



FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
SPECIALIZAREA AUTOMATICĂ ȘI INFORMATICĂ APLICATĂ



EvenTourTM - Platforma web dedicată gestiunii și promovării evenimentelor din Timișoara

Proiect de diplomă

Absolvent:

Sorin Popescu

Conducător științific:

Ș.l.dr.ing. Andreea ROBU

Timișoara,

2020

CUPRINS

LISTA FIGURILOR.....	4
CAPITOLUL 1 – INTRODUCERE.....	6
1.1 Context general.....	6
1.2 Tema proiectului.....	7
1.3 Structura pe capitole.....	8
CAPITOLUL 2 - FUNDAMENTAREA TEORETICĂ.....	10
2.1 ASP .NET Framework.....	10
2.2 MVC Pattern.....	11
2.3 Entity Framework.....	15
2.4 Microsoft SQL Server.....	17
2.5 Limbajul C#.....	18
2.6 Razor (CSHTML).....	20
2.7 Bootstrap.....	21
2.8 Microsoft Visual Studio.....	22
2.9 Pusher.....	22
CAPITOLUL 3 - DESCRIEREA GENERALĂ A PROIECTULUI.....	23
3.1 Prezentarea sumară a aplicației. Schema bloc.....	23
3.2 Funcționalitățile aplicației.....	24
3.3 Planificarea lucrărilor.....	26
3.3.1 Ciclul de viață V.....	26
3.3.2 Diagrama Gantt.....	28
CAPITOLUL 4 - PROIECTAREA ÎN DETALIU.....	30
4.1 Considerații privind implementarea.....	30
4.2 Arhitectura programului.....	31

4.3 Descrierea modulelor.....	35
4.4 Structura bazei de date.....	45
 CAPITOLUL 5 - UTILIZAREA APLICAȚIEI.....	 49
 CAPITOLUL 6 – CONCLUZII.....	 61
6.1 Privire de ansamblu asupra aplicației.....	61
6.2 Direcții de dezvoltare.....	61
 BIBLIOGRAFIE.....	 62

LISTA FIGURILOR

Figura 2.1 Pattern-ului MVC.....	12
Figura 2.2 Modul de funcționare al pattern-ului MVC.....	14
Figura 2.3 - Modul de functionare al pachetului Entity Framework.....	16
Figura 2.4 - Modul de funcționare al Razor (CSHTML).....	20
Figura 3.1 – Schema bloc a aplicației.....	23
Figura 3.2 – Modelul in V.....	26
Figura 3.3 – Diagrama Gantt.....	29
Figura 4.1 – Schema cu tehnologiile utilizate.....	31
Figura 4.2 – Arhitectura modulului Administrator.....	32
Figura 4.3 – Arhitectura modulului Utilizator autentificat.....	33
Figura 4.4 – Arhitectura modulului Utilizator neautentificat.....	34
Figura 4.5 – Structura bazei de date.....	46
Figura 4.6 – Structura tabelii Users.....	47
Figura 4.7 – Structura tabelii Eveniments.....	47
Figura 4.8 – Structura tabelii Pozes.....	48
Figura 4.9 – Structura tabelii Comments.....	48
Figura 5.1 – Pagina principală a aplicației.....	49
Figura 5.2 – Pagina de Login a aplicației.....	49
Figura 5.3 – Pagina de Creare cont a aplicației.....	50
Figura 5.4 – Pagina principală a aplicației pentru cei autentificați.....	51
Figura 5.5 – Lista categoriilor disponibile.....	51
Figura 5.6 – Lista evenimentelor dintr-o anumită categorie.....	52
Figura 5.7 – Pagina de detalii a unui eveniment.....	52

Figura 5.8 – Galeria foto a unui eveniment.....	53
Figura 5.9 – Lista evenimentelor care s-au desfășurat deja.....	53
Figura 5.10 – Interfața de mesagerie a aplicației/forum.....	54
Figura 5.11 – Pagina principală pentru cei neautentificați.....	54
Figura 5.12 – Lista evenimentelor pentru cei neautentificați.....	55
Figura 5.13 – Pagina de Detalii eveniment pentru cei neautentificați.....	55
Figura 5.14 – Pagina principală pentru administrator.....	56
Figura 5.15 – Lista cu evenimente aferentă administratorului.....	56
Figura 5.16 – Pagina de creare de eveniment nou.....	57
Figura 5.17 – Pagina de editare a unui eveniment.....	57
Figura 5.18 – Pagina de ștergere a unui eveniment.....	58
Figura 5.19 – Pagina Istoric evenimente pentru administrator.....	58
Figura 5.20 – Pagina de galerii foto.....	59
Figura 5.21 – Pagina de creare a unei noi galerii foto.....	59
Figura 5.22 – Pagina de editare a galeriei foto.....	60
Figura 5.23 – Pagina de ștergere a unei galerii foto.....	60
Figura 5.24 – Pagina cu utilizatorii care au cont în aplicație.....	60

CAPITOLUL 1

INTRODUCERE

1.1 Context general

Tehnologia actuală ne face viața de zi cu zi mult mai ușoară prin intermediul unor multitudini de aplicații, unele dintre ele fiind aplicațiile Web. Acestea vin în ajutorul oricărui dezvoltator din diferite domenii de activitate, prin intermediul internetului care joacă, la momentul actual, unul dintre cele mai importante roluri de dezvoltare de afaceri, divertisment, etc., mai mult decât simpla oportunitate de a căuta informații pentru uz personal. Astăzi, acest serviciu are o importanță majoră în industria comunicațiilor, media și a știrilor. Acest lucru se datorează, în principal, aplicațiilor Web deoarece reprezintă tot ceea ce un utilizator folosește când deschide un browser și se conectează, iar aici vorbim, în cele mai multe cazuri, de aplicații precum Facebook, Gmail, jocurile online și majoritatea site-urilor mai complexe. ^[1]

Fiind aproape universal folosite de cei care pun la dispoziție servicii online, era normal ca aplicațiile Web să aibă un număr important de beneficii, care să le facă atât de dezirabile. Fiind dezvoltate pentru a putea fi accesate de pe orice browser, aplicațiile Web trebuie testate doar pentru diferite astfel de navigatoare, iar faptul că la ele se poate lucra de pe orice sistem de operare, nu mai este necesară și testarea pe mai multe din acestea, rezultând, astfel, costuri mai scăzute în procesul de realizare a unei aplicații Web. ^[1]

Conținutul unei aplicații Web poate fi accesat de pe o varietate de dispozitive (PC, telefoane mobile, tablete, PDA-uri, mai nou console și multe altele), sporind și mai mult posibilitățile utilizatorilor să aibă acces și să interacționeze cu informațiile. Iar aceste aplicații pot interacționa între ele mai ușor, sporind, astfel, accesul clienților la date. Nu în ultimul rând, acestea sunt ușor de instalat (prin intermediul update-urilor se poate ajunge la toți utilizatorii deodată), se adaptează la un volum de informații din ce în ce mai mare și sunt mai bine securizate (găsindu-se pe servere dedicate) și ușor de creat. ^[1]

Peste 91% dintre organizatorii de evenimente consideră că aplicațiile pentru evenimente sunt utile în munca lor, conform unui studiu realizat de către Event Manager Blog la începutul anului. Aproape jumătate (46%) dintre participanții la studiu au declarat că folosesc în mod constant în prezent o aplicație pentru evenimente sau conferințe, în timp ce 26% au declarat că intenționează să utilizeze astfel de aplicații în viitorul apropiat. Mai mult de jumătate dintre

organizatorii de evenimente și agențiile de marketing participante la studiu (53%) nu utilizează în prezent niciun software pentru organizarea evenimentelor.^[2]

Astfel, realizarea aplicației „EvenTourTM” are ca scop o informare mult mai eficientă a locuitorilor dintr-un oraș asupra tuturor evenimentelor din orașul respectiv și, totodată, o gestionare mult mai bună a acelor evenimente. Oamenii nu mai sunt puși în situația în care trebuie să aștepte programul săptămânal sau lunar al evenimentelor din orașul lor publicat într-un ziar sau revistă, ori încercând să afle de la diverse persoane care posedă mai multe informații în legătură cu organizarea lor, obținute din surse nesigure deoarece informațiile la care au acces toți sunt destul de evazive. Pur și simplu, vor fi la un click distanță de fiecare gen de eveniment în parte, pentru care au un interes, și nu numai, și, totodată, vor fi capabili să vizualizeze informațiile oferite în detaliu și o galerie foto a fiecărui eveniment în parte, vor putea face schimb de opinii, iar pe lângă aceasta, vor fi înștiințați prin e-mail despre evenimentul ales.

1.2 Tema proiectului

Aplicația „EvenTourTM” a fost realizată în urma creșterii numărului de evenimente care au loc în orașul Timișoara și, totodată, a dorinței de informare în prealabil asupra acestora și a interesului oamenilor pentru vizionarea acestor evenimente care devin din ce în ce mai spectaculoase și merită promovate în mediul online. Prin urmare, aceasta aplicație urmărește promovarea tuturor evenimentelor de orice gen care se desfășoară pe teritoriul orașului Timișoara, asigurând o mai bună informare a cetățenilor orașului, dar și a eventualilor vizitatori în legătură cu desfășurarea lor.

Aplicația prezintă trei module, destinate administratorului, utilizatorilor care și-au creat cont în ea și una destinată simplilor vizitatori ai aplicației. Administratorul are posibilitatea de a adăuga evenimente aferente fiecărui gen în parte, poate șterge sau edita anumite evenimente dacă în prezentarea lor se află o greșeală care trebuie corectată sau, pur și simplu, o informație trebuie actualizată. Totodată, el poate adăuga galerie foto pentru fiecare eveniment în parte, putând modifica acele fotografii oricând este necesar și mai poate vizualiza lista cu cei care au cont în această aplicație.

În cadrul modulului pentru utilizatori, în primul rând, aceștia trebuie să își creeze cont în aplicație, folosind un nume, un e-mail și o parolă, apoi pot vizualiza pe categorii evenimentele ce urmează să se desfășoare, pot căuta un anumit eveniment dorit după denumirea acestuia în cadrul fiecărei categorii, iar pentru fiecare eveniment în parte au posibilitatea de a vizualiza toate detaliile aferente evenimentului respectiv, împreună cu o galerie foto a lui și de

a își trimite pe e-mail înștiințare cu acel eveniment. Totodată, în cadrul fiecărui eveniment, utilizatorii pot lăsa propriul comentariu și vizualiza comentariile altor utilizatori și mai pot participa la o conversație de grup/forum cu ceilalți utilizatori care se află și ei în momentul respectiv în chat-ul de conversații. După ce evenimentele care au fost programate s-au desfășurat, acestea se mută automat într-o altă secțiune denumită „Istoric” unde se găsesc toate evenimentele care s-au desfășurat deja, pentru o arhivă a acestora.

Modulul destinat simplilor vizitatori este asemănător cu modulul destinat utilizatorilor cu cont în aplicație, în sensul că permite accesul la vizualizarea și căutarea fiecărui eveniment în parte pe categorii, la adăugarea de comentarii pentru fiecare eveniment, însă nu permite accesul la funcția de e-mail a aplicației, la posibilitatea adăugării de noi comentarii și nici la funcția de chat/forum a aplicației.

1.3 Structura pe capitole

Lucrarea de față se întinde pe un număr de 63 de pagini, structurate pe 6 capitole, care cuprind un total de 40 de figuri și o tabelă. Bibliografia consultată pe parcursul realizării acestui proiect cuprinde 24 de surse bibliografice.

➤ CAPITOLUL 1. INTRODUCERE

Acest capitol cuprinde o scurtă introducere în ceea ce înseamnă aplicațiile web și avantajele utilizării lor, tendința generală pentru folosirea unor aplicații dedicate pentru promovarea evenimentelor de orice gen prin intermediul platformei web și câteva informații legate de aplicația propusă.

➤ CAPITOLUL 2. FUNDAMENTAREA TEORETICĂ

Acest capitol cuprinde toate tehnologiile, limbajele de programare și mediile de dezvoltare necesare pentru crearea aplicației și o scurtă descriere a acestora.

