfServiceLibrary/WCFServ
ice/" />
```

```
          </baseAddresses>
        </host>
      </service>
</services>
```

Figura 5.5 – Configurarea serviciului WCF

După cum se observă, binding-urile folosite pentru endpoints sunt "basicHttpBinding" și "mexHttpBinding":

- Specificarea elementului "basicHttpBinding" indică faptul că se va folosi protocolul de transport HTTP pentru enpoint-ul respectiv. Acesta are contractul "WcfServiceLibrary.IWCFService", contract ce este de fapt interfața ce conține metode ce vor putea fi apelate de client.
- Serviciul este expus folosind metadate (mex – Metadata Exchange). Binding-ul acesta folosește contractul predefinit IMetadataExchange și este folosit pentru a obține metadate despre metodele serviciului, contracte de date, etc, astfel încât să se poata construi proxy-uri pe partea de client care să apeleze metodele serviciului WCF. Adresa este irelevantă, "mex" fiind considerată doar o bună practică.

Când un client dorește să acceseze serviciul prin intermediul unui endpoint, nu îi este suficientă doar cunoașterea contractului, ci mai este necesară și acceptarea tipului de legătură, specificată de endpoint. Deci atât clientul, cât și serverul trebuie să aibă endpoint-uri compatibile.

Modelele de la acest nivel se află în pachetul DataContracts. Un *Data Contract* descrie tipul de date folosit de serviciu – este o înțelegere formală între serviciu și client care descrie datele ce urmează a fi comunicate. Cu alte cuvinte, pentru a comunica, clientul și serviciul trebuie să cunoască aceleași contracte. Astfel, există clase ce au proprietățile entităților, cu diferența că înaintea fiecărei proprietăți este specificat cuvântul cheie *[DataMember]*, astfel încât serviciul va ști ce date vor fi serializate (transformate în XML) pentru a fi trimise. WCF folosește implicit motorul de serializare numit Data Contract Serializer pentru a serializa și deserializa datele[12]. Și acest nivel conține un pachet de convertoare, pentru a face conversia între modelele de la nivelul de business și cele de la nivelul de servicii.

Un alt tip de contract existent la acest nivel este cel de servicii (*Service Contract*). Pentru a defini serviciile puse la dispoziție clientului, se folosește o interfață: IWCFService. Această interfață conține definirea metodelor din layerul de business (cu diferența că aici tipul de obiecte cu care se lucrează și care se returnează este cel definit în Data Contract). Astfel, un Service Contract este gateway-ul către un serviciu, folosit de aplicațiile externe care doresc să consume funcționalitățile sale. Fiecare metodă din interfață are înainte cuvântul cheie *[OperationContract]*. Un astfel de contract definește metodele serviciului care sunt expuse către clienți.

Implementarea interfeței se face în clasa WCFService. Pentru a controla modul în care obiectele serviciului WCF sunt instanțiate pe server, framework-ul pune la dispoziție 3 moduri de control:

- Per Call – se creează o instanță de serviciu WCF la fiecare apel de metodă făcut de către client
- Per Session – se creează o singură instanță pentru fiecare sesiune a unui client
- Single Instance – se creează o singură instanță pentru toți clienții

S-a ales să se folosească metoda Per Call. Astfel, fiecare client va avea o instanță a serviciului. Modul de funcționare este următorul: clientul apelează proxy-ul, iar acesta apelează serviciul, WCF creează o instanță și apelează metoda dorită iar la returnarea rezultatului apelează IDisposable (pentru a scăpa de resurse ce nu sunt controlate regulat). Această metodă nu menține starea între apeluri diferite, dar consumă mai puțină memorie iar capacitatea de procesare este mai mare decât în cazul celorlalte abordări.

A doua parte componentă a acestui nivel de servicii este un alt proiect, o aplicație consolă pentru hostingul serviciului. Un astfel de procedeu se numește self-hosting (este "găzduit" local, în aceeași soluție). S-a ales să se facă configurația în fișierul App.config, asemenea proiectului WCF Library.

```
<service name="WcfServiceLibrary.WCFService"
behaviorConfiguration="MetadataBehavior">

        <endpoint address="basic" binding="basicHttpBinding"
        contract="WcfServiceLibrary.IWCFService"></endpoint>
        <endpoint address="ws"
        bindingConfiguration="NoSecurityPlusReliableMessaging"
        binding="wsHttpBinding"
        contract="WcfServiceLibrary.IWCFService"></endpoint>
        <endpoint address="mex" binding="mexHttpBinding"
contract="IMetadataExchange" />

         <host>
           <baseAddresses>
             <add baseAddress="http://localhost:8080/travel_agency/" />
           </baseAddresses>
         </host>

</service>
```

Figura 5.6 – Configurația aplicației de hosting a serviciului WCF

Adresa de bază unde este hostat serviciul este http://localhost:8080/travel_agency/. Tipul serviciului este WcfServiceLibrary.WCFService (clasa din libraria WCF unde s-au implementat operațiile), iar contractul este interfața IWCFService (interfața din libraria WCF unde s-au definit operațiile). Se folosesc 3 endpoints: unul cu binding basicHttp, unul wsHttp iar ultimul este cel pentru schimb de metadate.

Comportamentul serviciului (service behavior) trebuie descris pentru a permite schimbul de metadate astfel încât clienții să poată genera clase proxy.

```
<serviceBehaviors>
        <behavior name="MetadataBehavior">
          <serviceMetadata httpGetEnabled="True" httpsGetEnabled="True"/>
        </behavior>
</serviceBehaviors>
```

Figura 5.7 – Comportamentul serviciului WCF

Se observă că s-a setat opțiunea httpGetEnabled. Această proprietate este folosită pentru a obține sau a seta o valoare care indică dacă să se publice sau nu metadate –

obținerea metadatelor se face prin requesturi HTTP GET. Dacă această proprietate ar fi setată la valoarea fals, atunci nu s-ar putea vedea fișierul WSDL de descriere a serviciului prin adăugarea "?wsdl" la sfârșitul URL-ului serviciului WCF. De asemenea, nu s-ar putea adăuga o referință la acest serviciu deoarece WSDL este folosit și pentru descrierea modului de acces la serviciu.

Funcționalitatea propriu-zisă a aplicației este dată de singura clasă din proiect, clasă ce implementează metoda Main. Aceasta instanțiază un Service Host de tipul WCFService (clasa din librăria WCF) și pornește comunicația (host.Open() – metoda folosește configurația aplicației, ce binding e folosit, la ce adresă se află serviciul și ce contract a implementat, iar apoi deschide un canal de comunicație).

5.2.4. Layerul de prezentare

Acest nivel este nivelul cu care interacționează utilizatorul. Acest layer este construit pe baza pattern-ului Model-View-Controller – este un proiect ASP .NET MVC.

MVC este un model arhitectural care separă funcționalitatea specifică domeniului pentru care este dezvoltat sistemul software de interfața grafică a aplicației, permițând dezvoltarea, întreținerea și testarea separată a celor două părți. Acest model împarte un sistem software în trei părți, și anume: model, view și controller.

Modelul gestionează datele sistemului software. View-ul redă modelul într-o formă care permite interacțiunea cu utilizatorul, prin intermediul elementelor de interfață cu utilizatorul. Pentru un singur model pot exista mai multe view-uri pentru a deservi diferite scopuri. Controller-ul recepționează acțiunile utilizatorului și răspunde interogând modelul [1].

Astfel, în pachetul Model al proiectului se află clase mapate pe entități, precum și alte clase care se mapează pe view-uri cum ar fi, model pentru adăugare de date, pentru editare, pentru afișare (management model ce conține o listă de modele astfel încât să se poată face afișarea mai multor rezultate pe pagină – de exemplu afișarea listei de clienți). Aceste modele conțin doar proprietăți, nu și metode pentru procesarea datelor. Fiecare proprietate este precedată de *[Required(ErrorMessage = "Field is required")]*, unde mesajul de eroare este customizat pentru fiecare câmp – astfel încât la introducerea in view a datelor să se afișeze mesaje de eroare dacă acele câmpuri nu sunt completate.

Pachetul de view-uri conține toate fișierele *.cshtml necesare pentru afișarea datelor. Aceste fișiere se mapează pe un model. Un model trebuie să fie prepopulat pentru a putea fi afișat într-un view sau, dacă se dorește popularea unui model cu date provenite dintr-un view, acest lucru se va face în controller.

Pachetul de controllere conține două controllere: AdminController și HomeController. Cel pentru administrator este folosit doar pentru efectuarea operațiilor CRUD de către administratorii de sistem, iar al doilea este folosit pentru operațiile specifice unui utilizator simplu (inclusiv logarea utilizatorilor). Controllerele au metode ce returnează un ActionResult – acesta poate fi un view sau o redirectare către o altă metodă ce returnează un ActionResult. Aceste metode pot să fie precedate de *[HttpGet]* sau *[HttpPost]* – se cere o resursă pentru afișare sau se procesează datele primite de la client.

Metodele pentru procesarea datelor se află într-un alt pachet, numit Logic. Un prim rol al claselor din acest pachet este să se conecteze la serviciul WCF expus. Pentru a putea face acest lucru, trebuie ca proiectul MVC (clientul) să aibă o referință la serviciu.

Odată pornit, serviciul va putea fi descoperit, astfel încât la alegerea opțiunii "Add service reference" a proiectului, se poate introduce adresa URL a serviciului sau se poate găsi automat, fiind în aceeași soluție, cu opțiunea "Discover".

După adăugarea serviciului la referințele proiectului, acesta va putea fi folosit pentru a apela metode. Pentru a consuma serviciul, trebuie folosită referința pentru a deschide canalul de comunicare (crearea proxy-ului):

private static WCFServiceClient channel = new WCFServiceClient("BasicHttpBinding_IWCFService");

S-a ales arbitrar unul din endpoint-urile puse la dispoziție pentru conectarea la serviciu și anume basicHttpBinding. După instanțierea acestui proxy, se pot folosi metodele serviciului.

Asemenea pachetului din layerul de business, pachetul Logic conține o clasă pentru operații pe clienți, una pentru locații (operații pe țări, orașe și rute), hoteluri (operații pe hoteluri și camerele lor), pachete de vacanțe, zboruri (operații pe avioane și zboruri), curse de autocar (operații pe autocare și cursele lor), rezervări (rezervări de camere de hotel, vacanțe, bilete de avion sau bilete de autocar). Aceste clase sunt folosite pentru a popula view-urile (s-a ales să nu existe logică în controller pentru o decuplare mai mare și un grad de refolosire mai mare) și pentru a efectua operațiile de business. Astfel, fiecare clasă conține o instanțiere a proxy-ului prin care se pot apela funcționalitățile serviciului pentru afișare de date, adăugare, editare sau ștergere.

5.3. Proiectarea datelor - diagrama bazei de date

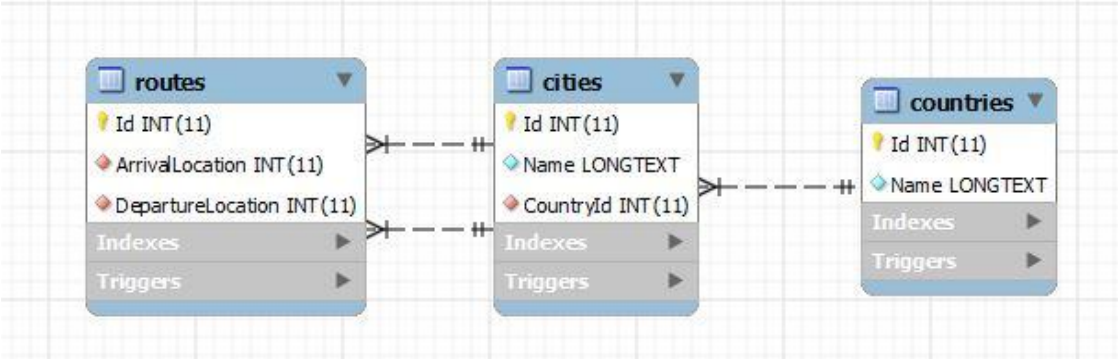

Figura 5.8 – Tabelele "Location" – Țări, Orașe, Rute

Tabela *countries* conține lista de țări existente în baza de date. Conține următoarele câmpuri:

- Id – cheie primară a tabelei
- Name – numele țării

Tabela *cities* conține lista de orașe din țările din baza de date. Conține următoarele câmpuri:

- Id – cheie primară a tabelei
- Name – numele orașului

- CountryId – cheie străină – pot să existe doar orașe din țările introduse la momentul respectiv în baza de date

Tabela *routes* conține rutele pe care se pot deplasa mijloacele de transport oferite de sistem. O rută are un punct de început și un punct de sfârșit (orașe). Tabela conține următoarele câmpuri:

- Id – cheie primară a tabelei
- ArrivalLocation – cheie străină – locația poate fi doar un oraș existent în baza de date
- DepartureLocation - cheie străină – locația poate fi doar un oraș existent în baza de date

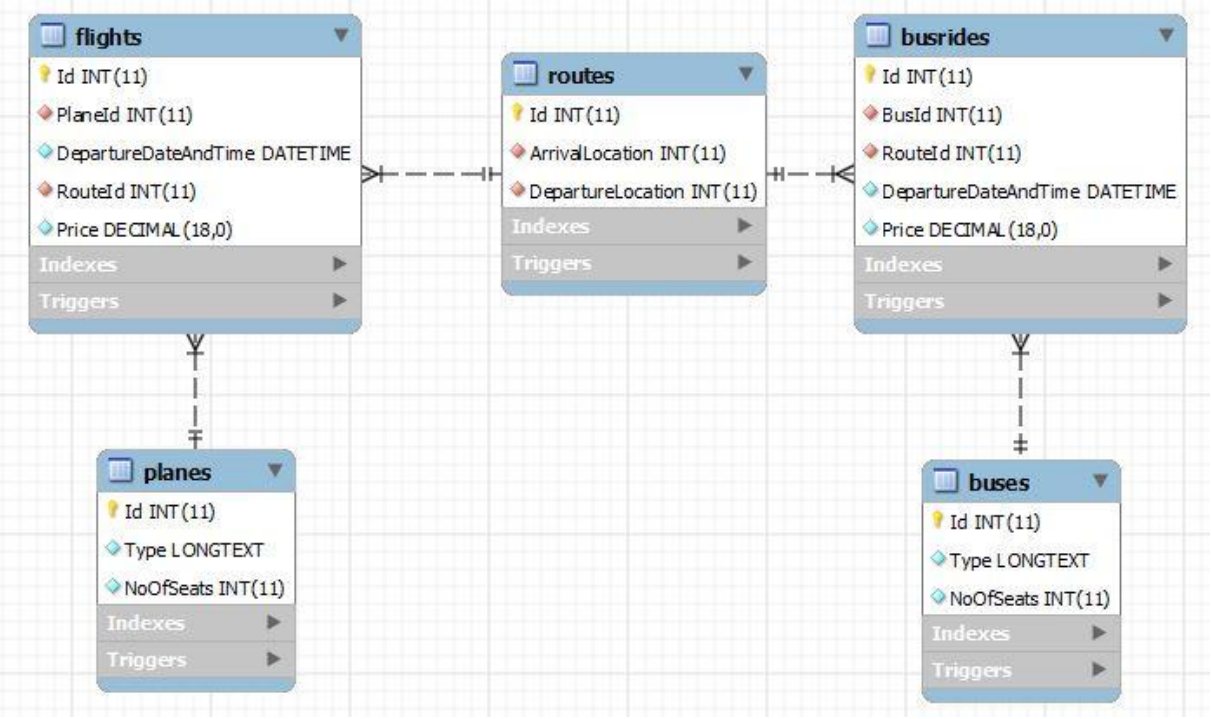

Figura 5.9 – Tabelele "BusRide" și "Flight" – Avioane, Zboruri, Autocare, Curse de autocare

Tabela *planes* conține avioanele disponibile pentru clienți. Prezintă următoarele cîmpuri:

- Id – cheie primară a tabelei
- Type – tipul avionului
- NoOfSeats – numărul de locuri disponibile

Tabela *flights* conține informații despre cursele aeriene disponibile pentru clienți. Prezintă următoarele cîmpuri:

- Id – cheie primară a tabelei
- PlaneId – cheie străină – un zbor poate fi efectuat doar cu un avion existent în baza de date

- DepartureDateAndTime – data și ora de decolare a zborului