➤ CAPITOLUL 3. DESCRIEREA GENERALĂ A PROIECTULUI

În acest capitol am prezentat o descriere sumară a aplicației, schema bloc și funcționalitățile pe care aceasta va trebui să le aibă.

➤ **CAPITOLUL 4. PROIECTAREA ÎN DETALIU**

În acest capitol sunt prezentate considerații privind implementarea, arhitectura programului, cele trei module și descrierea bazei de date. De asemenea, sunt prezentate detaliat funcționalitățile aplicației și modul de realizare.

➤ **CAPITOLUL 5. MANUAL DE UTILIZARE**

În acest capitol sunt prezentate interfețele aplicației și modul în care acestea trebuie utilizate astfel încât utilizatorul să poată folosi cu ușurință aplicația.

➤ **CAPITOLUL 6. CONCLUZII**

În acest capitol de încheiere sunt prezentate ideile de bază și inovatoare ale aplicației, cerințele la care răspunde aceasta și care ar fi eventualele soluții de îmbunătățire și dezvoltare ale acesteia.

CAPITOLUL 2

FUNDAMENTAREA TEORETICĂ

2.1 ASP .NET Framework

.NET Framework (pronunțat Dot Net) este o arhitectură software care rulează în principal pe Microsoft Windows, care include o gamă largă de librării și suportă diferite limbaje de programare, astfel permițând interoperabilitatea limbajelor.^[3]

.NET Framework este o tehnologie care acceptă construirea și rularea aplicațiilor Windows și a serviciilor web. Este proiectată să îndeplinească următoarele obiective:^[4]

- Pentru a furniza un mediu de programare consistent orientat pe obiect, indiferent dacă codul obiectului este stocat și executat local, executat local, dar distribuit web sau executat de la distanță.^[4]
- Pentru a furniza un mediu de execuție a codului, care reduce la minimum implementarea software și conflictele de versionare.^[4]
- Pentru a furniza un mediu de execuție a codului care promovează executarea sigură a codului, inclusiv codul creat de o terță parte necunoscută sau de mică încredere.^[4]
- Pentru a oferi un mediu de execuție a codului care elimină problemele de performanță ale mediilor scriptate sau interpretate.^[4]
- Pentru a face experiența dezvoltatorului să fie consecventă în diferite tipuri de aplicații variate, cum ar fi aplicațiile bazate pe Windows și aplicațiile bazate pe Web.^[4]
- Pentru a construi toate comunicările pe standarde din industrie pentru a asigura că orice cod bazat pe .NET Framework se integrează cu orice alt cod.^[4]

.NET Framework constă din timpul de rulare al limbajului comun (CLR) și biblioteca de clase .NET Framework. **Runtime-ul limbajului comun** este fundamentul .NET Framework. Gândiți-vă la timpul de execuție ca la un agent care gestionează codul în momentul execuției, oferind servicii de bază, cum ar fi gestionarea memoriei, gestionarea thread-ului și remoting-ul, în același timp, aplicând siguranța de tip strict și alte forme de precizie a codului care promovează securitatea și robustetea. De fapt, conceptul de gestionare a codurilor este un principiu fundamental al timpului de rulare. Codul care vizează timpul de execuție este cunoscut sub numele de cod gestionat, în timp ce codul care nu vizează timpul de execuție este cunoscut

ca cod negestionat. Biblioteca de clase este o colecție cuprinzătoare, orientată spre obiecte, de tipuri reutilizabile pe care le utilizați pentru a dezvolta aplicații, de la aplicațiile tradiționale de linie de comandă sau de interfață grafică (GUI) la aplicații bazate pe cele mai noi inovații oferite de ASP.NET, cum ar fi Web Formulare și servicii web XML. ^[4]

Biblioteca de clasă .NET Framework este o colecție de tipuri reutilizabile care se integrează strâns cu timpul de rulare a limbajului comun. Biblioteca de clase este orientată pe obiect, oferind tipuri din care propriul cod gestionat derivă funcționalitate. Acest lucru nu numai că face ca tipurile .NET Framework să fie ușor de utilizat, dar reduce și timpul asociat cu învățarea noilor funcții ale .NET Framework. În plus, componente terțe se integrează perfect cu clasele din .NET Framework. ^[4]

După cum vă așteptați de la o bibliotecă de clase orientată pe obiecte, tipurile .NET Framework vă permit să îndepliniți o serie de sarcini de programare comune, inclusiv gestionarea șirurilor, colectarea datelor, conectivitatea bazei de date și accesul la fișiere. În plus față de aceste sarcini comune, biblioteca de clase include tipuri care acceptă o varietate de scenarii de dezvoltare specializate. Puteți utiliza .NET Framework pentru a dezvolta următoarele tipuri de aplicații și servicii: ^[4]

- Aplicații pentru console^[4]
- Aplicații GUI pentru Windows (Windows Forms). ^[4]
- Aplicații Windows Presentation Foundation (WPF). ^[4]
- Aplicații ASP.NET^[4]
- Servicii Windows. ^[4]
- Aplicații orientate către servicii utilizând Windows Communication Foundation (WCF).^[4]

2.2 MVC Pattern

Model-View-Controller (MVC) este un model arhitectural care separă o aplicație în trei componente logice principale: modelul, vizualizarea și controlerul. Fiecare dintre aceste componente este construită pentru a trata aspecte specifice de dezvoltare ale unei aplicații. MVC este unul dintre cele mai frecvent utilizate cadre de dezvoltare web standard pentru industrie pentru a crea proiecte scalabile și extensibile. ^[5]

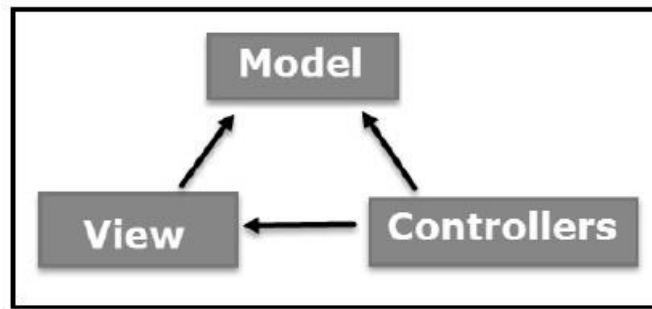


Figura 2.1 Pattern-ului MVC^[5]

În continuare sunt descrise componentele MVC:

Model

Componenta Model corespunde întregii logici legate de date cu care utilizatorul lucrează. Aceasta poate reprezenta fie datele care sunt transferate între componentele View și Controller, fie orice alte date referitoare la logica de afaceri. De exemplu, un obiect Client va prelua informațiile despre client din baza de date, le va manipula și va actualiza datele înapoi în baza de date sau le va folosi pentru a reda date. ^[5]

View

Componenta View este utilizată pentru toată logica UI a aplicației. De exemplu, vizualizarea Clientului va include toate componentele UI, cum ar fi casete de text, meniuri derulante etc., cu care utilizatorul final interacționează. ^[5]

Controller

Controlerele acționează ca o interfață între componentele Model și View pentru a procesa toată logica de afaceri și solicitările primite, manipulează datele folosind componenta Model și interacționează cu View-urile pentru a reda rezultatul final. De exemplu, controlorul pentru clienți va gestiona toate interacțiunile și intrările din Clientul de vizualizare și va actualiza baza de date utilizând Modelul Clienților. Același controler va fi utilizat pentru a vizualiza datele Clientului. ^[5]

Întreaga idee din spatele utilizării modelului de design reprezintă separarea conceptelor. ^[6]

ASP.NET acceptă trei modele majore de dezvoltare: pagini web, formulare web și MVC (Model View Controller). Cadrul ASP.NET MVC este un cadru de prezentare ușor, foarte testabil, care este integrat cu caracteristicile existente ASP.NET, cum ar fi paginile master,

autentificarea, etc. În cadrul .NET, acest cadru este definit în assembly-ul System.Web.Mvc. Cea mai recentă versiune a cadrului MVC este 5.0. Utilizăm Visual Studio pentru a crea aplicații ASP.NET MVC, care pot fi adăugate ca șablon în Visual Studio. ^[5]

ASP.NET MVC este un cadru de dezvoltare web de la Microsoft care combină eficiența și curățenia arhitecturală model-view-controller (MVC), cele mai actualizate idei și tehnici de dezvoltare agilă, și cele mai bune părți ale platformei ASP.NET existente. ^[7]

ASP.NET MVC oferă următoarele caracteristici:

- Ideal pentru dezvoltarea aplicațiilor complexe, dar ușoare. ^[5]
- Oferă un cadru extensibil și conectabil, care poate fi ușor înlocuit și personalizat. De exemplu, dacă nu doriți să utilizați Razor încorporat sau ASPX View Engine, atunci puteți utiliza orice alte motoare de vizualizare terțe sau chiar să le personalizați pe cele existente. ^[5]
- Utilizează proiectarea bazată pe componente a aplicației prin împărțirea logică a acestora în componentele Model, View și Controller. Acest lucru permite dezvoltatorilor să gestioneze complexitatea proiectelor la scară largă și să lucreze la componente individuale. ^[5]
- Structura MVC îmbunătățește dezvoltarea test-driven și testabilitatea aplicației, deoarece toate componentele pot fi proiectate pe baza interfeței și testate folosind obiecte mock. Prin urmare, ASP.NET MVC Framework este ideal pentru proiecte cu o mare echipă de dezvoltatori web. ^[5]
- Suportă toate funcționalitățile vaste existente ASP.NET, cum ar fi autorizarea și autentificarea, paginile principale, legarea datelor, controlul utilizatorilor, apartenența, rutarea ASP.NET etc. ^[5]
- Nu folosește conceptul de View State (care este prezent în ASP.NET). Acest lucru ajută la construirea aplicațiilor, care sunt ușoare și oferă un control deplin dezvoltatorilor. ^[5]

Astfel, puteți considera MVC Framework ca un framework major construit pe baza ASP.NET care oferă un set mare de funcționalități adăugate care se concentrează pe dezvoltarea bazată pe componente și pe testare. ^[5]

Anterior, am studiat fluxul de arhitectură la nivel înalt al cadrului MVC. Acum să aruncăm o privire la modul în care are loc execuția unei aplicații MVC atunci când există o anumită solicitare din partea clientului. Următoarea diagramă ilustrează fluxul: ^[5]

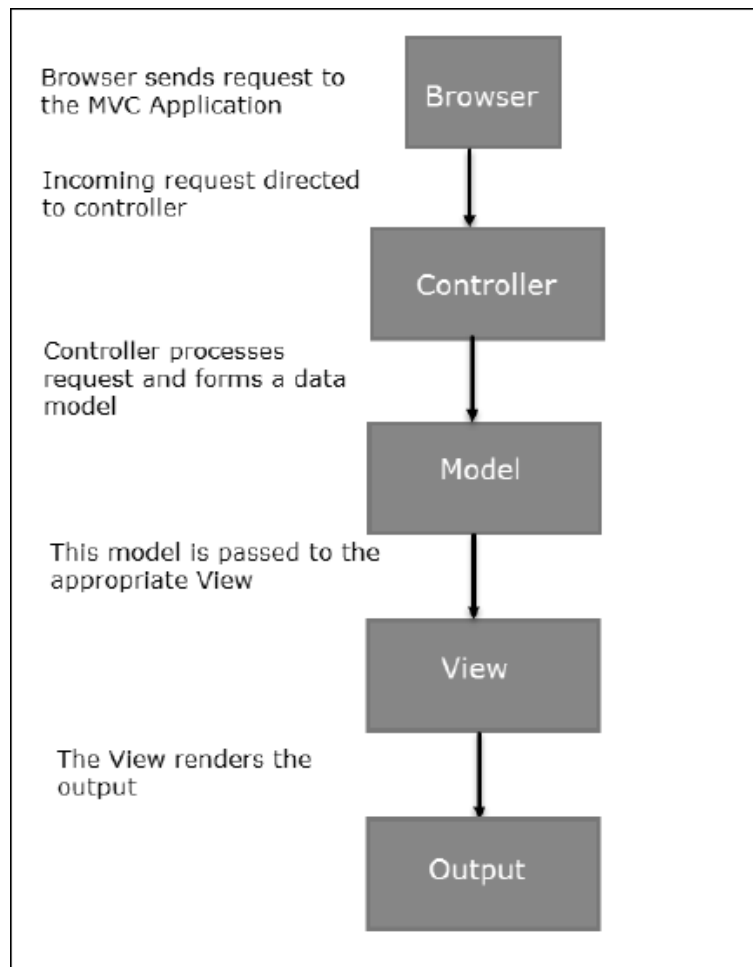


Figura 2.2 Modul de funcționare al pattern-ului MVC^[5]