- RouteId – cheie străină – un zbor poate circula doar pe o rută existentă în baza de date
- Price – prețul unui bilet

Tabela *buses* conține autocarele disponibile pentru clienți. Prezintă următoarele cîmpuri:

- Id – cheie primară a tabelei
- Type – tipul autocarului
- NoOfSeats – numărul de locuri disponibile

Tabela *busrides* conține informații despre cursele de autocar disponibile pentru clienți. Prezintă următoarele cîmpuri:

- Id – cheie primară a tabelei
- BusId – cheie străină – un zbor poate fi efectuat doar cu un autocar existent în baza de date
- DepartureDateAndTime – data și ora de plecare a autocarului
- RouteId – cheie străină – un autocar poate circula doar pe o rută existentă în baza de date
- Price – prețul unui bilet

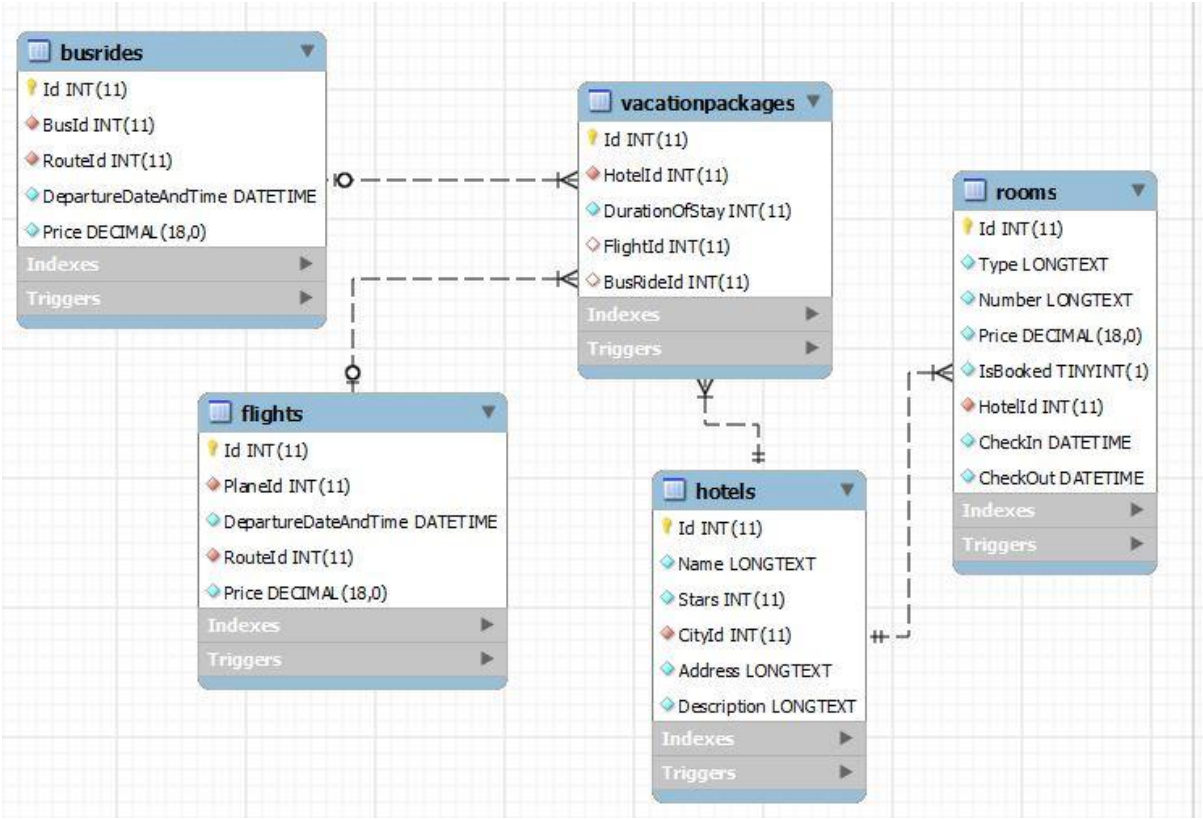

Figura 5.10 – Tabelele "VacationPackage" – Pachete de vacanță, Hoteluri, Camere, Zboruri, Curse de autocar

Tabela *vacationpackages* conține informații despre pachetele de vacanțe disponibile pentru clienți. Un astfel de pachet conține cazare la hotel și un mijloc de transport. Prezintă următoarele cîmpuri:

- Id – cheie primară a tabelei
- HotelId – cheie străină – oferta de cazare poate fi făcută doar la un hotel existent în baza de date
- DurationOfStay – numărul de nopți de cazare oferite
- FlightId – cheie străină - oferta de transport poate fi făcută doar pe zborurile existente în baza de date
- BusRideId – cheie străină - oferta de transport poate fi făcută doar pe cursele de autocar existente în baza de date

Tabela *hotels* conține hotelurile puse la dispoziția clienților. Prezintă următoarele cîmpuri:

- Id – cheie primară a tabelei
- Name – numele hotelului
- Stars – numărul de stele al hotelului
- CityId – cheie străină – hotelul se poate afla doar într-un oraș existent în baza de date
- Address – adresa hotelului
- Description – descrierea hotelului

Tabela *rooms* conține camerele hotelurilor. Prezintă următoarele cîmpuri:

- Id – cheie primară a tabelei
- Type – tipul camerei – single, double, twin, suite
- Number – numărul camerei
- Price – prețul camerei
- IsBooked – câmp boolean ce semnalează dacă la un moment dat camera este rezervată
- HotelId - cheie străină – camera poate să aparțină doar unui hotel existent în baza de date
- CheckIn – în cazul în care camera este rezervată se salvează data de început a rezervării
- CheckOut – în cazul în care camera este rezervată se salvează data de eliberare a camerei

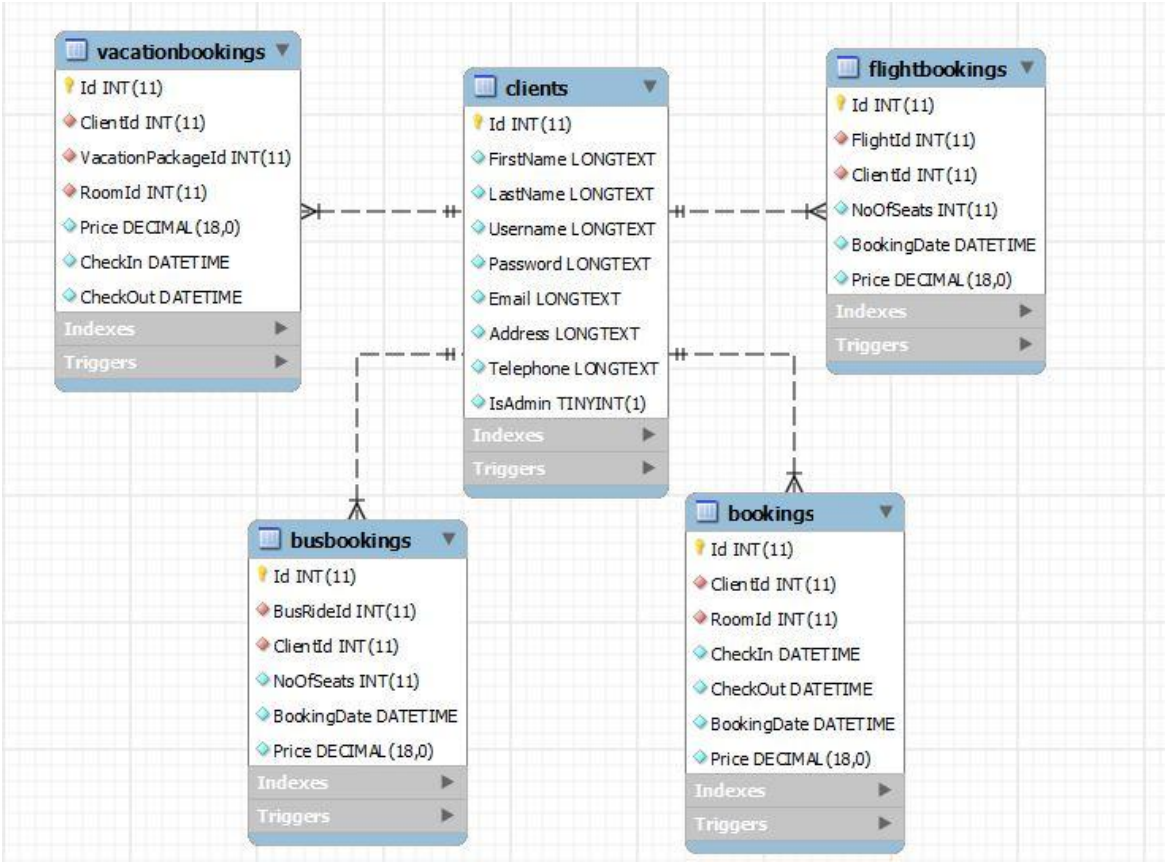

Figura 5.11 – Tabelele "Bookings" – Clienți, Rezervări camere, Rezervări avion, Rezervări autocar, Rezervări vacanțe

Tabela *clients* conține lista utilizatorilor sistemului. Prezintă următoarele cîmpuri:

- Id – cheie primară a tabelei
- FirstName – prenumele clientului
- LastName – numele clientului
- Username – numele de utilizator al clientului
- Password – parola clientului
- Email – adresa de email a clientului
- Address – adresa clientului
- Telephone – numărul de telefon al clientului
- IsAdmin – câmp boolean ce semnalează dacă utilizatorul este un administrator al sistemului sau nu

Tabela *bookings* conține lista rezervărilor din unitățile de cazare ale sistemului. Prezintă următoarele cîmpuri:

- Id – cheie primară a tabelei
- ClientId – cheie străină – doar un client din baza de date poate face o rezervare
- RoomId – cheie străină – rezervarea se poate face doar pentru camerele puse la dispoziție de sistem

- CheckIn – data de început a rezervării
- CheckOut – data de eliberare a camerei
- BookingDate – data la care s-a făcut rezervarea
- Price – prețul total

Tabela *flightbookings* conține lista rezervărilor pe cursele aeriene ale sistemului. Prezintă următoarele cîmpuri:

- Id – cheie primară a tabelei
- ClientId – cheie străină – doar un client din baza de date poate face o rezervare
- FlightId – cheie străină – rezervarea se poate face doar pentru zborurile puse la dispoziție de sistem
- NoOfSeats – numărul de locuri rezervate
- BookingDate – data la care s-a făcut rezervarea
- Price – prețul total

Tabela *busbookings* conține lista rezervărilor pe cursele de autocar ale sistemului. Prezintă următoarele cîmpuri:

- Id – cheie primară a tabelei
- ClientId – cheie străină – doar un client din baza de date poate face o rezervare
- BusRideId – cheie străină – rezervarea se poate face doar pe cursele de autocar puse la dispoziție de sistem
- NoOfSeats – numărul de locuri rezervate
- BookingDate – data la care s-a făcut rezervarea
- Price – prețul total

Tabela *vacationbookings* conține lista rezervărilor de pachete de vacanțe ale sistemului. Un astfel de pachet conține cazare la hotel și un mijloc de transport. Tabela prezintă următoarele cîmpuri:

- Id – cheie primară a tabelei
- ClientId – cheie străină – doar un client din baza de date poate face o rezervare
- VacationPackageId – cheie străină – rezervarea se poate face doar pentru ofertele de vacanțe existente în baza de date
- RoomId – cheie străină – rezervarea se poate face doar pentru camerele puse la dispoziție de sistem
- CheckIn – data de început a rezervării
- CheckOut – data de eliberare a camerei
- Price – prețul total

5.4. Diagrame de pachete

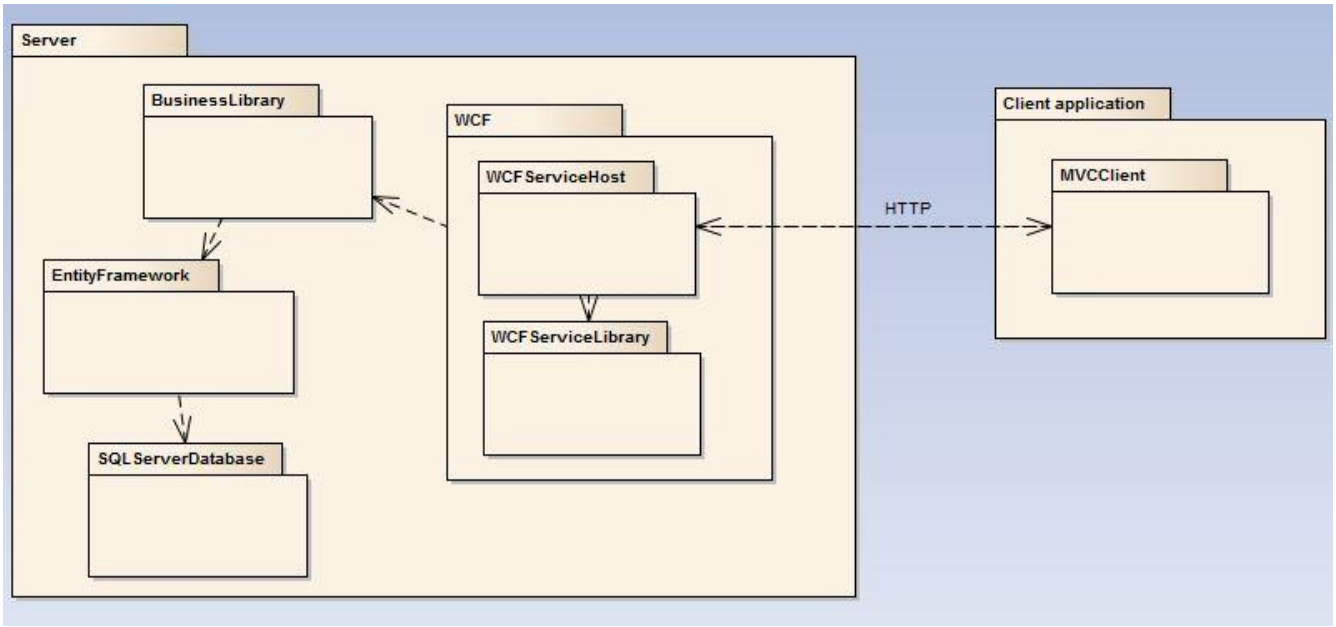

Figura 5.12 – Diagrama de pachete a sistemului

După cum se observă, sistemul conține 5 pachete mari, reprezentate fiecare de un proiect:

1. EntityFramework (proiect TravelAgencyEntities) – acest pachet conține modelul de date în urma căruia au fost generate clasele model. Pentru fiecare entitate existentă în acest model, s-a generat automat o clasă ce are ca atribute proprietățile entității (proprietățile scalare) și proprietățile de navigație (date de relația dintre două entități). Există 15 entități, fiecare devenind ulterior o tabelă în baza de date. De asemenea, aici mai există un pachet numit **Repository** ce conține clase care prin intermediul modelului accesează datele din baza de date.

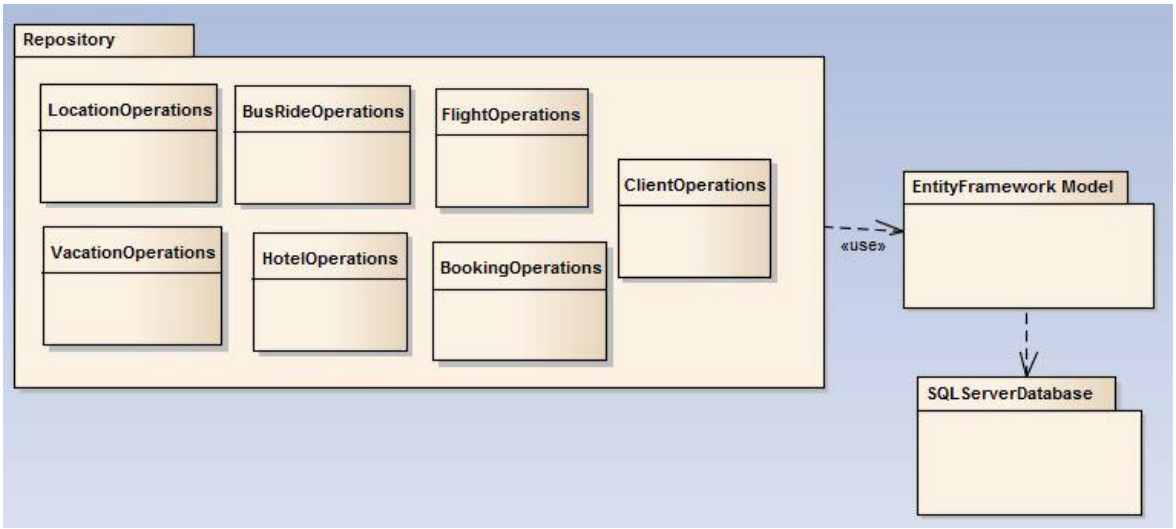

Figura 5.13 – Diagrama de pachete din proiectul TravelAgencyEntities

Pentru accesul la baza de date, pentru a nu avea câte o clasă pentru fiecare entitate din model, s-a făcut o împărțire pe categorii. Astfel, există 7 clase ce conțin doar metode ce folosesc contextul oferit de modelul EF pentru a citi date și pentru a le modifica:

- LocationOperations: cuprinde operații pe entitățile Country, City și Route
- BusRideOperations: cuprinde operații pe entitățile Bus și BusRide
- FlightOperations: cuprinde operații pe entitățile Plane și Flight
- HotelOperations: cuprinde operații pe entitățile Hotel și Room
- VacationOperations: cuprinde operații pe entitatea VacationPackage
- BookingOperations: cuprinde operații pe entitățile Booking, BusRideBooking, FlightBooking și VacationBooking
- ClientOperations: cuprinde operații pe entitatea Client

După cum se observă, aceste clase folosesc ca și model returnat al metodelor, clasele generate de EF.

2. BusinessLibrary (proiect BusinessLibrary) – acest pachet conține modelele de la layerul de business, convertoare pentru aceste modele și clase ce apelează clasele din pachetul Repository din pachetul EF.

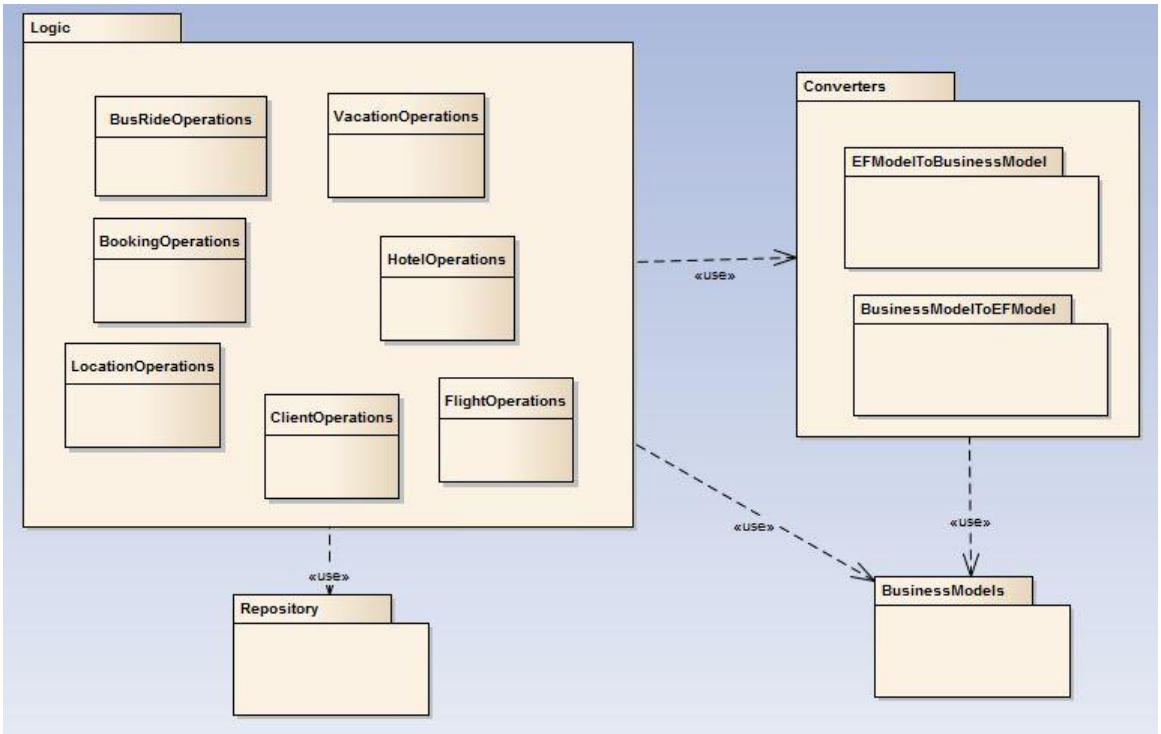

Figura 5.14 – Diagrama de pachete din proiectul BusinessLibrary

Pachetul **BusinessModels** conține același număr de clase ca și cel de modele EF, ele având aceleași proprietăți scalare, schimbându-se doar tipul proprietăților provenite din cele de navigație.

Pachetul **Logic** conține clase ale căror metode apelează clasele din pachetul **Repository** pentru a avea acces la date. Asemenea acelui pachet, clasele sunt împărțite având în vedere funționalitatea lor. Spre deosebire de clasele din Repository, metodele acestora returnează modele de business, nu modele EF. Din această cauză a fost nevoie de convertoare – pentru a putea transforma un model EF într-unul de business și invers. Astfel, există un pachet **Converters** ce conține clase care facilitează această transformare a modelelor. Pentru fiecare model există doua metode: una pentru conversie EFToBusiness și una BusinessToEF.

3. WCFServiceLibrary (proiect WCFServiceLibrary) - acest pachet conține modelele de la layerul de servicii, convertoare pentru aceste modele și clase ce apelează clasele din pachetul Logic din pachetul BusinessLibrary.

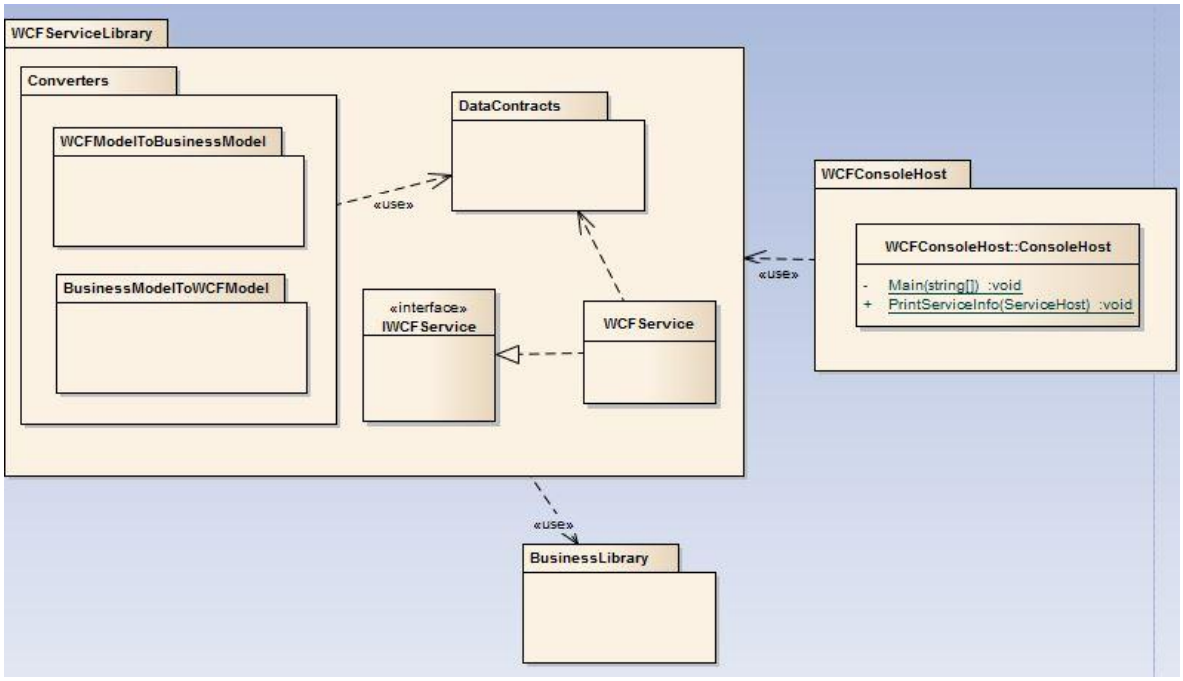

Figura 5.15 – Diagrama de pachete din proiectele WCFServiceLibrary și WCFConsoleHost

Acest pachet lucrează cu modele proprii numite *DataContracts*. Astfel, pachetul **DataContracts** conține modelele de bază, cu aceleași proprietăți, schimbându-se doar tipul proprietăților provenite din cele de navigație.

Pachetul conține clasa ce este folosită pentru expunerea serviciilor către clienți: *WCFService*. Aceasta implementează metodele descrise în interfața *IWCFService*, interfață folosită drept contract. Prin accesarea acestui contract, orice client va putea beneficia de serviciul oferit de sistem. Interfața conține toate metodele existente în clasele din pachetul **Logic** din **BusinessLibrary** – o singură clasă de serviciu care cuprinde toată

funcționalitatea oferită în layerele inferioare. Clasa *WCFService* implementează aceste metode, apelând metodele din layerul de business și returnând modele WCF. Datorită faptului că la acest nivel se lucrează cu alt tip de modele, este nevoie și aici de un pachet **Converters**. Acesta conține clase cu câte două metode pentru fiecare model: una pentru transformarea modelului de business în model WCF (BusinessToWCF) și cealaltă pentru transformarea din model WCF în model de business (WCFToBusiness).

Configurarea ABC-urilor binding-urilor se face în fișierul App.config, existent în acest proiect – un fișier xml de configurare ce cuprinde toate informațiile necesare pentru ca un client să știe cum să acceseze în mod corect datele.

4. WCFConsoleHost (proiect WCFConsoleHost) – acest pachet conține o singură clasă ce asigură deschiderea unui canal de comunicație astfel încât clienții să poată accesa serviciul. Aceasta conține două metode: metoda Main pentru pornirea host-ului și o metodă de printare a informațiilor despre endpoint-uri. În metoda Main se instanțiază un *ServiceHost* de tipul *WCFService* (clasa din pachetul **WCFServiceLibrary** în care erau implementate metodele de acces la date). Apoi, se pornește host-ul prin metoda *host.Open()* și pentru a verifica dacă au fost făcute bine configurațiile, se printează informații în consolă despre endpoint-uri – la ce adresă sunt, ce binding au, ce nume au, etc.

La fel ca în pachetul **WCFServiceLibrary**, configurarea ABC-urilor binding-urilor se face în fișierul App.config, existent în proiect. Tot în acest fișier se configurează și host-ul, pentru permite clienților să se conecteze.

5. MVCClient (proiect TravelAgencyMVC) – conține pachetele pentru modele, view-uri și controllere, convertoarele pentru modele, pachet de clase ce apelează serviciul pentru a accesa datele și pachete pentru conținut (imagini și fișiere .css) și scripturi necesare pentru prelucrarea HTML.

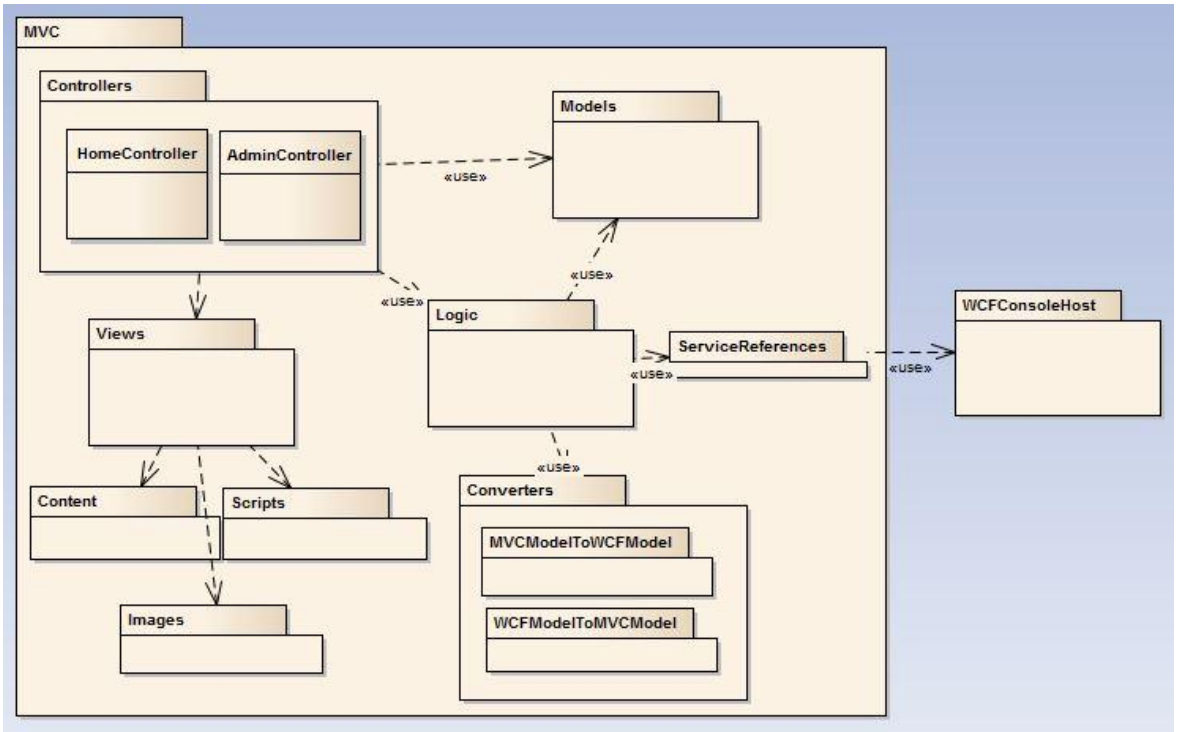

Figura 5.16 - Diagrama de pachete din proiectul TravelAgencyMVC

La fel ca și celelalte proiecte, cel MVC are și el propriile modele. Aceste modele sunt în număr mai mare decât în layerele anterioare, deoarece aici este nevoie de un model pentru fiecare View – fiecare view necesită anumite informații și de aceea trebuie făcute modele speciale.

View-urile se accesează prin acțiuni din controller. Având două tipuri de utilizatori, administrator și utilizator obișnuit, există două clase controller: *AdminController* și *HomeController*. Controller-ul pentru administrator conține toate acțiunile pe care le poate face acesta, adică operații CRUD pe fiecare tabelă din baza de date, iar controllerul home conține acțiunile făcute de un simplu utilizator al sistemului – logare, înregistrare, căutări de hoteluri, vacanțe, efectuare de rezervări, etc. Deoarece fiecare acțiune din controller are un view specific, și pachetul de Views este împărțit în două: view-uri pentru administrator și pentru utilizatori. Pentru așezarea în pagină se folosesc fișierele *.css din pachetul **Content.** Imaginile ce apar pe pagini se găsesc în pachetul **Images**. În pachetul script se află scripturi folosite pentru a îmbogăți experiența utilizatorului prin oferirea componentelor interactive (spre exemplu pentru funcția de afișare a calendarului când utilizatorul dorește să selecteze o dată).

Pentru a accesa serviciul pus la dispoziție, acest proiect trebuie să se conecteze la respectivul serviciu. Pentru a face acest lucru, se adaugă o referință la el. Dacă serviciul este pornit, el va putea fi descoperit, astfel încât la alegerea opțiunii "Add service reference" a proiectului, se poate introduce adresa URL a serviciului sau se poate găsi automat, fiind în aceeași soluție, cu opțiunea "Discover". Acestă referință se află în pachetul **ServiceReferences**.

Pachetul care are clase ce accesează serviciul este pachetul **Logic**. Asemănător pachetului din layerul de business, acesta conține cele 7 clase pentru operațiile pe tipurile de date – LocationOperations, BusRideOperations, FlightOperations, HotelOperations, VacationOperations, BookingOperations și ClientOperations. Fiecare clasă conține o instanțiere a proxy-ului prin care se pot apela funcționalitățile serviciului (pentru a consuma serviciul, trebuie folosită referința pentru a deschide canalul de comunicare – crearea proxy-ului). Datorită faptului ca aceste metode returnează modele de layer WCF și aici există un pachet de convertoare, pentru a transforma aceste modele în modele corespunzătoare de la nivelul de prezentare. Pachetul **Converters** conține clase cu câte două metode pentru fiecare model: una pentru transformarea modelului MVC în model WCF (MVCToWCF) și cealaltă pentru transformarea din model WCF în model MVC (WCFToMVC).

5.5. Elemente de integrare a nivelurilor cu alte sisteme

5.5.1. Facebook

Sistemul permite o logare mai ușoară, folosind rețeaua socială Facebook. Pentru a beneficia de serviciile oferite de Facebook, un developer trebuie să se logheze, să își creeze o aplicație pentru care să își înregistreze un domeniu și astfel va primi credențiale pentru aplicația sa – un id de client și un secret. Acestea vor fi setate ca și chei în fișierul web.config al proiectului, la secțiunea appSettings.