Pași:

Pasul 1 - Browserul client trimite cererea către Aplicația MVC. ^[5]

Pasul 2 - Global.ascx primește această solicitare și efectuează rutarea bazată pe adresa URL a cererii primite folosind obiectele RouteTable, RouteData, UrlRoutingModule și MvcRouteHandler. ^[5]

Pasul 3 - Această operație de rutare apelează controlerul adecvat și îl execută folosind obiectul IControllerFactory și metoda Execute a obiectului MvcHandler. ^[5]

Pasul 4 – Controlerul prelucrează datele folosind Modelul și invocă metoda corespunzătoare folosind obiectul ControllerActionInvoker^[5]

Pasul 5 - Modelul procesat este apoi trecut la View, care la rândul său face rezultatul final. ^[5]

2.3 Entity Framework

Entity Framework poate face multe pentru noi ca dezvoltatori Microsoft. În primul rând, este capabil de expunere a bazei de date ca un set de obiecte.^[8]

Entity Framework este un framework ORM, open-source, folosit pentru aplicațiile .NET, acceptat de Microsoft.^[9]

Permite dezvoltatorilor să lucreze cu date folosind obiecte din clase specifice domeniului, fără a se concentra pe tabelele și coloanele bazei de date de bază unde sunt stocate aceste date. Cu Entity Framework, dezvoltatorii pot lucra la un nivel mai înalt de abstractizare atunci când se ocupă de date și pot crea și menține aplicații orientate către date cu mai puțin cod în comparație cu aplicațiile tradiționale.^[10]

Un mare avantaj al lui Entity Framework este posibilitatea folosirii interogărilor de tipul LINQ. LINQ este un model de programare care introduce interogări ca un concept de primă clasă în orice limbaj Microsoft .NET Framework.^[11]

Entity framework are o serie de caracteristici:

- **Cross-platform:** EF Core is a cross-platform framework which can run on Windows, Linux and Mac.^[10]
- **Modelare:** EF (Entity Framework) creează un EDM (Entity Data Model) bazat pe entități POCO (Plain Old CLR Object) entități cu proprietăți get / set de diferite tipuri de date. Acesta folosește acest model pentru a interoga sau pentru a salva datele entității în baza de date de bază.^[10]
- **Interogare:** EF ne permite să utilizăm interogările de tipul LINQ pentru a prelua date din baza de date principal. Provider-ul bazei de date va traduce aceste întrebări LINQ în limbajul de interogare specific bazei de date (de ex. SQL pentru o bază de date relațională). EF ne permite, de asemenea, să executăm interogări SQL brute direct în baza de date.^[10]
- **Evidența modificărilor:** EF ține evidența modificărilor apărute la instanțele entităților (valorile proprietății) care trebuie trimise la baza de date.^[10]
- **Salvarea:** EF execută comenzi INSERT, UPDATE și DELETE în baza de date pe baza modificărilor apărute la entitățile dvs. atunci când apeleți la metoda SaveChanges (). EF oferă de asemenea metoda SaveChangesAsync () asincronă.^[10]
- **Concurență:** EF utilizează implicit Optimistic Concurrency pentru a proteja modificările de suprascriere efectuate de un alt utilizator, deoarece datele au fost preluate din baza de date.^[10]

- **Tranzacții:** EF efectuează gestionarea automată a tranzacțiilor în timp ce interogă sau salvează date. De asemenea, oferă opțiuni pentru a personaliza gestionarea tranzacțiilor.^[10]
- **Caching:** EF include primul nivel de caching din cutie. Deci, interogarea repetată va returna date din cache în loc să lovească baza de date.^[10]
- **Convențiile Build-In:** EF respectă convențiile asupra modelului de programare a configurației și include un set de reguli implicite care configurează automat modelul EF.^[10]
- **Configurații:** EF ne permite să configurăm modelul EF utilizând atributele de adnotare a datelor sau API-ul Fluent pentru a suprascrie convențiile implicite.^[10]
- **Migrații:** EF oferă un set de comenzi de migrare care pot fi executate pe NuGet Package Manager Console sau pe Interfața liniei de comandă pentru a crea sau gestiona schema bazei de date subiacente.^[10]

Microsoft a introdus Entity Framework în 2008 cu .NET Framework 3.5. De atunci, a lansat multe versiuni ale Entity Framework. În prezent, există două cele mai recente versiuni ale Entity Framework: EF 6 și EF Core.^[10]

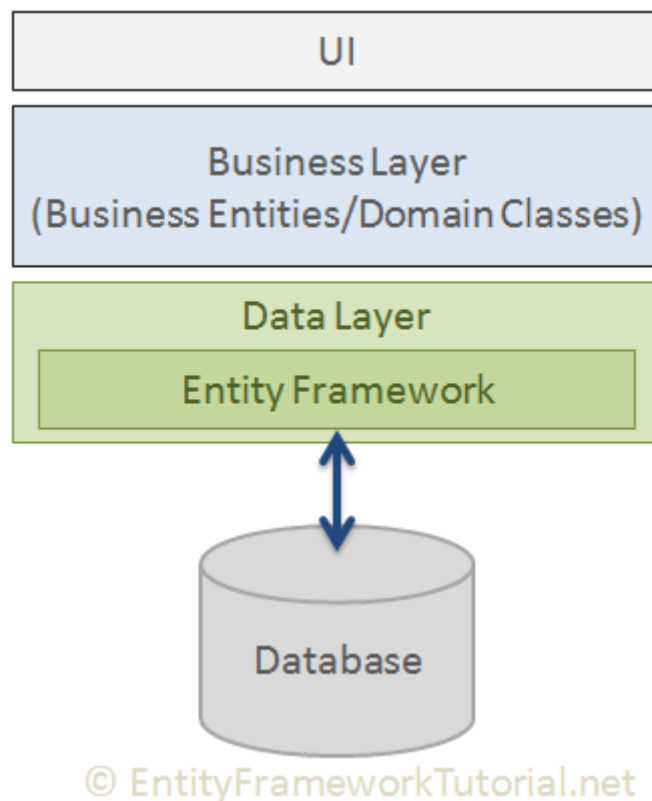


Figura 2.3 - Modul de funcționare al pachetului Entity Framework^[10]

2.4 Microsoft SQL Server

Microsoft SQL Server este un sistem relațional de gestionare a bazelor de date dezvoltat de Microsoft. Ca server de baze de date, este un produs software cu funcția principală de stocare și preluare a datelor, așa cum este solicitat de alte aplicații software - care poate rula fie pe același computer sau pe un alt computer de pe o rețea (inclusiv pe Internet). Microsoft comercializează cel puțin o duzină de ediții diferite de Microsoft SQL Server, destinate unor audiențe diferite și pentru sarcini de lucru, de la mici aplicații cu o singură mașină până la aplicații mari care se confruntă cu mulți utilizatori concomitenți pe internet.^[12]

Microsoft SQL Server include mai multe opțiuni pentru a face procesul de interogare mai rapid.^[13]

Ce este SQL Server:

- Este un software, dezvoltat de Microsoft, care este implementat din specificația RDBMS.^[14]
- Este, de asemenea, un ORDBMS.^[14]
- Este dependent de platformă.^[7]
- Este atât software GUI, cât și software bazat pe comenzi.^[14]
- Suporta limbajul SQL (SEQUEL), care este un produs IBM, non-procedural, baza de date comună și limbaj insensitive, adică nu face diferența între litere mari și litere mici^[14]

Utilizarea SQL Server:

- Pentru a crea baze de date.^[14]
- Pentru a menține bazele de date.^[14]
- Pentru a analiza datele prin serviciile de analiză SQL Server (SSAS).^[14]
- Pentru a genera rapoarte prin intermediul Serviciilor de raportare SQL Server (SSRS).^[14]
- Pentru a efectua operațiuni ETL prin serviciile de integrare SQL Server (SSIS).^[14]

Componentele SQL Server:

SQL Server funcționează în arhitectura client-server, deci acceptă două tipuri de componente - (a) Stație de lucru și (b) Server.^[14]

- Componentele **stației de lucru** sunt instalate în fiecare dispozitiv / operator de server SQL. Acestea sunt doar interfețe pentru a interacționa cu componentele serverului. Exemplu: SSMS, SSCM, Profiler, BIDS, SQLEM etc. ^[14]
- Componentele serverului sunt instalate în serverul centralizat. Acestea sunt servicii. Exemplu: SQL Server, SQL Server Agent, SSIS, SSAS, SSRS, browser SQL, SQL Server căutare text complet etc. ^[14]

Instanța SQL Server:

- O instanță este o instalare a SQL Server. ^[14]
- O instanță este o copie exactă a aceluiași software. ^[14]
- Dacă instalăm de „n” ori, atunci se vor crea instanțe „n”. ^[14]
- Există două tipuri de instanțe în SQL Server: a) Default b) Numit. ^[14]
- O singură instanță implicită va fi acceptată într-un singur server. ^[14]
- Mai multe instanțe numite vor fi acceptate într-un singur server. ^[14]
- Instanța implicită va lua numele serverului ca nume de instanță. ^[14]
- Numele implicit al serviciului de instanță este MSSQLSERVER. ^[14]

Avantajele instanțelor:

- Pentru a instala diferite versiuni într-o singură mașină. ^[14]
- Pentru a reduce costul. ^[14]
- Pentru a menține separat mediile de producție, dezvoltare și testare. ^[14]
- Pentru a reduce problemele temporare ale bazei de date. ^[14]
- Pentru a separa privilegiile de securitate. ^[14]
- Pentru a menține serverul de așteptare. ^[14]

2.5 Limbajul C#

C # (pronunțat „Vezi Rechin”) este un limbaj de programare modern, orientat pe obiecte și type-safe. C # își are rădăcinile în familia de limbi C și va fi imediat familiar cu programatorii C, C ++, Java și JavaScript. ^[15]

Visual C # moștenește multe dintre cele mai bune caracteristici ale C ++ și Microsoft Visual Basic, dar puține dintre incoerențele și anacronismele, ceea ce duce la un curat și un limbaj mai logic. ^[16]

C # este un limbaj orientat pe obiecte, dar C # include suplimentar suport pentru programarea orientată către componente. Proiectarea software-ului contemporan se bazează tot mai mult pe componentele software sub forma unor pachete de funcționalități independente și auto-descrie. Cheia pentru astfel de componente este faptul că prezintă un model de programare cu proprietăți, metode și evenimente. Au atribute care furnizează informații declarative despre componentă. Acestea includ propria lor documentație. C # oferă construcții de limbaj pentru a susține direct aceste concepte, făcând din C # un limbaj natural în care să creezi și să folosești componente software.^[15]

Câteva caracteristici C # ajută la construcția aplicațiilor robuste și durabile. Colectarea gunoiului recuperează automat memoria ocupată de obiectele neutilizate care nu pot fi atinse. Tratarea excepțiilor oferă o abordare structurată și extensibilă pentru detectarea și recuperarea erorilor. Designul type-safe al limbajului face imposibilă citirea de la variabilele neinițializate, la tablourile de index dincolo de limitele lor sau la efectuarea cast-urilor de tip neverificate.^[17]