```
public ActionResult FacebookLogin()
{
	var redirectUri = new UriBuilder(Request.Url);
	redirectUri.Path = Url.Action("FacebookCallback", "Home");
	var client = new FacebookClient();
	var uri = client.GetLoginUrl(new { client_id = AppId, redirect_uri =
	redirectUri.Uri.AbsoluteUri });
	return Redirect(uri.ToString());
}
```

Figura 5.17 – Funcția de logare cu Facebook

Această acțiune face o redirectare către pagina de logare a Facebook după autentificare id-ului client (AppId). Dacă utilizatorul s-a logat cu succes, el va fi redirectat la o altă acțiune și anume FacebookCallback (dată ca redirectUri).

```
public ActionResult FacebookCallback(string code)
{
	var fbclient = new FacebookClient();
	var oauthResult = fbclient.ParseOAuthCallbackUrl(Request.Url);

	// Build the Return URI form the Request Url
	var redirectUri = new UriBuilder(Request.Url);
	redirectUri.Path = Url.Action("FacebookCallback", "Home");

	// Exchange the code for an access token
	dynamic result = fbclient.Get("/oauth/access_token",
			new {
			client_id = AppId,
			redirect_uri = redirectUri.Uri.AbsoluteUri,
			client_secret = SecId, code = oauthResult.Code, });

	// Read the auth values
	string accessToken = result.access_token;
	this.Session["access_token"] = accessToken;
	DateTime expires =
		DateTime.UtcNow.AddSeconds(Convert.ToDouble(result.expires));

	// Get the user's profile information
	dynamic me = fbclient.Get("/me",
			new { fields = "first_name,last_name,email",
			access_token = accessToken });
}
```

Figura 5.18 – Funcția de preluare a datelor unui utilizator

Având credențialele de logare, se eliberează un token de acces. Pe baza acestui token va fi identificat utilizatorul. Astfel, i se pot prelua datele – numele, prenumele și adresa de email – necesare pentru o logare în sistem.

5.5.2. Google maps

Pentru a oferi funcționalitatea unei hărți ce să arate utilizatorului unde se află unitățile de cazare, s-a folosit Google Maps. Acest API oferă posibilitatea de a construi hărți ce pot fi personalizate cu conținut propriu și imagini. Folosind limbajul JavaScript

se poate afișa și stiliza o hartă după preferințele programatorului – acesta poate alege să folosească și una sau mai multe din librăriile Google puse la dispoziție (Geocoding, Directions, Street View, etc.).

Pentru a folosi API-ul, un developer trebuie să se logheze pe site-ul cu API-uri Google pentru a primi o cheie unică, folosită pentru ca programul să funcționeze corect. De asemenea, de aici se vor activa și serviciile ce se doresc a fi integrate în program. S-a ales să se activeze 3 servicii pentru a le folosi în sistemul de față: Google Maps JavaScript API v3, Google Maps Geolocation API și Directions API.

```
if (navigator.geolocation)
{
       navigator.geolocation.getCurrentPosition(function (position)
       {
              pos = new google.maps.LatLng(position.coords.latitude,
       position.coords.longitude);

           });
         }
}
```

Figura 5.19 – Aflarea poziției unui utilizator

Folosind Geolocation, se poate afla poziția unei persoane (a calculatorului care îl folosește). Astfel, cu acest serviciu s-au aflat coordonatele unui utilizator al sistemului – latitudinea și longitudinea.

```
    function calcRoute()
    {
        if (navigator.geolocation)
        {
            navigator.geolocation.getCurrentPosition(function (position) {

                var pos = new google.maps.LatLng(position.coords.latitude,
                                                 position.coords.longitude);

                var start = pos;
                var end = '@Model.Address';
                var request = {
                    origin: start,
                    destination: end,
                    travelMode: google.maps.TravelMode.DRIVING
                };
                directionsService.route(request, function (response, status) {
                    if (status == google.maps.DirectionsStatus.OK) {
                        directionsDisplay.setDirections(response);
                    }
                });
            });
        }
```

Figura 5.20 – Funcția de calculare a rutei dintre două puncte

Folosind serviciul de direcționare Google Directions, se poate afla ruta dintre două coordonate de pe hartă. Astfel, având ca și punct de start poziția utilizatorului sistemului și punct final adresa unui hotel. Direcțiile date sunt pentru drum rutier.

După setarea rutei se face un request cu datele ce se doresc a fi afișate, iar răspunsul se va afișa pe ecran.

```
function initialize()
{
        directionsDisplay = new google.maps.DirectionsRenderer();
        var pos;
        if (navigator.geolocation)
        {
            navigator.geolocation.getCurrentPosition(function (position) {

                pos = new google.maps.LatLng(position.coords.latitude,
                                                    position.coords.longitude);

            });
        }
        var mapOptions = {
            zoom: 7,
            center: pos
        };
        var map = new google.maps.Map(document.getElementById('map-canvas'),
            mapOptions);
        directionsDisplay.setMap(map);
        directionsDisplay.setPanel(document.getElementById('directions-panel'));
}
```

Figura 5.21 – Funcția de afișare a hărții

După aflarea poziției utilizatorului se va inițializa o hartă, ce va fi centrată în poziția găsită. Pe display se va seta acestă hartă și un panou ce conține direcțiile pentru a ajunge din poziția curentă la adresa hotelului căutat.

5.6. Concluzii

Sistemul de față prezintă o arhitectură pe niveluri. Există patru niveluri: nivelul de prezentare (implementat folosind ASP .NET MVC – conține funcționalitatea necesară utilizatorilor), nivelul de servicii (implementat folosind WCF – expune serviciile la care se pot conecta clienții – funcționalitățile sistemului), nivelul de business (implementează funcționalitatea de bază a sistemului) și nivelul de acces la date (implementat folosind EntityFramework – comunică cu baza de date pentru a prelua și salva datele din tabele).

Folosirea unei astfel de arhitecturi prezintă anumite avantaje, cum ar fi o bună decuplare a modulelor deoarece fiecare nivel comunică doar cu nivelul anterior, astfel poate fi mai ușor de întreținut și modificat, iar componentele pot fi refolosite cu ușurință.

Capitolul 6. Testare şi Validare

Pentru aplicația de față a fost realizată o testare pentru fiecare dintre funcționalitățile sale. Pentru a verifica dacă serviciul funcționează corect s-a folosit tool-ul pus la dispoziție de Microsoft Visual Studio numit WCF Test Client. Pentru testarea metodelor de pe partea clientului s-au folosit Unit Testing.

Înainte de cele două metode de testare s-a folosit o simplă testare a serviciului, pentru a verifica faptul că funcționează și permite clienților să îl acceseze. Pentru acest test, s-a folosit o afișare în consolă la pornirea host-ului. Când host-ul este pornit, acesta pornește de fapt serviciul, făcând endpoint-urile sale vizibile și disponibile.

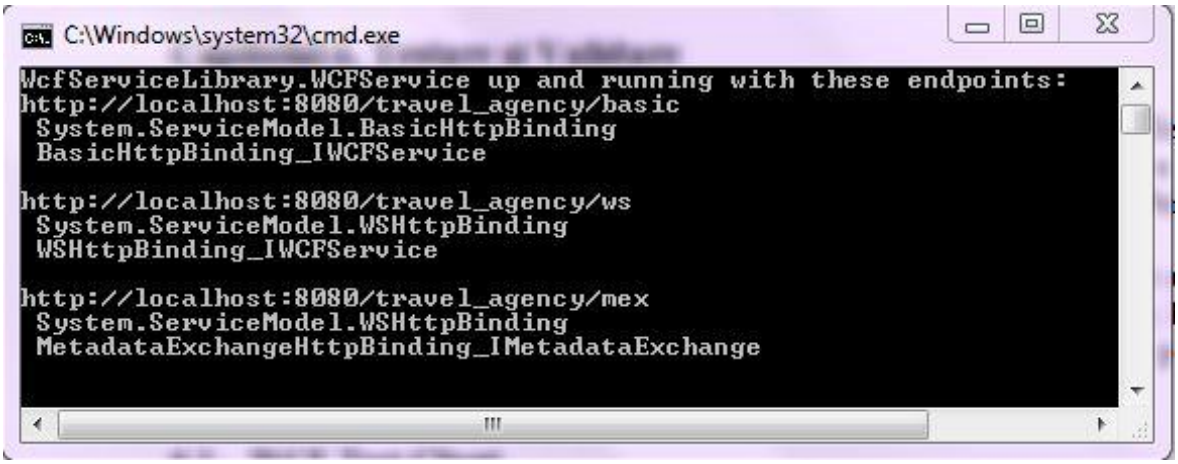

Figura 6.1 – Console host – endpoint-uri disponibile

În figura 6.1 se pot observa cele 3 endpoint-uri disponibile, fiecare cu o adresă, un binding și un nume. Un client se poate lega la oricare din aceste endpoint-uri.

6.1. WCF Test Client

WCF Test Client (WcfTestClient.exe) este un tool ce permite utilizatorilor să introducă parametri de test, să transmită acei parametri serviciului și să vizualizeze răspunsul pe care acesta îl trimite înapoi.

Pentru a folosi acest tool, trebuie introdus serviciul ce se dorește a fi testat – adresa endpoint-ului. După introducerea acestei adrese, în partea stângă a interfeței vor fi afișate toate metodele puse la dispoziție de serviciu, fiecare putând fi testată. Dacă se selectează o metodă, în panoul din dreapta vor fi afișate două cadrane: unul *Request* și unul *Response*.

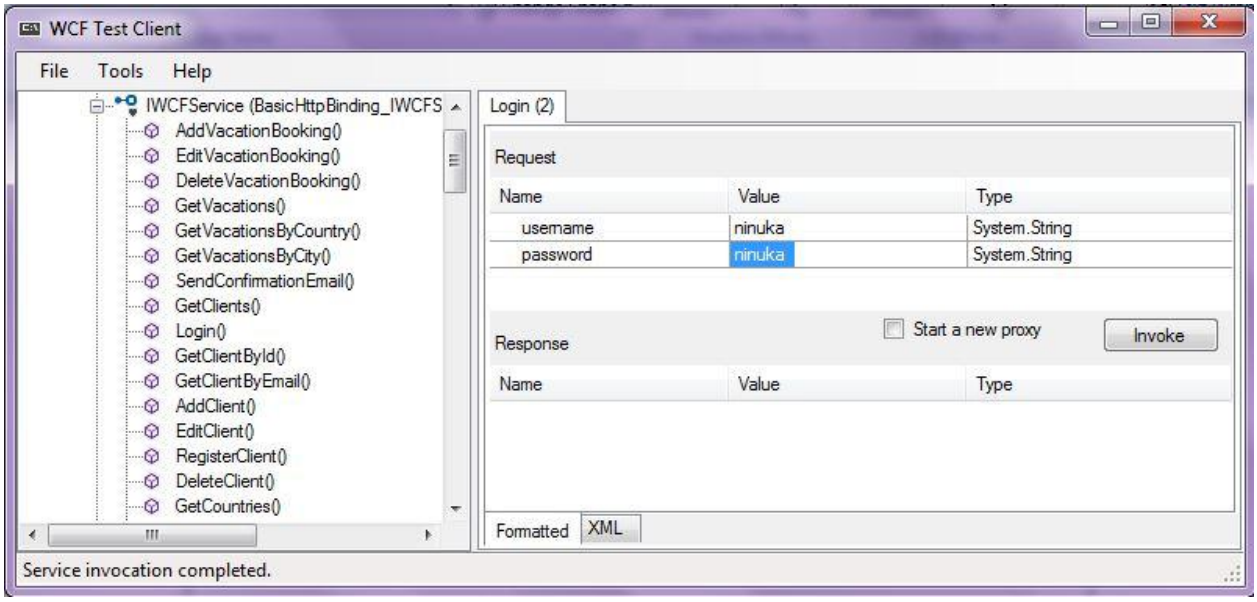

Figura 6.2 – WCF Test Client – request metoda *Login*

În primul cadran se vor introduce datele (parametrii metodei), dacă există, iar apoi se va selecta opțiunea *Invoke*. În urma acestei comenzi se va afișa rezultatul metodei.

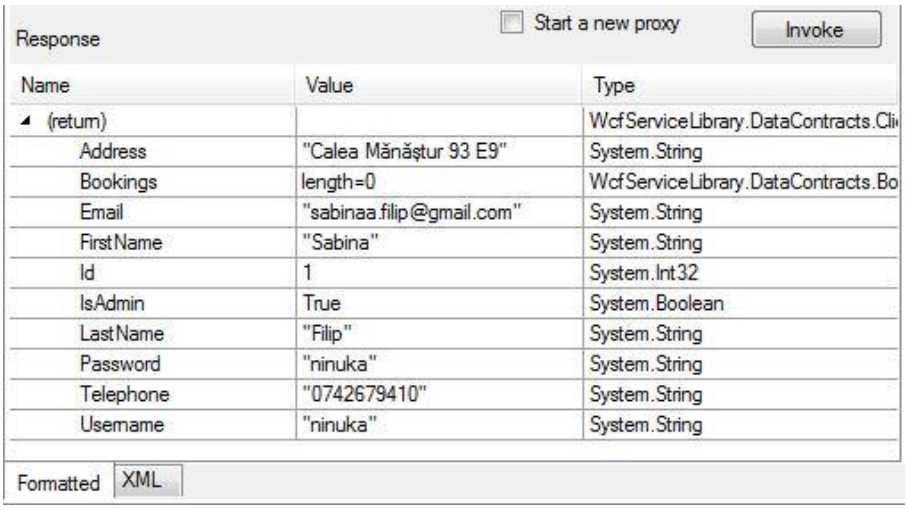

Figura 6.3 – WCF Test Client – response metoda *Login*

În mod similar au fost testate toate funcțiile oferite de serviciu:

- Funcțiile de afișare au fost testate prin compararea rezultatelor afișate în tool și a celor din baza de date.
- Funcțiile de adăugare, editare și stergere s-au verificat prin invocarea unei metode de afișare a înregistrărilor după fiecare operație efectuată.
- Funcțiile de căutare au fost testate prin parcurgerea înregistrărilor din baza de date pentru a verifica dacă toate datele s-au extras corect și complet

- Funcțiile de preluare a unor date specifice (extragerea unui utilizator după adresa sa de mail, extragerea unui hotel după id, etc.) au fost testate prin compararea datelor rezultate din metode cu cele din baza de date

6.2. Unit Testing

Microsft Visual Studio pune la dispoziție un pachet ce conține clase ce oferă suport pentru testarea unitară și anume Microsoft.VisualStudio.TestTools.UnitTesting. Testarea unitară reprezintă o modalitate de testare care verifică dacă unităţile individuale de cod au comportamentul aşteptat. Astfel, s-au putut testa metodele din controllerul proiectului MVC.

Pentru a efectua acest tip de testare s-a creat un nou proiect în soluție – *TravelAgencyMVCTests* ce conține clasele de test. Având în vedere că aceste teste verifică nu doar datele la nivelul interfață, ci și metodele serviciului (unele acțiuni din controller apelează serviciul pentru a prelua sau a modifica datele din baza de date), a fost nevoie să se adauge o referință la serviciu și acestui proiect. Astfel, în mod asemănător adăugării unei referințe descris în capitolul 5.2.4, se adaugă o referință la serviciul WCF expus.

În clasele din acest proiect s-a testat funcționalitatea unor acțiuni din cele două controllere precum:

- Returnarea view-ului potrivit unei acțiuni
- Popularea corectă a unui view
- Preluarea corectă a datelor din view
- Afișarea corectă a datelor din baza de date
- Accesul la serviciul WCF