C # are un sistem de tip unificat. Toate tipurile C #, inclusiv tipurile primitive, cum ar fi int și dublu, moștenesc de la un singur obiect de rădăcină. Astfel, toate tipurile împărtășesc un set de operații comune, iar valorile de orice tip pot fi stocate, transportate și operate în mod consecvent. Mai mult, C # acceptă atât tipuri de referință definite de utilizator, cât și tipuri de valori, permițând alocarea dinamică a obiectelor, precum și stocarea în linie a structurilor ușoare.^[15]

Pentru a se asigura că programele și bibliotecile C # pot evolua în timp într-o manieră compatibilă, s-a pus mult accent pe **versionare** în design-ul C #. Multe limbaje de programare acordă puțină atenție acestei probleme. Drept urmare, programele scrise în celelalte limbi se defectează mai des decât este necesar atunci când sunt introduse versiuni mai noi ale bibliotecilor dependente. Aspectele de design ale lui C # care au fost direct influențate de considerente de versionare includ modificatorii virtuali și de override, reguli pentru rezolvarea suprascrierii metodei și suportul pentru declarațiile de membru ale interfeței explicite.^[15]

În versiunile mai recente, C # a cuprins alte paradigme de programare. C # a inclus caracteristici care acceptă tehnici de programare funcționale precum expresiile lambda. Alte funcții noi acceptă separarea datelor și algoritmilor, cum ar fi pattern matching.^[15]

2.6 Razor (CSHTML)

Razor este o sintaxă de marcare care permite să încorporați codul bazat pe server în paginile web folosind C # și VB.Net. Nu este un limbaj de programare. Este un limbaj de markup pe partea de server. Razor nu are legături cu ASP.NET MVC, deoarece Razor este un engine cu scop general de șablon. Se poate utiliza oriunde pentru a genera ieșire ca HTML. Doar că ASP.NET MVC a implementat un motor de vizualizare care ne permite să folosim Razor în interiorul unei aplicații MVC pentru a produce HTML.^[18]



Figura 2.4 - Modul de funcționare al Razor (CSHTML)^[18]

Veți avea un fișier șablon, care este un mix de text literal și câteva blocuri de cod. Se combină șablonul respectiv cu unele date sau cu un model specific în care șablonul specifică unde ar trebui să apară datele, apoi se execută șablonul pentru a genera ieșirea.^[18]

Razor Vs ASPX

- Razor este foarte similar cu modul în care funcționează fișierele ASPX. Fișierele ASPX sunt șabloane, care conțin text literal și un cod C # care specifică unde ar trebui să apară datele dvs.^[18]
- Fișierele ASPX au o dependență de timpul de rulare ASP.NET pentru a fi disponibile pentru a analiza și executa acele fișiere ASPX. Razor nu are astfel de dependențe.^[18]
- Spre deosebire de fișierele ASPX, Razor are câteva obiective de proiectare diferite.^[18]

Obiective

Microsoft a dorit ca Razor să fie ușor de utilizat și ușor de învățat și să lucreze în interiorul unor instrumente precum Visual Studio, astfel încât IntelliSense să fie disponibil, debugger-ul să fie disponibil, dar au vrut ca Razor să nu aibă legături cu o tehnologie specifică, cum ar fi ASP.NET sau ASP.NET MVC. Dacă sunteți familiarizați cu ciclul de viață al unui fișier ASPX, atunci sunteți probabil conștienți de faptul că există o dependență de runtime ASP.NET care să fie disponibilă pentru a analiza și executa acele fișiere ASPX. Microsoft a dorit ca Razor să fie inteligent, pentru a facilita munca unui dezvoltator.^[18]

2.7 Bootstrap

Bootstrap este cel mai popular front-end framework din ultima perioadă. Este o aplicație elegantă, intuitivă și puternică pentru o dezvoltare web mai rapidă și mai ușoară. Utilizează HTML, CSS și Javascript.^[19]

Bootstrap-ul aduce un incredibil sistem receptiv de tabele și CSS de bază, inclusiv clase extensibile pentru implementare și îmbunătățirea stilului pentru diverse elemente, de la tipografie, butoane, tabele, formulare și imagini printre altele.^[20]

Avantaje Bootstrap:

- **Mobile first approach** - Bootstrap 3, cadru constă din primele stiluri mobile din întreaga bibliotecă, în loc de fișiere separate.^[19]
- **Suport pe majoritatea browser-elor** - Este suportat de toate browserele populare.^[19]
- **Ușor de învățat** - Doar cu cunoștințele de HTML și CSS, oricine poate începe cu Bootstrap. De asemenea, site-ul oficial Bootstrap are o documentație bună.^[19]
- **Responsive design** - CSS responsive al Bootstrap se adaptează la desktopuri, tablete și mobile.^[19]
- Oferă o soluție curată și uniformă pentru construirea unei interfețe pentru dezvoltatori.^[19]
- Conține componente încorporate frumoase și funcționale ușor de personalizat.^[19]
- De asemenea, oferă personalizare bazată pe web.^[19]
- Și cel mai important, este un open source.^[19]

Aplicații ale Bootstrap:

- **Scaffolding** (structură) - Bootstrap oferă o structură de bază cu Grid System, stiluri de legături și fundal.^[19]
- **CSS** - Bootstrap vine cu caracteristica setărilor CSS globale, elemente HTML fundamentale în stil și îmbunătățite cu clase extensibile și un sistem avansat de grid.^[19]
- **Componente** - Bootstrap conține peste o duzină de componente reutilizabile construite pentru a oferi iconografie, meniuri derulante, navigare, alerte, pop-over și multe altele.^[19]
- **Plugin-uri JavaScript** - Bootstrap conține peste o duzină de plugin-uri personalizate jQuery. Se pot include cu ușurință toate, sau unul câte unul.^[19]
- **Personalizări** – Se pot personaliza componentele Bootstrap, variabilele LESS și plugin-urile jQuery pentru a obține propria versiune.^[19]

2.8 Microsoft Visual Studio

Microsoft Visual Studio este un mediu de dezvoltare integrat (IDE) de la Microsoft. Este utilizat pentru a dezvolta programe de calculator, precum și site-uri web, aplicații web, servicii web și aplicații mobile. Visual Studio folosește platforme de dezvoltare software Microsoft, cum ar fi Windows API, Windows Forms, Windows Presentation Foundation, Windows Store și Microsoft Silverlight. Poate produce atât codul nativ, cât și codul gestionat.^[21]

Visual Studio include un editor de cod care acceptă IntelliSense (componenta de completare a codului), precum și refactorizarea codului. Debuggerul integrat funcționează atât ca un depanator la nivel sursă, cât și ca un depanator la nivel de mașină. Alte instrumente încorporate includ un profilator de coduri, proiectant pentru construirea aplicațiilor GUI, designer web, designer de clase și proiectare de scheme de baze de date. Accepta plug-in-uri care îmbunătățesc funcționalitatea la aproape toate nivelurile, inclusiv adăugarea de suport pentru sistemele de control sursă (precum Subversion și Git) și adăugarea de seturi de instrumente noi precum editori și designeri vizuali pentru limbaje sau seturi de instrumente specifice domeniului pentru alte aspecte ale ciclul de viață al dezvoltării software (cum ar fi clientul Azure DevOps: Team Explorer).^[21]

Visual Studio acceptă 36 de limbaje de programare diferite și permite editorului de coduri și depanatorului să sprijine (în grade diferite) aproape orice limbaj de programare, cu condiția să existe un serviciu specific limbajului. Limbajele încorporate includ C, [8] C ++, C ++ / CLI, Visual Basic .NET, C #, F #, [9] JavaScript, TypeScript, XML, XSLT, HTML și CSS. Asistența pentru alte limbi, cum ar fi Python, [10] Ruby, Node.js și M, printre altele, este disponibilă prin pluginuri. Java (și J #) au fost acceptate în trecut.^[21]

Cea mai simplă ediție a Visual Studio, ediția comunitară, este disponibilă gratuit. Sloganul pentru ediția Visual Studio Community este „IDE gratuit, cu caracteristici complete pentru studenți, open-source și dezvoltatori individuali”.^[21]

Versiunea Visual Studio acceptată în prezent este 2019.^[21]

2.9 Pusher

Pusher este un serviciu de găzduire care face foarte ușor adăugarea de date și funcționalități în timp real în aplicațiile web și mobile.^[22]

Pusher acționează ca un strat în timp real între servere și clienți. Pusher menține conexiuni persistente cu clienții - prin intermediul web-socket, dacă este posibil și revenind la conectivitatea bazată pe HTTP - astfel încât imediat ce serverele dvs. au date noi pe care vor să le împingă către clienți, pot face acest lucru prin intermediul Pusher.^[22]

CAPITOLUL 3

DESCIEREA GENERALĂ A PROIECTULUI

3.1 Prezentarea sumară a aplicației. Schema bloc

Proiectul realizat, numit „EvenTourTM” este o platformă web care are ca scop promovarea tuturor evenimentelor de orice gen care se desfășoară în orașul Timișoara și, totodată, o informare mai bună a cetățenilor orașului, dar și a eventualilor turiști, asupra acestor evenimente și a desfășurării lor.

După cum se poate observa în schema bloc de mai jos, aplicația a fost realizată în jurul a trei tipuri de utilizatori: utilizatori de tip abonați (utilizatori care și-au făcut cont în aplicație și au acces la funcția de e-mail, de a adăuga comentarii și vorbi pe forum), utilizatori de tip vizitatori (utilizatori care beneficiază de aceleași privilegii ca cei care au cont, în afară de cele trei funcții) și utilizatorul de tip administrator care are acces la toate datele aplicației și la gestionarea acestora (figura 3.1).

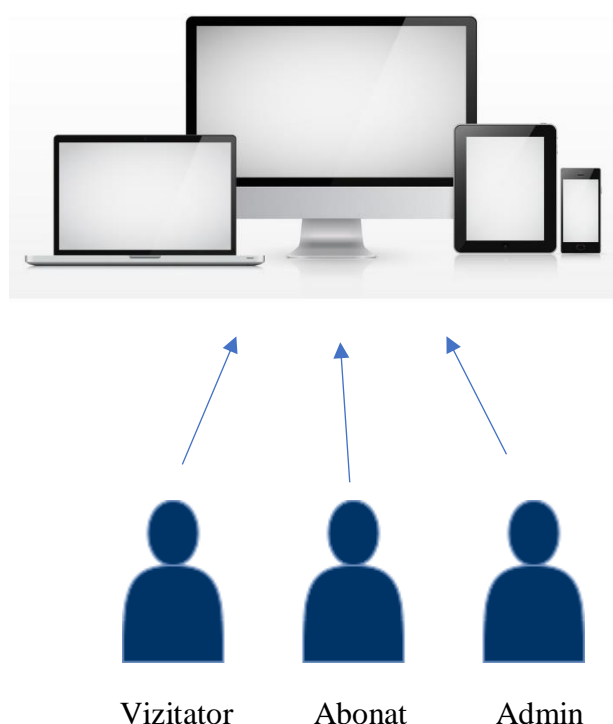


Figura 3.1 – Schema bloc a aplicației

3.2 Funcționalitățile aplicației

Aplicația va permite vizualizarea, adaugarea, modificarea, ștergerea, trimiterea de informații pe e-mail, adaugarea de comentarii și mesaje, în funcție de tipul utilizatorului.

Aplicația prezintă următoarele funcționalități, în funcție de utilizator:

- Pentru modulul **utilizator neautentificat/vizitator**:
 - Vizualizare evenimente pe categorii
 - Vizualizare detalii și comentarii legate de fiecare eveniment
 - Vizualizare galerie foto pentru fiecare eveniment
 - Căutare evenimente după denumire
 - Vizualizarea separată a evenimentelor care s-au desfășurat deja

În cadrul acestui modul, **utilizatorii neautentificați** pot vizualiza toate evenimentele pe categorii, pot căuta un anumit eveniment după denumire, iar pentru fiecare eveniment în parte, aceștia pot vedea și detaliile aferente acestor evenimente. Totodata, aceștia au acces la secțiunea de comentarii ale evenimentelor, pentru a putea vedea impresiile celorlalți utilizatori care au mai fost în trecut la aceste evenimente, și pot vizualiza o galerie foto în cadrul fiecărui eveniment, pentru o mai bună informare legată de evenimentul interesat. În același timp, pot vizualiza și o listă cu evenimentele care s-au desfășurat deja, împreună cu comentariile aferente lăsate de cei care au participat.

- Pentru modulul **utilizatorului autentificat/abonat**:
 - Creare cont
 - Autentificare
 - Vizualizare evenimente pe categorii
 - Vizualizare detalii și comentarii legate de fiecare eveniment
 - Posibilitatea de a adăuga comentarii pentru fiecare eveniment
 - Vizualizare galerie foto pentru fiecare eveniment
 - Trimiterea pe e-mail a informațiilor unui eveniment
 - Căutare evenimente după denumire
 - Trimiterea de mesaje utilizatorilor activi în aplicație
 - Vizualizarea separată a evenimentelor care s-au desfășurat deja

Utilizatorul autentificat se loghează cu datele aferente lui sau dacă este un utilizator fără cont, are posibilitatea de a își crea cont și apoi de a se loga cu datele furnizate. După această etapă, utilizatorul poate vizualiza toate evenimentele pe categorii, poate căuta un anumit

eveniment după denumire, iar pentru fiecare eveniment în parte, acesta poate vedea și detaliile aferente acestor evenimente. Totodată, acesta are acces la câte o galerie foto a fiecărui eveniment în parte, dar și la secțiunea de comentarii a evenimentelor, pentru a putea vedea impresiile celorlalți utilizatori care au mai fost în trecut la aceste evenimente. Utilizatorul poate lăsa și el comentarii, la rândul lui, la evenimentele la care a participat, dar și la cele la care urmează să participe. Pentru fiecare eveniment în parte, poate trimite pe e-mail-ul personal informațiile referitoare la evenimentele alese de el, iar pe lângă asta, mai poate trimite mesaje în cadrul aplicației către ceilalți utilizatori activi în momentul acela. În același timp, poate vizualiza și o listă cu evenimentele care s-au desfășurat deja, împreună cu comentariile aferente lăsate de cei care au participat.

- Pentru modulul **utilizatorului de tip administrator**:
 - Autentificare
 - Vizualizare, adăugare, editare și ștergere evenimente pe categorii
 - Vizualizare detalii și comentarii, plus editare detalii
 - Vizualizare galerie foto, adăugare, editare și ștergere fotografii
 - Căutare evenimente după categorie
 - Vizualizare, adăugare, editare și ștergere evenimente din istoric
 - Vizualizare utilizatori cu cont creat pe aplicație

Administratorul dispune de cele mai importante drepturi. Acesta se autentifică cu un user și o parolă primită special pentru el. După ce se autentifică, acesta poate căuta un anumit eveniment după denumire și poate vizualiza toate evenimentele și, în același timp, poate adăuga evenimente noi, poate edita evenimentele existente și poate șterge evenimente dacă este necesar acest lucru, iar în cadrul fiecărui eveniment poate vizualiza și comentariile aferente. Totodată, poate vizualiza galeria foto a fiecărui eveniment, iar pentru fiecare eveniment în parte poate adăuga, edita sau șterge fotografii, în funcție de ce este nevoie. Pe lângă acestea, el mai poate vizualiza lista cu evenimente trecute, pe care le poate edita sau șterge, și mai poate vizualiza lista cu utilizatorii care și-au creat cont în aplicație.

3.3 Planificarea lucrărilor

3.3.1 Ciclul de viață V

Modelul V este un tip de model SDLC în care procesul se execută într-o manieră secvențială în formă de V. Este cunoscut și sub denumirea de model de verificare și validare. Se bazează pe asocierea unei faze de testare pentru fiecare etapă de dezvoltare corespunzătoare. Dezvoltarea fiecărei etape asociate direct cu faza de testare. Faza următoare începe doar după finalizarea fazei anterioare, adică pentru fiecare activitate de dezvoltare, există o activitate de testare corespunzătoare acesteia.^[23] Modelul este reprezentat în figura 3.2.

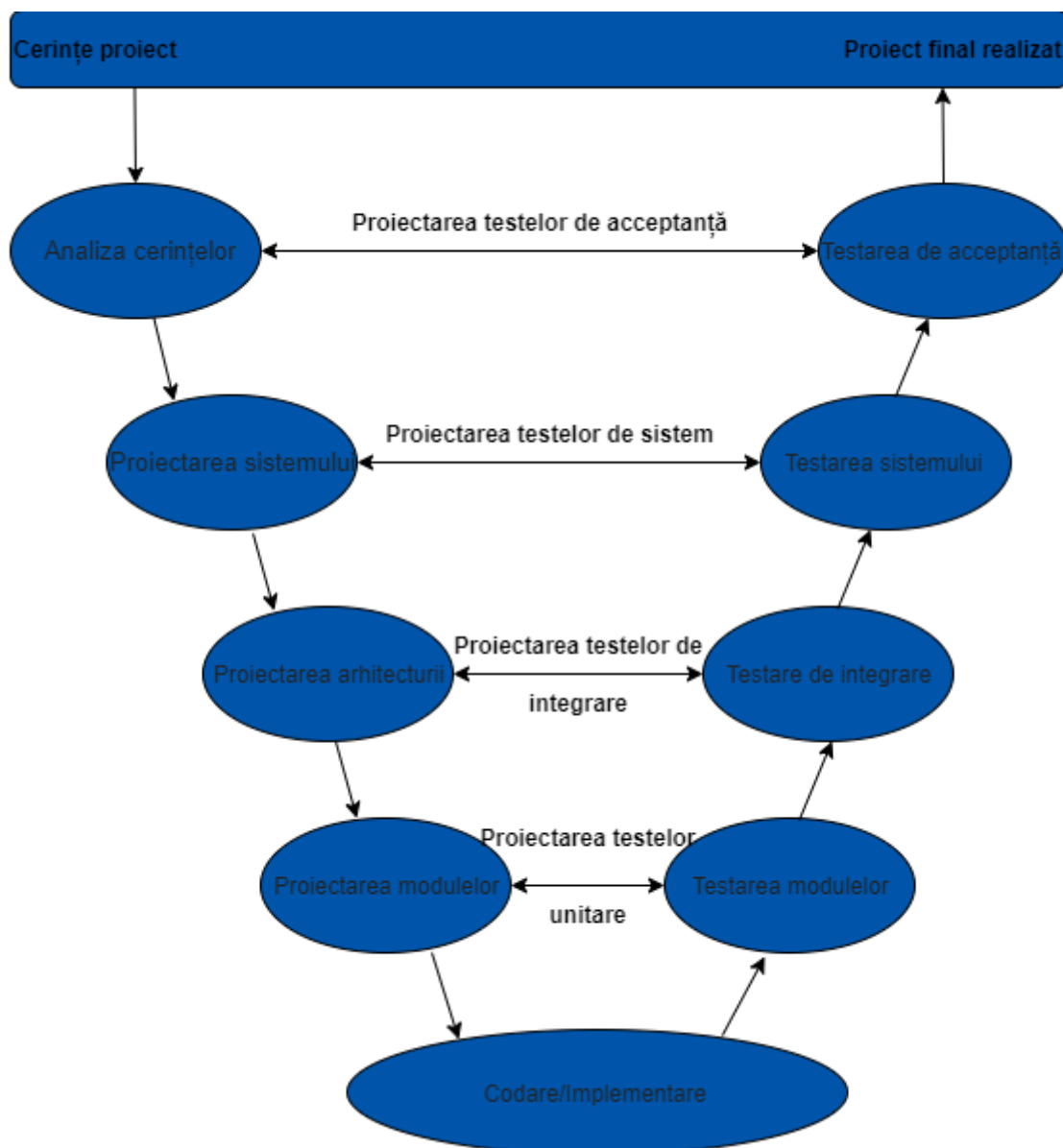


Figura 3.2 – Modelul în V

Ramura stângă de dezvoltare/proiectare:

- **Analiza cerințelor:** această fază conține o comunicare detaliată cu clientul pentru a înțelege cerințele și așteptările acestuia. Această etapă este cunoscută sub denumirea de Colectarea cerințelor. ^[23]
- **Proiectarea sistemului:** această fază conține proiectarea sistemului și configurarea completă a hardware-ului și comunicării pentru dezvoltarea produsului. ^[23]
- **Proiectarea arhitecturii:** proiectarea sistemului este defalcată în module, care ocupă diferite funcționalități. Transferul de date și comunicarea dintre modulele interne și lumea exterioară (alte sisteme) este înțeleasă clar și realizată foarte bine. ^[23]
- **Proiectarea modulelor:** în această fază, sistemul se descompune în module mici. Proiectarea detaliată a modulelor este specifică, cunoscută și sub denumirea de Design la nivel scăzut (LLD). ^[23]

Ramura dreaptă de testare:

- **Testarea Unitară:** planurile de testare a unității sunt dezvoltate în faza de proiectare a modulului. Aceste planuri de testare a unității sunt executate pentru a elimina erorile la nivel de cod sau unitate. ^[23]
- **Testare de integrare:** după finalizarea testării unitare, se efectuează testarea de integrare. În testarea de integrare, modulele sunt integrate și sistemul este testat. Testarea de integrare se realizează în faza de proiectare a arhitecturii. Acest test verifică comunicarea modulelor între ele. ^[23]
- **Testarea sistemului:** testarea sistemului testează aplicația completă cu funcționalitatea, inter dependența și comunicarea sa. Testează cerințele funcționale și non-funcționale ale aplicației dezvoltate. ^[23]
- **Testarea de acceptanță:** UAT se realizează într-un mediu utilizator care seamănă cu mediul de producție. UAT verifică dacă sistemul livrat îndeplinește cerințele utilizatorului și că sistemul este gata de utilizare în lumea reală. ^[23]

Avantaje:

- Acesta este un model extrem de disciplinat, iar etapele sunt completate pe rând. ^[13]
- Modelul V este utilizat pentru proiecte mici în care cerințele proiectului sunt clare. ^[23]
- Simplu și ușor de înțeles și de utilizat. ^[23]

- Acest model se concentrează pe activități de verificare și validare la începutul ciclului de viață, sporind astfel probabilitatea construirii unui produs fără erori și de bună calitate. ^[23]
- Permite managementului de proiect să urmărească progresul cu exactitate. ^[23]

3.3.2 Diagrama Gantt

Diagrama GANTT este un instrument folosit în planificarea proiectelor, evenimentelor și a muncii, în general. Urmărește etapele desfășurării unui proiect în funcție de durata acestora. Este o metoda des întâlnită în managementul proiectelor. A fost inventată de către Henry L. Gantt (inginer și sociolog de origine americană) în 1917 și a fost considerată o tehnică revoluționară la acea vreme, autorul ei fiind premiat pentru contribuțiile aduse în domeniul managementului. ^[24]

Mai exact, diagrama Gantt ilustrează grafic un program de activitate care ajută la planificarea, coordonarea și monitorizarea unor sarcini specifice unui proiect. Are forma unei matrici, cu două axe: una orizontală care indică perioada de timp necesară fiecărei activități (cu datele de începere și de finalizare ale sarcinii); alta verticală, care indică sarcinile ce trebuie îndeplinite. ^[24]

Sarcină	Data începerii	Număr de zile	Data finalizării
Analiza cerințelor	31.03.2020	3	02.04.2020
Realizarea specificațiilor	03.04.2020	7	09.04.2020
Proiectarea arhitecturii	10.04.2020	8	17.04.2020
Proiectarea modulelor	18.04.2020	9	26.04.2020
Codate/Implementare	27.04.2020	28	24.05.2020
Testare	25.05.2020	3	27.05.2020
Pordus final	28.05.2020	3	30.05.2020

Tabelul 1 – Planificarea lucrării

Diagrama Gantt aferentă lucrării (figura 3.3):