```
[TestMethod]
public void TestReturnedView()
{
        var controller = new AdminController();
        var result = controller.Index() as ViewResult;
        Assert.AreEqual("Index", result.ViewName);
}
```

Figura 6.4 – Testare returnare view

O metodă de test instanțiază obiectul de testat – controllerul – și apelează metoda care se dorește a fi testată. În cazul de față, metoda Index() ce are ca rol afișarea paginii de start a programului și anume pagina denumită *Index.cshtml*. Metoda verifică dacă numele celor doua view-uri coincid.

```
[TestMethod]
public void TestClientView()
{
    var controller = new AdminController();
    var result = controller.ManageClients() as ViewResult;
    var clientManagementModel =
    (ClientManagementModel)result.ViewData.Model;
    Assert.AreEqual("Sabina", clientManagementModel.Clients[0].FirstName);
}
```

Figura 6.5 – Testare afișare date preluate din baza de date

Pentru a vedea dacă un view afișează corect datele, se apelează o funcție din controller responsabilă cu acest lucru. Funcția *ManageClients()* returnează un view ce are ca model *ClientManagementModel* (model ce conține o listă de clienți). Astfel, pentru acest view se poate extrage informație pentru a vedea cum a fost populat modelul.

Funcția *ManageClients()* apelează o metodă a serviciului WCF pentru a prelua lista cu toți clienții sistemului, iar apoi populează modelul cu rezultatele primite. Acest lucru se întâmplă în controller, iar prin preluarea modelului se poate vedea de fapt ce date s-au extras din baza de date și se vor afișa pe ecran. În cazul de față s-a ales să se verifice prima înregistrare preluată și anume să se verifice numele primului client, știind ce înregistrări există în baza de date.

```
[TestMethod]
public void TestClientAdd()
{
    var controller = new AdminController();
    var clientModel = new ClientModel
    {
        FirstName = "John",
        LastName = "Doe",
        Username = "john",
        Password = "john",
        Address = "address",
        Telephone = "telephone",
        Email = "john@yahoo.com",
        IsAdmin = false
    };

    controller.AddClient(clientModel);
    var client = ClientOperations.GetClientByEmail("john@yahoo.com");
    Assert.AreEqual("John", client.FirstName);
}
```

Figura 6.6 – Testare populare view și introducere în baza de date

Pentru a verifica dacă o acțiune preia corect datele din view și le transmite mai departe până la baza de date, se apelează o funcție din controller responsabilă cu acest lucru. Funcția *AddClient(ClientModel model)* prelucrează datele din model, date ce provin dintr-un view unde utilizatorul a introdus informațiile noului client. Pentru a simula introducerea datelor, s-a creat un model *ClientModel* și s-a populat cu date. Acest model a fost folosit apoi pentru a vedea dacă acțiunea din controller are efectul dorit – aceasta trebuie să preia datele și să apeleze serviciul WCF pentru a le introduce în baza de date. Verificarea corectitudinii se face apoi prin testarea existenței noului client în lista de înregistrări – clasa *ClientOperations* este o clasă din pachetul *Logic* al proiectului TravelAgencyMVC, de acolo se apelează funcția de extragere a unui client pe baza email-ului său.

6.3. Concluzii

Toate metodele au fost verificate, demonstrându-se că sistemul funcționează corect. S-au testat funcționalitățile folosind mai multe metode: o afișare în consolă pentru verificarea disponibilității serviciului (printarea unor informații despre enpoint-urile puse la dispoziția clientului), WCF Test Client – un tool vizual oferit de Microsoft pentru verificarea serviciilor WCF și unit tests pentru verificarea acțiunilor provenite de la client.

Capitolul 7. Manual de Instalare și Utilizare

7.1. Specificații de instalare

Proiectul a fost dezvoltat în mediul de programare Microsoft Visual Studio 2012. Pentru a funcționa, el trebuie rulat într-un mediu cu aceeași versiune sau o versiune mai nouă.

Sistemul este compus dintr-o singură soluție, ce se va încărca în Visual Studio prin deschiderea fișierului *TravelAgency.sln*. La deschiderea proiectului, se vor încărca toate cele 5 proiecte ce compun soluția, împreună cu proiectul ce conține testele.

Rularea propriu-zisă se face în două etape:

- Se pornește prima dată host-ul aplicației, ce va face serviciul disponibil. Acest lucru se face selectând din *Solution Explorer* al Visual Studio proiectul *WCFConsoleHost*. Apăsând click dreapta, se poate selecta opțiunea Debug – Start new instance. În urma acestei acțiuni, se va porni serviciul – utilizatorul poate vedea acest lucru datorită consolei de afișare.
- Se pornește clientul. Din Solution Explorer se selectează proiectul TravelAgencyMVC și din nou se va face click dreapta și se va alege opțiunea Debug – Start new instance.

7.2. Manual de utilizare

La pornirea clientului, se va deschide în browser-ul de Internet o pagină nouă, pagina de start a aplicației. Pe această pagină, un utilizator poate efectua mai multe operații: poate alege să vizualizeze hoteluri, zboruri, curse de autocar sau pachete de vacanțe.

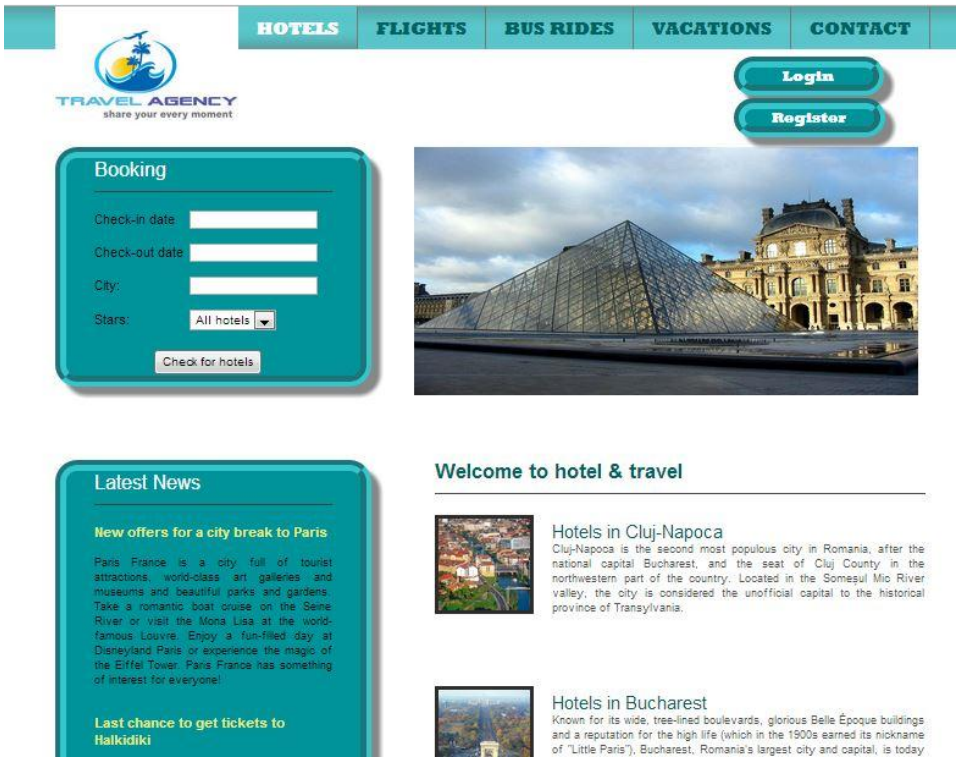

Figura 7.1 – Pagina de start a aplicației

Se observă că această primă pagină este destinată căutării de hoteluri. Utilizatorul poate alege din variantele propuse pe pagină (hoteluri din diferite orașe) sau poate căuta hotelul dorit folosind căsuța de căutare din partea stângă a ecranului. Dacă niciun câmp nu este completat, iar utilizatorul apasă butonul "Check for hotels", sistemul va returna o listă cu toate hotelurile din baza de date. În cazul în care dorește o căutare mai specifică, el poate introduce data la care dorește să se cazeze și decazeze, poate introduce orașul unde dorește să se afle hotelul și/sau numărul de stele al hotelului.

Pentru introducerea datelor se dă click pe caseta de text din dreptul datei de check-in și va apărea un calendar de pe care se poate alege data dorită doar cu un singur click. Pentru a căuta hoteluri într-un anumit oraș, se introduce numele său în caseta de text. După tastarea a cel puțin o literă, sistemul va face sugestii oferind o listă cu toate orasele ce incep cu litera sau literele respective. Pentru alegerea numărului de stele al hotelului se alege din lista de pe ecran o opțiune (1 – 5).

În funcție de parametrii de căutare, sistemul va avea un anumit răspuns.

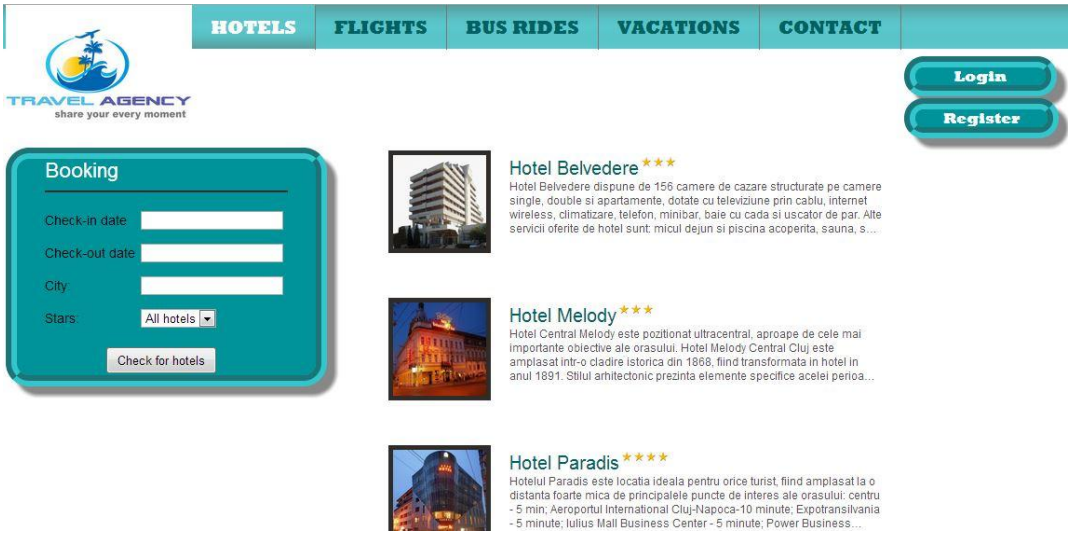

Figura 7.2 – Pagina de vizualizare hoteluri

Din lista de hoteluri afișate, userul poate alege să vadă în detaliu informațiile unuia, prin accesarea link-ului din titlul său. Astfel, el poate vedea o scurtă descriere a hotelului, o listă cu prețurile camerelor și o galerie cu poze. De asemenea, aici este oferită posibilitatea de a vedea unitatea de cazare pe hartă prin alegerea opțiunii "View on map".

Dacă se selectează opțiunea "Book a room", sistemul va verifica dacă utilizatorul este logat. În cazul în care acesta nu este logat, se va afișa un mesaj de eroare. În caz contrar, se va afișa pagina de finalizare a rezervării.

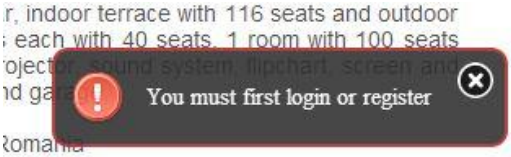

Figura 7.3 - Mesaj de eroare pentru efectuare rezervare
cu utilizator nelogat

Figura 7.4 – Pagina de detalii a unui hotel

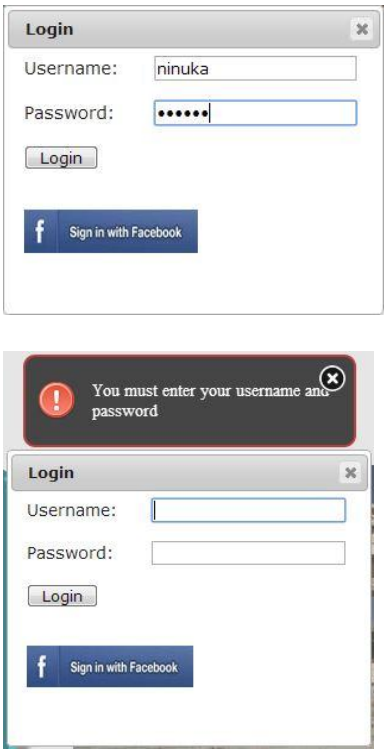

Figura 7.5 – Formular de logare

Pentru a putea efectua rezervarea, utilizatorul va trebui să aleagă una din opțiunile existente pe ecran: să se logheze (utilizatorul are deja cont) sau să se înregistreze în sistem. De asemenea, un utilizator se poate loga cu contul de Facebook – acestă opțiune îl va duce la pagina Facebook de introducere a datelor de logare, iar apoi va reveni la pagina principală a sistemului de față.

După efectuarea logării/înregistrării, sistemul va afișa un mesaj de "Bun venit pentru utilizator" și îi va pune la dispoziție alte două opțiuni: să își vadă istoria rezervărilor făcute sau să se delogheze.

Figura 7.6 – Mesaj de întâmpinare utilizator

După finalizarea logării/înregistrării, utilizatorul poate să efectueze o rezervare. În primă fază, formularul de rezervare conține doar data de check in, data de check out și o opțiune pentru alegerea numărului de camere dorite. Un utilizator poate rezerva până la 6 camere o dată. După ce se alege un număr de camere dorit, pe formular vor mai apărea opțiuni pentru selectarea tipului camerelor dorit. După ce utilizatorul completează toate datele și apasă butonul "Book now", sistemul verifică în baza de date pentru a vedea dacă la hotelul respectiv mai există tipul de camere cerut. Dacă da, confirmă rezervarea printr-

un mesaj pe ecran și trimite mail utilizatorului cu datele rezervării. Dacă nu sunt disponibile, se va afișa un mesaj de eroare și utilizatorul va trebui să aleagă din nou, un alt tip de cameră sau poate chiar un alt hotel.

Figura 7.7 – Formular de rezervare cameră

O altă operație ce poate fi efectuată de utilizator este cea de căutare și vizualizare de zboruri și curse de autocar, prin selectarea opțiunii "Flights"/ "Bus Rides". O astfel de căutare se poate efectua pentru bilete doar dus, sau dus-întors. Pentru a căuta zboruri, utilizatorul trebuie să completeze câmpurile de introducere a datei de plecare și a orașelor de plecare, respectiv sosire.

Figura 7.8 – Pagina căutare zboruri

Pentru rezervarea biletelor, utilizatorul va alege opțiunea "Book flight" (respectiv "Book bus ride" pentru cursele de autocar). Dacă este logat, va fi dus la pagina de detalii a zborului, unde va trebui să aleagă și numărul de bilete dorit. Pe această pagină este afișat și prețul total, acesta schimbându-se de fiecare dată când utilizatorul alege un număr de bilete.

Figura 7.9 – Pagina de detalii a unui zbor

La alegerea opțiunii “Book”, sistemul va verifica în baza de date dacă mai sunt disponibile locurile cerute și va afișa un mesaj corespunzător pe ecran.

O altă opțiune ce poate fi aleasă din meniul principal este “Vacations”. Pentru căutarea unui pachet de vacanțe, utilizatorul trebuie să introducă destinația dorită, precum și orașul de plecare. Sistemul va afișa apoi o listă de hoteluri, existente în pachete de vacanță, ce corespund filtrelor de căutare – hotelurile sunt în orașul cerut și există mijloace de transport între orașele introduse. Accesând pagina de detalii pentru fiecare hotel (în mod similar celui descris anterior), utilizatorul va putea vedea detaliile hotelului, precum și cele pentru cele două curse (cea de plecare și cea de sosire). Dacă dorește să rezerve locuri, va proceda conform pașilor descriși pentru rezervarea de camere de hoteluri.

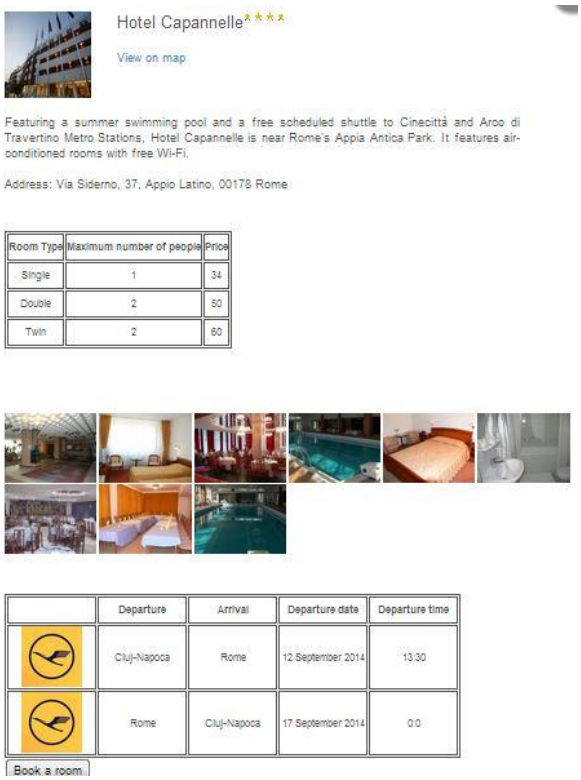

Figura 7.10 – Pagina de detalii a unei vacanțe

Pentru a vedea toate rezervările făcute, un utilizator poate alege opțiunea “Booking history”. Aceasta va afișa o listă cu toate rezervările făcute de respectivul utilizator (rezervări de camere, mijloace de transport și vacanțe), oferindu-i și opțiunea de a putea să anuleze unele dintre ele. Dacă o rezervare nu a trecut de termenul limită de o săptămână, ea mai poate fi anulată.

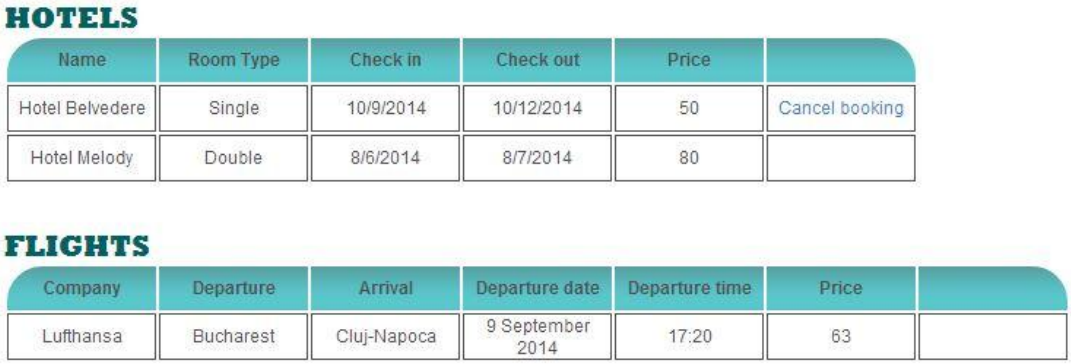

Figura 7.11 – Pagina de vizualizare rezervări utilizator

Pentru administratorii de sistem, este pusă la dispoziție o interfață simplă, care conține toate operațiile pe care aceștia le pot face. Operațiile sunt grupate logic, asemenea modului prezentat în capitolul 5.2. Astfel, un administrator poate vizualiza toate înregistrările din orice tabelă a bazei de date și poate face orice modificări pe ea: poate adăuga înregistrări, le poate edita sau le poate șterge.

Figura 7.12 – Interfața administratorului

Capitolul 8. Concluzii

În acest capitol vor fi prezentate concluziile dezvoltării acestei lucrări. În prima parte va fi prezentat un rezumat al realizărilor care au dus la construcţia proiectului, urmând mai apoi o analiză a rezultatelor obţinute, iar în final va fi prezentată o serie de dezvoltări şi îmbunătăţiri ulterioare.

8.1. Realizări

Sistemul implementat prezintă capabilităţile de furnizare a unor servicii turistice pentru utilizatori. Printr-o interfata prietenoasă, se doreşte a fi un instrument ce să îi ajute să își organizeze vacanțele dorite.

S-a reuşit implementarea cu succes a cerinţelor enunţate în capitolul 1.3. Sistemul este capabil să ofere servicii clienților – pe terminale desktop sau mobile. Consumarea serviciilor s-a demonstrat folosind un client desktop ASP .NET MVC.