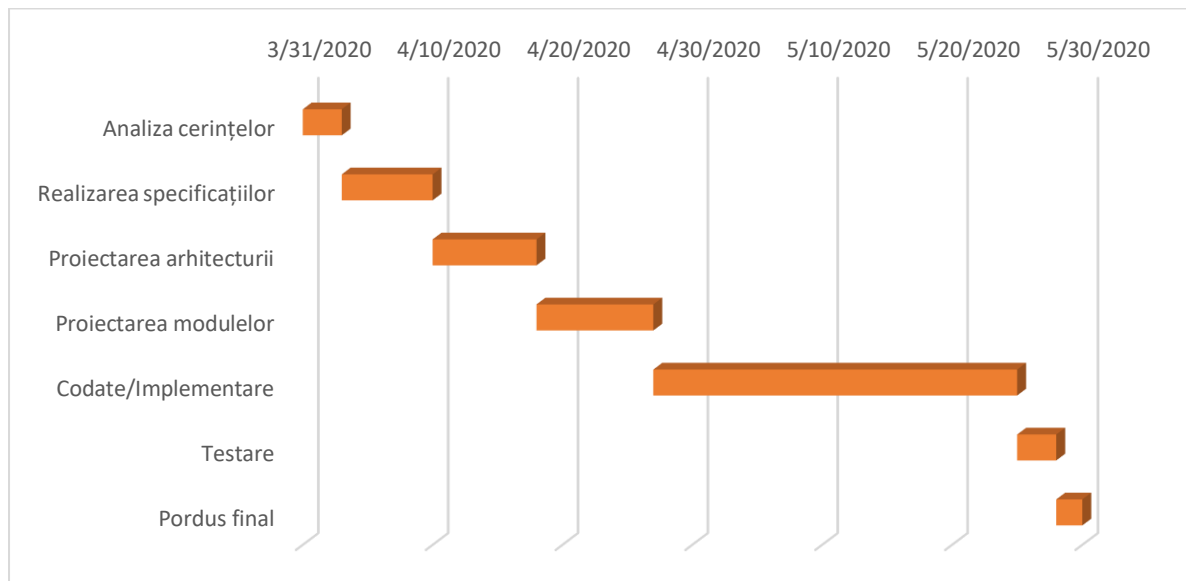


Figura 3.3 – Diagrama Gantt

CAPITOLUL 4

PROIECTAREA ÎN DETALIU

4.1 Considerații privind implementarea

Aplicația web dedicată evenimentelor din Timișoara, prezentată în lucrare, este realizată folosind tehnologia ASP .NET Framework, arhitectura MVC, limbajul C# pentru partea de server, și a sintaxei Razor împreună cu framework-ul Bootstrap pentru partea de client (figura 4.1).

Pentru partea de server, după cum am spus mai sus, am utilizat mediul de dezvoltare Visual Studio împreună cu limbajul C# și aplicația ASP .NET Framework, pentru a trimite request-uri HTTP de pe partea de client și pentru a putea manipula informațiile stocate în baza de date. După cum spuneam, am folosit principiul MVC (Model – View – Controller) care implică:

- **Controlere** pentru a gestiona cererile utilizatorilor și pentru a prelua date, prin utilizarea modelelor, aici aflându-se logica aplicației
- **Modele** pentru a interacționa cu baza de date și pentru a prelua informațiile transmise de utilizator
- **Vederi (View-uri)** pentru afișarea interfețelor cu utilizatorul

Pe lângă acestea, am utilizat framework-ul Entity Framework pentru a putea face legătura dintre aplicație și baza de date. Totodată, acesta permite interogări de tipul LINQ utilizând limbaj C#, fără a mai utiliza limbaj SQL.

Pentru comentarii am folosit un serviciu de host numit Pusher care face foarte ușor adăugarea de date și funcționalități în timp real în aplicațiile web și mobile. Alături de acesta am utilizat pe partea de client un script de JavaScript numită Vue care este folosit pentru a gestiona transmiterea comentariilor și actualizările în timp real ale lui Pusher.

Pentru partea de mesaje în timp real, am adăugat librăria SignalR la șablonul MVC 5 care simplifică adăugarea de funcționalități în timp real aplicației. Funcționalitatea în timp real creată permite serverului să împingă conținutul către clienți instantaneu. SignalR oferă un

API pentru crearea de apeluri de procedură de la server la client. Aceste proceduri apelează funcții JavaScript la clienți din codul .NET Framework al serverului (figura 4.1).

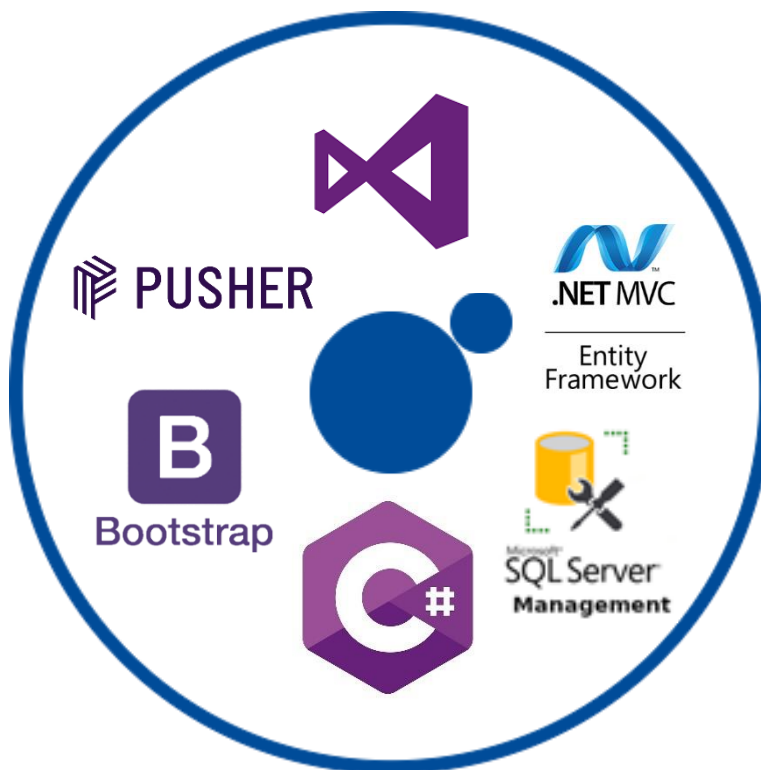


Figura 4.1 – Schema cu tehnologiile utilizate

4.2 Arhitectura programului

Aplicația este împărțită în trei module cu arhitecturi diferite fiecare. Arhitectura modulului Administrator este prezentată în figura 4.1, cea a modulului Utilizator autentificat în figura 4.2 și cea a modulului Utilizator neautentificat în figura 4.3.

Administratorul se autentifică cu un user și o parolă primite special pentru el. După ce se autentifică poate căuta evenimentele după denumire și poate vizualiza evenimentele. În același timp, poate adăuga evenimente noi, poate edita un anumit eveniment și poate șterge evenimente dacă este necesar acest lucru, iar în cadrul fiecărui eveniment poate vizualiza și comentariile aferente. Totodată, poate vizualiza galeria foto a fiecărui eveniment, iar pentru fiecare eveniment în parte poate adăuga, edita sau șterge fotografii, în funcție de ce este nevoie. Pe lângă acestea, el mai poate vizualiza lista cu evenimente care s-au desfășurat deja, pe care le poate edita sau șterge, și mai poate vizualiza lista cu utilizatorii care și-au creat cont pe aplicație (figura 4.2).

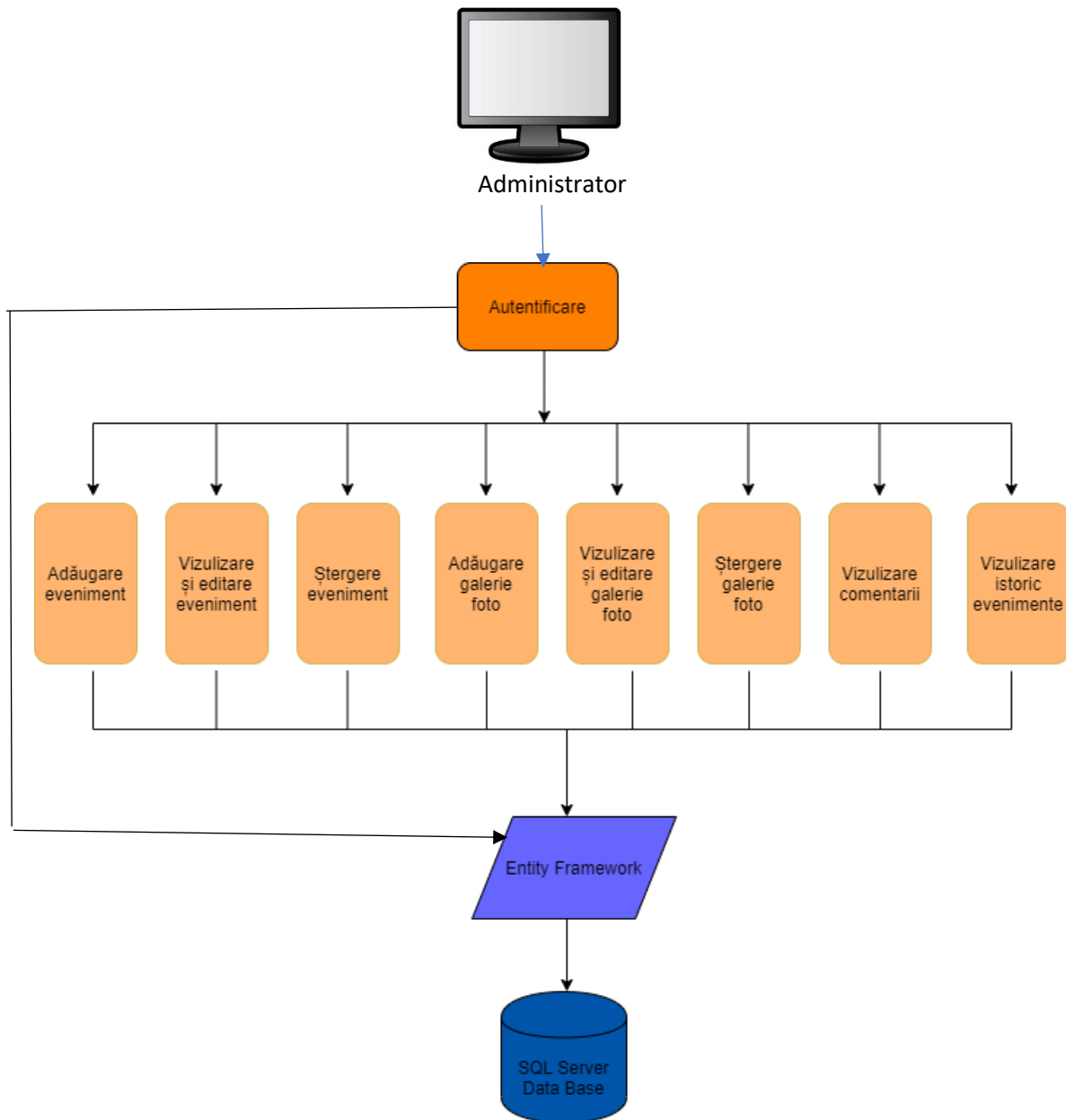


Figura 4.2 – Arhitectura modului Administrator

Utilizatorul autentificat se loghează cu datele aferente lui sau dacă este un utilizator fără cont, are posibilitatea de a își crea cont și apoi să se logheze cu datele furnizate. După această etapă, utilizatorul poate vizualiza toate evenimentele pe categorii, poate căuta un anumit eveniment după denumire, iar pentru fiecare eveniment în parte, acesta poate vedea și detaliile aferente. Totodata, acesta are acces la câte o galerie foto a fiecărui eveniment în parte, dar și la secțiunea de comentarii a evenimentelor, pentru a putea vedea impresiile celorlalți utilizatori care au mai fost în trecut la aceste evenimente, și poate lăsa și el comentarii, la rândul lui, la evenimentele la care a participat, dar și la cele la care urmează să participe. Utilizatorul poate

primi pe e-mail-ul personal informațiile referitoare la evenimentele alese de el, și, totodată, poate trimite mesaje în cadrul aplicației către ceilalți utilizatori activi în acel moment. În același timp, poate vizualiza și o listă cu evenimentele care s-au desfășurat deja, împreună cu comentariile aferente lăsate de cei care au participat (figura 4.3).