Sistemul oferă utilizatorilor conectați la Internet posibilitatea de a vizualiza și rezerva camere de hoteluri, bilete de avion sau autocar și oferte de vacanțe ce conțin cazare și bilete de transport. Pentru a vizualiza toate aceste informații utilizatorul trebuie doar să acceseze site-ul și să caute servicii conform preferințelor sale. Pentru a face rezervări, acesta trebuie să fie înregistrat în sistem.

Interfaţa site-ului este prietenoasă şi permite utilizatorului o navigare facilă în cadrul site-ului prin acţiuni simple, care necesită doar câteva click-uri. Partea grafică este îndeajuns de atractivă încât să atragă utilizatorul, folosindu-se poze sugestive pentru exemplificarea destinaţiilor oferite. Ca şi orice sistem de acest gen realizat, clientul este un factor important, facilitaţile oferite acestuia de vizualizare a anumitor oferte sau rezervări efectuate, dar şi permiterea acestuia de a efectua rezervări, a le anula, a căuta oferte după anumite destinaţii au scopul de a-i oferi acestuia confortul aşteptat.

Toate datele aplicaţiei sunt salvate într-o bază de date. Administratorul are posibilitatea de a vizualiza, adăuga, modifica și șterge date din orice tabelă. Pentru întreținerea bazei de date a fost realizat un job zilnic de backup.

Sistemul a fost realizat folosind mai multe componente. Astfel, s-a obținut un grad de decuplare mare. Relaţiile între componente sunt astfel realizate încât modificarea unora dintre acestea să nu afecteze pe cât posibil funcţionalitatea celorlalte, iar schimbările în cadrul uneia sa poată fi realizate rapid şi fără erori propagate în celelalte. Gradul mare de decuplare oferă de asemenea și o reuzabilitate a componentelor.

8.2. Rezultate obținute

În prezent, funcțiile aplicației sunt destul de reduse, prezentând doar o mică parte din funcționalitățile care ar putea fi implementate pentru domeniul ales. Cu toate acestea, ea poate fi folosită cu succes de persoanele interesate.

Prin intermediul aplicației un utilizator poate vizualiza informații despre hotelurile existente în baza de date – descriere, tarife, galerie de poze, localizare pe hartă și direcționare până la destinație. De asemenea, poate vizualiza informații despre zboruri și curse de autocar – companii ce oferă aceste servicii, destinații, date de plecare, prețuri. Nu în ultimul rând, el poate beneficia de oferte de vacanțe – conțin informații despre

hoteluri și despre mijloacele de transport pentru a ajuge la destinație și pentru întoarcere acasă.

Pentru a putea face o rezervare pentru oricare dintre aceste servicii, un utilizator trebuie să fie înregistrat în sistem. Astfel, înainte de a-și face o rezervare, utilizatorul trebuie să se autentifice (în cazul în care are deja cont) sau să se înregistreze cu datele personale în caz contrar. O altă opțiune oferită de sistem este logarea cu contul de Facebook.

La efectuarea fiecărei operații sistemul va răspunde cu mesaje informative: mesaje de succes când operațiunea s-a desfășurat în mod așteptat sau mesaje de eroare atunci când operațiunea nu s-a putut efectua, când nu se introduc datele așteptate, când utilizatorul dorește să facă o acțiune ilegală (de exemplu modificarea URL-ului astfel încât să se logheze cu un cont de administrator sau să efectueze rezervări fără să fie logat).

8.3. Dezvoltări ulterioare

Un prim pas în a dezvolta mai departe acest sistem, ar fi extinderea bazei de date. Cu o bază de date mai mare, cu mai multe oferte, experiența utilizatorilor ar putea fi cât mai reală.

Având în vedere faptul că sistemul oferă doar funcționalități pentru cazare, zboruri, curse de autocar și pachete de vacanțe, s-ar mai putea adăuga servicii pentru oferte last-minute, circuite, croaziere, închirieri de mașini.

De asemenea, s-ar putea adăuga o funcționalitate “Things to do” – prin selectarea unei destinații să se poată rezerva activități, cum ar fi tururi de obiective turistice, excursii prin oraș, etc.

Pe lângă adăugarea de noi funcționalități, o dezvoltare semnificativă a sistemului ar fi integrarea unui sistem pentru plată online. Este un mod foarte ușor și convenabil care în ziua de azi se folosește din ce în ce mai mult.

O altă modalitate de dezvoltare a programului ar fi migrarea acestuia spre platformele Mobile, pentru sistemele de operare iOS sau Android. Momentan, serviciile acestuia sunt consumate doar de un client ASP .NET MVC. Prin modificarea elementelor de interfață și logica clientului, aplicația ar putea să deservească și utilizatori de telefonie mobilă.

O altă îmbunătățire ar fi realizarea de suport pentru diferite limbi străine, pentru a putea beneficia mai mulți utilizatori de acest sistem. Pentru aceasta ar trebui implementat un mecanism prin care să se facă localizarea resurselor în funcție de limba selectată de utilizator.

Bibliografie

[1] A. M. Antochi, R. Sarghie, L. Pricop, "Framework-ul ASP .NET MVC", http://www.slideshare.net/radusarghie/aspnet-mvc-2233212
[2] D. Buhalis, "ETourism: Information Technology for Strategic Tourism Management", Pearson (Financial Times/Prentice Hall), London, 2003
[3] D. Buhalis, R. Law"Progress in information technology and tourism management: 20 years on and 10 years after the Internet", The state of eTourism research, Tourism Management 29 (2008) 609–623, 2008
[4] Enterprise Application Architecture: Designing Applications and Sevices in .NET, http://www.codeproject.com/Articles/111270/Enterprise-Application-Architecture-Designing-Appl
[5] G. Şoavă, A. Bădică, "Electronic Tourism", Annals of University of Craiova - Economic Sciences Series, vol. 2, nr. 36, pg 657-662, 2008
[6] I. Condratov, " E-Tourism: Concept and evolution", ECOFORUM Volume 2, Issue 1(2), 2013
[7] I. Gavrilă, Laboratoare Sisteme Distribuite, Capitolul 9 – ADO .NET, http://users.cs.tuiasi.ro/~igavrila/sdm2011/l09sd.pdf
[8] R. Dollinger, L. Andron**,** „Baze de date si Gestiunea Tranzactiilor", Editura Albastra, 2004
[9] S.H. Jun, D. Buhalis, "E-Tourism Contemporary Tourism Reviews", Goodfellow Publishers Limited, Woodeaton, Oxford
[10] S. Sandersaon**,** "Pro ASP. NET MVC Framework", Apress, 2009
[11] United Nations World Tourism Organization, Annual Report 2013, http://dtxtq4w60xqpw.cloudfront.net/sites/all/files/pdf/unwto_annual_report_2013_web.pdf
[12] Using Data Contracts, http://msdn.microsoft.com/en-us/library/ms733127.aspx
[13] The ADO .NET EntityFramework Overview, http://msdn.microsoft.com/en-us/library/aa697427(v=vs.80).aspx
[14] Windows Communication Foundation, http://ro.wikipedia.org/wiki/Windows_Communication_Foundation
[15] Windows Communication Foundation Architecture, http://msdn.microsoft.com/en-us/library/ms733128(v=vs.110).aspx

Anexa 1. Acronime

CRS – Computer Reservation System

GDS – Global Distribution System

TIC – Tehnologii de Informație și Comunicare

EF – Entity Framework

DBMS – Database Management System

RBDMS – Relational DataBase Management System

ORM – Object Relational Mapping

EDM – Entity Data Model

CRUD – Create, Read, Update, Delete

WCF – Windows Communication Foundation

CLR – Common Language Runtime

ABC – Address / Binding / Contract

TCP – Transmission Control Protocol

SOAP – Simple Object Access Protocol

WSDL – Web Services Description Language

URL – Uniform Resource Locator

XML – Extensible Markup Language

IIS – Internet Information Services

MVC – Model-View-Controller

UI – User Interface

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

API – Application Programming Interface

Anexa 2. Lista de figuri și tabele

Capitolul 1. Introducere	-
Capitolul 2. Obiectivele proiectului	-
Capitolul 3. Studiu Bibliografic	Tabelul 3.1 – Comparație între sisteme similare
Capitolul 4 – Analiză și fundamentare teoretică	Figura 4.1 – Model simplificat al arhitecturii ADO.NET [7] Figura 4.2 – Arhitectura WCF [15] Tabelul 4.3 – Cerințe funcționale Tabelul 4.4 – Cerințe non-funcționale Figura 4.5 – Diagrama cazurilor de utilizare pentru un utilizator simplu al sistemului Figura 4.6 – Diagrama cazurilor de utilizare pentru un administrator al sistemului
Capitolul 5 – Proiectare de detaliu și implementare	Figura 5.1 – Arhitectura unei aplicații pe niveluri [4] Figura 5.2 – Arbore decizional pentru alegerea abordării Entity Framework Figura 5.3 – Modelul Entity Framework Figura 5.4 – Schimbul de informații între client și serviciu Figura 5.5 – Configurarea serviciului WCF Figura 5.6 – Configurația aplicației de hosting a serviciului WCF Figura 5.7 – Comportamentul serviciului WCF Figura 5.8 – Tabelele “Location” – Țări, Orașe, Rute Figura 5.9 – Tabelele “BusRide” și “Flight” – Avioane, Zboruri, Autocare, Curse de autocare Figura 5.10 – Tabelele “VacationPackage” – Pachete de vacanță, Hoteluri, Camere, Zboruri, Curse de autocar Figura 5.11 – Tabelele “Bookings” – Clienți, Rezervări camere, Rezervări avion, Rezervări autocar, Rezervări vacanțe Figura 5.12 – Diagrama de pachete a sistemului Figura 5.13 – Diagrama de pachete din proiectul TravelAgencyEntities Figura 5.14 – Diagrama de pachete din proiectul BusinessLibrary Figura 5.15 – Diagrama de pachete din proiectele WCFServiceLibrary și WCFConsoleHost Figura 5.16 – Diagrama de pachete din proiectul TravelAgencyMVC Figura 5.17 – Funcția de logare cu Facebook

	Figura 5.18 – Funcția de preluare a datelor unui utilizator Facebook Figura 5.19 – Aflarea poziției unui utilizator Figura 5.20 – Funcția de calculare a rutei dintre două puncte Figura 5.21 – Funcția de afișare a hărții
Capitolul 6 – Testare și validare	Figura 6.1 – Console host – endpoint-uri disponibile Figura 6.2 – WCF Test Client – request metoda *Login* Figura 6.3 – WCF Test Client – response metoda *Login* Figura 6.4 – Testare returnare view Figura 6.5 – Testare afișare date preluate din baza de date Figura 6.6 – Testare populare view și introducere în baza de date
Capitolul 7 – Manual de instalare și utilizare	Figura 7.1 – Pagina de start a aplicației Figura 7.2 – Pagina de vizualizare hoteluri Figura 7.3 – Mesaj de eroare pentru efectuare rezervare cu utilizator nelogat Figura 7.4 – Pagina de detalii a unui hotelFigura Figura 7.5 – Formular de logare Figura 7.6 – Mesaj de întâmpinare utilizator Figura 7.7 – Formular de rezervare cameră Figura 7.8 – Pagina căutare zboruri Figura 7.9 – Pagina de detalii a unui zbor Figura 7.10 – Pagina de detalii a unei vacanțe Figura 7.11 – Pagina de vizualizare rezervări utilizator Figura 7.12 – Interfața administratorului
Capitolul 8 – Concluzii	