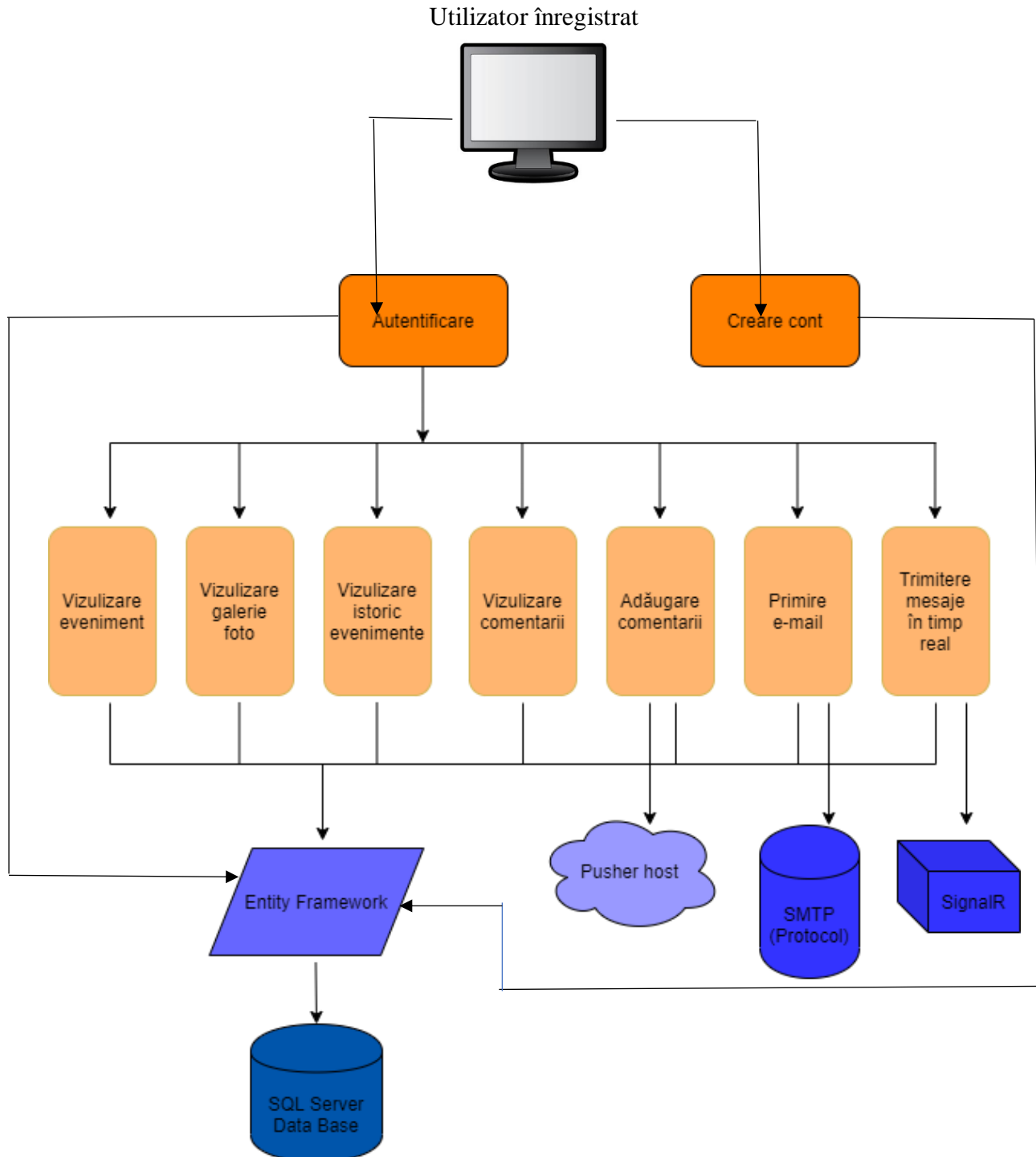


Figura 4.3 – Arhitectura modului Utilizator autentificat

Utilizatorii neautentificați pot vizualiza toate evenimentele pe categorii, pot căuta un anumit eveniment după denumire, iar pentru fiecare eveniment în parte, aceștia pot vedea și detaliile aferente acestor evenimente. Totodata, aceștia au acces la secțiunea de comentarii ale evenimentelor, pentru a putea vedea impresiile celorlalți utilizatori care au mai fost în trecut la aceste evenimente, și pot vizualiza o galerie foto în cadrul fiecărui eveniment, pentru o mai bună informare în legătură cu evenimentul interesat. În același timp, pot vizualiza și o listă cu evenimentele care s-au desfășurat deja, împreună cu comentariile aferente lăsate de cei care au participat (figura 4.4).

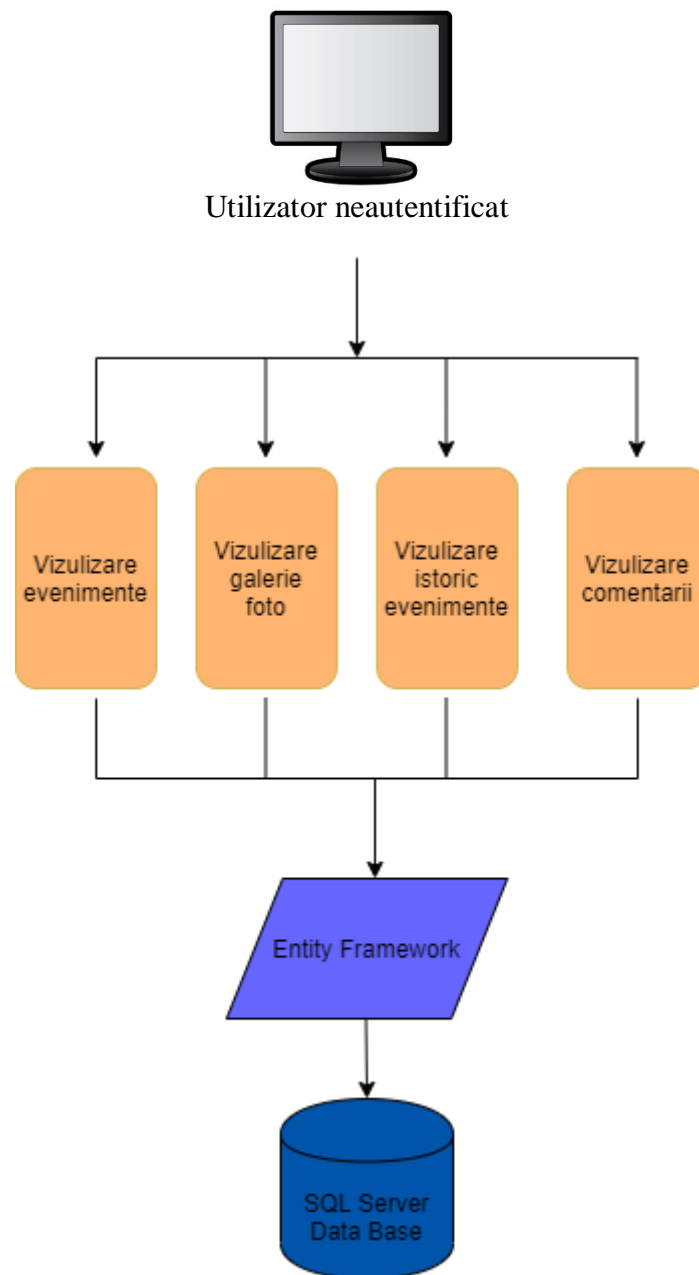


Figura 4.4 – Arhitectura modului Utilizator neautentificat

4.3 Descrierea modulelor

Modulul Administrator

➤ Autentificare

Autentificarea în aplicație a administratorului se face printr-o cerere de tip HTTP POST utilizând tehnologia Entity Framework. Prin autentificare, se va face o cerere de tip POST către user-ul și parola aferente administratorului, stocate într-o bază de date securizată, iar în cazul în care datele introduse sunt aceleași cu cele aflate în baza de date, se va iniția o sesiune activă pentru user-ul care s-a logat în momentul respectiv, administratorul în cazul de față, afișându-se printr-o acțiune de tip GET view-ul returnat în urma acestei acțiuni și atribuindu-se drepturile de editare corespunzătoare cu tipul utilizatorului autentificat. În caz contrar, se va semnaliza un mesaj de eroare.

```
[HttpPost]
[ValidateAntiForgeryToken]
0 references
public ActionResult Index(User users)
{
    if (ModelState.IsValid)
    {
        using (LicentalContext db = new LicentalContext())
        {
            var obj = db.Useri.Where(u => u.Email.Equals(users.Email) && u.Password.Equals(users.Password)).FirstOrDefault();
            if (obj != null)
            {
                Session["Id"] = obj.Id.ToString();
                Session["Email"] = obj.Email.ToString();
                Session["Password"] = obj.Password.ToString();
                return RedirectToAction("Index", "Home");
            }
            else
            {
                ViewBag.Message = "Email or password invalid!";
            }
        }
    }
    return View(users);
}
```

➤ Adăugare eveniment

Adăugarea unui nou eveniment se realizează făcând o cerere de tip HTTP POST cu un model ce are în componența lui datele aferente evenimentului respectiv, către baza de date, în care se înregistrează noul eveniment adăugat de administrator, după validarea câmpurilor completate. În același timp se realizează și încărcarea căii în baza de date, aferente fiecărei fotografii în parte din galeria foto, folosindu-ne de clasa abstractă HttpPostedFileBase care asigură acces individual către fiecare fișier încărcat de un utilizator. Salvarea se realizează folosind funcțiile specifice tehnologiei Entity Framework, care fac referire la Context-ul creat, aferent bazei de date.

```
[HttpPost]
[ValidateAntiForgeryToken]
0 references
public ActionResult Create([Bind(Include = "EvenimentId,NumeEvent,DataEvent,DescriereEvent,PozeEvent,GenEvent")] Eveniment eveniment, HttpPostedFileBase file)
{
    if (ModelState.IsValid)
    {
        var fileName = Path.GetFileName(file.FileName);
        var path = Path.Combine(Server.MapPath("~/images/"), fileName);
        file.SaveAs(path);
        eveniment.PozeEvent = Url.Content("~/images/" + fileName);
        db.Evenimente.Add(eventiment);
        db.SaveChanges();
        return RedirectToAction("Index");
    }
    return View(eventiment);
}
```

➤ Vizualizare și editare eveniment

Pentru a vizualiza un eveniment sau o lista de evenimente din cadrul unui gen, se folosește metoda HTTP GET care extrage din baza de date informațiile referitoare la evenimentul sau evenimentele alese. Pentru a putea edita evenimentele, se folosește o cerere de tip HTTP POST în cadrul căreia se modifică datele aferente evenimentului editat, iar după ce se salvează modificările se apelează la o metodă GET pentru a afișa evenimentul modificat împreună cu lista din care face parte.

```
// GET: Eveniments
0 references
public ActionResult Index(string linktext)
{
    if (linktext == "Festival")
        return View(db.Evenimente.Where(m => m.GenEvent == "Festival" && m.DataEvent >= DateTime.Now).ToList());
    if (linktext == "Concert live")
        return View(db.Evenimente.Where(m => m.GenEvent == "Concert live" && m.DataEvent >= DateTime.Now).ToList());
    if (linktext == "Teatru")
        return View(db.Evenimente.Where(m => m.GenEvent == "Teatru" && m.DataEvent >= DateTime.Now).ToList());
    else return View(db.Evenimente.Where(m => m.DataEvent >= DateTime.Now));
}
```

```
[HttpPost]
[ValidateAntiForgeryToken]
0 references
public ActionResult Edit([Bind(Include = "EvenimentId,NumeEvent,DataEvent,DescriereEvent,PozeEvent,GenEvent")] Eveniment eveniment, HttpPostedFileBase file)
{
    if (ModelState.IsValid)
    {
        if (file != null)
        {
            var fileName = Path.GetFileName(file.FileName);
            db.Evenimente.Add(eventiment);
            var path = Path.Combine(Server.MapPath("~/images/"), fileName);
            file.SaveAs(path);
            eveniment.PozeEvent = Url.Content("~/images/" + fileName);
        }
        db.Entry(eventiment).State = EntityState.Modified;
        db.SaveChanges();
        return RedirectToAction("Index");
    }
    return View(eventiment);
}
```

➤ Ștergere eveniment

Dacă se dorește ștergere unui anumit eveniment, din diverse motive, aceasta se realizează inițiind o cerere HTTP POST către baza de date, după apăsarea butonului de ștergere. Este

trimis către baza de date un model cu informațiile evenimentului care se dorește a fi șters și se caută în baza de date în funcție de Id-ul evenimentului. Din nou, se utilizează funcții specifice Entity Framework pentru lucrul cu baza de date.

```
// POST: Eveniments/Delete/5
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
0 references
public ActionResult DeleteConfirmed(int id)
{
    Eveniment eveniment = db.Evenimente.Find(id);
    db.Evenimente.Remove(eventiment);
    db.SaveChanges();
    return RedirectToAction("Index");
}
```

➤ Adăugare galerie foto

Pentru a adăuga o galerie foto fiecărui eveniment, se extrage numele fotografiei și calea din calculatorul de pe care se încarcă fotografia, iar acestea se salvează într-o variabilă de tip `HttpPostFileBase` fiind o clasă abstractă care asigură acces individual către fiecare fișier încărcat de un utilizator, iar datele din aceste variabile se salvează în baza de date printr-o cerere de tipul `Http Post`, folosind metoda `db.SaveChanges()` specifică Entity Framework.

```
[HttpPost]
[ValidateAntiForgeryToken]
0 references
public ActionResult Create([Bind(Include = "PozeId,Poza1,Poza2,Poza3,Poza4")] Poze poze, HttpPostedFileBase file1, HttpPostedFileBase file2, HttpPostedFileBase file3, HttpPostedFileBase file4)
{
    if (ModelState.IsValid)
    {
        var fileName = Path.GetFileName(file1.FileName);
        var path = Path.Combine(Server.MapPath("~/images/"), fileName);
        file1.SaveAs(path);
        poze.Poza1 = Url.Content("~/images/" + fileName);

        var fileName2 = Path.GetFileName(file2.FileName);
        var path2 = Path.Combine(Server.MapPath("~/images/"), fileName2);
        file2.SaveAs(path2);
        poze.Poza2 = Url.Content("~/images/" + fileName2);

        var fileName3 = Path.GetFileName(file3.FileName);
        var path3 = Path.Combine(Server.MapPath("~/images/"), fileName3);
        file3.SaveAs(path3);
        poze.Poza3 = Url.Content("~/images/" + fileName3);

        var fileName4 = Path.GetFileName(file4.FileName);
        var path4 = Path.Combine(Server.MapPath("~/images/"), fileName4);
        file4.SaveAs(path4);
        poze.Poza4 = Url.Content("~/images/" + fileName4);

        db.Foto.Add(poze);
        db.SaveChanges();

        return RedirectToAction("Index");
    }
    ViewBag.PozeId = new SelectList(db.Evenimente, "EvenimentId", "NumeEvent", poze.PozeId);
    return View(poze);
}
```

➤ Vizualizarea și editarea galeriei foto

Editarea galeriei foto se realizează în aceeași manieră cu cea a creării galeriei foto, a încărcării pozelor în galerie, metodă care a fost prezentată mai sus. Vizualizarea galeriei se realizează printr-o cerere de tipul `Http Get`, în care se extrag datele din baza de date pentru

vizualizare, iar în același timp se face o legătură de unul la unul, asociindu-i-se fiecărui eveniment câte o galerie foto. Pentru afișarea detaliată a galeriei se identifică Id-ul galeriei care corespunde cu Id-ul evenimentului și se afișează prin aceeași cerere Http Get.

```
// GET: Pozes
0 references
public ActionResult Index()
{
    var foto = db.Foto.Include(p => p.Eventiment);
    return View(foto.ToList());
}

// GET: Pozes/Details/5
0 references
public ActionResult Details(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    Poze poze = db.Foto.Find(id);
    if (poze == null)
    {
        return HttpNotFound();
    }
    return View(poze);
}
```

➤ Ștergere galerie foto

În cazul ștergerii unei galerii foto, procedeul de realizare este exact același cu cel al ștergerii unui eveniment, procedeu prezentat mai sus în document.

➤ Vizualizare comentarii

Pentru vizualizarea comentariilor oricărei postări, se realizează o cerere de tip Http Get care returnează datele de tip JSON ale tuturor comentariilor pentru o anumită postare. Metoda va fi apelată printr-un Ajax Post.

```
public ActionResult Comments(int? id)
{
    var comments = db.Comments.Where(x => x.EventimentId == id).ToArray();
    return Json(comments, JsonRequestBehavior.AllowGet);
}
```

➤ Vizualizare istoric evenimente

Vizualizarea istoricului evenimentelor se realizează în același mod cu afișarea evenimentelor, se verifică în plus doar dacă data de început a evenimentului este mai mică decât data curentă.

Modulul Utilizator autentificat**➤ Creare cont**

Prima acțiune pe care un utilizator o face când ajunge în aplicație și dorește să își facă un cont, este să meargă la secțiunea de Creare cont unde își crează un cont personal, furnizând datele lui proprii. După ce utilizatorul a completat toate câmpurile aferente cu date valide, se inițiază o cerere de tip Http Post cu datele completate către baze de date, iar aceste informații se salvează în baza de date folosind metoda `db.SaveChanges()` specifică Entity Framework.

```
[HttpPost]
[ValidateAntiForgeryToken]
0 references
public ActionResult Register([Bind(Include = "UserId, Nume, Email, Password")] User user)
{
    if (ModelState.IsValid)
    {
        db.Useri.Add(user);
        db.SaveChanges();
        return RedirectToAction("Index", "Home");
    }

    return View(user);
}
```

➤ Autentificare

După ce și-a făcut contul, acesta trebuie să se autentifice folosind informațiile furnizate la crearea contului. După ce utilizatorul a completat datele necesare cu informații valide, se inițiază o cerere Http Post pentru a face o interogare asupra tabelului și a verifica dacă datele introduse coincid cu cele din baza de date ale utilizatorului respectiv, acest lucru verificându-se printr-o expresie de tip LINQ. Dacă datele introduse sunt valide, se generează o sesiune pentru utilizatorul respectiv, în care se memorează în niște variabile de sesiune datele aferente utilizatorului respectiv. În caz contrar, se afișează un mesaj de eroare sub formularul de autentificare.

```
[HttpPost]
[ValidateAntiForgeryToken]
0 references
public ActionResult Index(User users)
{
    if (ModelState.IsValid)
    {
        using (LicentalContext db = new LicentalContext())
        {
            var obj = db.Useri.Where(u => u.Email.Equals(users.Email) && u.Password.Equals(users.Password)).FirstOrDefault();
            if (obj != null)
            {
                Session["Id"] = obj.Id.ToString();
                Session["Email"] = obj.Email.ToString();
                Session["Password"] = obj.Password.ToString();
                return RedirectToAction("Index", "Home");
            }
            else
            {
                ViewBag.Message = "Email or password invalid!";
            }
        }
    }
    return View(users);
}
```

➤ Vizualizare evenimente

În această etapă, utilizatorul navighează către ce categorie de evenimente dorește să vadă, iar după ce și a ales evenimentul, poate vedea detalii aferente evenimentului. Aceste lucruri se realizează creând o cerere de tip Http Get în care extragem din baza de date informațiile referitoare la toate evenimentele din categoria aleasă, acest lucru făcându-se prin folosirea unor sintaxe LINQ pentru determinarea categoriei alese, iar, apoi, informațiile sunt afișate pe ecran. După ce s-a ales evenimentul și se dorește vizualizarea detaliilor lui, se procedează într-o manieră similară cu cea prezentată mai sus.

```
0 references
public ActionResult IndexUser(string linktext, string searchString)
{
    if (linktext == "Festival")
        return View(db.Eventimente.Where(m => m.GenEvent == "Festival" && m.DataEvent >= DateTime.Now).ToList());
    if (linktext == "Concert live")
        return View(db.Eventimente.Where(m => m.GenEvent == "Concert live" && m.DataEvent >= DateTime.Now).ToList());
    if (linktext == "Teatru")
        return View(db.Eventimente.Where(m => m.GenEvent == "Teatru" && m.DataEvent >= DateTime.Now).ToList());
    else return View(db.Eventimente.Where(m => m.DataEvent >= DateTime.Now));
}
```

➤ Vizualizare istoric evenimente

Pentru vizualizarea istoricului, se procedează în același fel ca și vizualizarea evenimentelor care urmează, doar că se mai utilizează o sintaxă de LINQ, în plus, pentru a afișa evenimentele cu data de desfășurare mai mică decât data curentă.

```
0 references
public ActionResult IndexIstoric(string linktext)
{
    return View(db.Eventimente.Where(m => m.DataEvent < DateTime.Now));
}
```


➤ Vizualizare galerie foto pentru fiecare eveniment

În acest cadru, utilizatorul poate vedea câte o galerie foto pentru fiecare eveniment în parte. Această specificație este aceeași ca funcția de vizualizare evenimente a administratorului, iar acest lucru a fost prezentat acolo.

➤ Vizualizare comentarii

Vizualizarea comentariilor postate de către alți utilizatori este la fel cu cea generată pentru vizualizarea de către administrator, iar acest lucru a fost explicat în paragraful despre care fac referire.

➤ Vizualizare și adăugare comentarii

Partea de vizualizare comentarii este aceeași cu vizualizarea comentariilor de către administrator, fiind explicată în acea secțiune. Pentru a se adăuga comentarii la fiecare eveniment în parte, se utilizează o metodă asincronă de trimitere care se ocupă de adăugarea comentariilor în baza de date și trimiterea datelor către Pusher. Folosim o metodă asincronă deoarece librăria Pusher trimite datele asincron și este nevoie de așteptarea răspunsului ei.

```
[HttpPost]
0 references
public async Task<ActionResult> Comment(Comment data)
{
    db.Comments.Add(data);
    db.SaveChanges();
    var options = new PusherOptions();
    options.Cluster = "eu";
    var pusher = new Pusher("1006665", "0c257db48809018c5fd3", "41ce8ec4dd6e7d560533", options);
    ITriggerResult result = await pusher.TriggerAsync("asp_channel", "asp_event", data);
    return Content("ok");
}
```

Scriptul de pe partea de client pentru vizualizare și adăugare comentarii:

```
<script>
var pusher = new Pusher('0c257db48809018c5fd3', {
  cluster: 'eu'
});
var my_channel = pusher.subscribe('my-channel');
var app = new Vue({
  el: '#app',
  data: {
    comments: [],
    comment: {
      Name: '',
      Body: '',
      EventimentId: @Model.EventimentId
    }
  },
  created: function() {
    this.get_comments();
    this.listen();
  },
  methods: {
    get_comments: function() {
      axios.get('@Url.Action("Comments", "Eveniments", new { id = @Model.EventimentId }, protocol: Request.Url.Scheme)')
        .then((response) => {
          this.comments = response.data;
        });
    },
    listen: function() {
      my_channel.bind("my-event", (data) => {
        if (data.EventimentId == this.comment.EventimentId) {
          this.comments.push(data);
        }
      })
    },
    submit_comment: function() {
      axios.post('@Url.Action("Comment", "Eveniments", new {}, protocol: Request.Url.Scheme)', this.comment)
        .then((response) => {
          this.comment.Name = '';
          this.comment.Body = '';
          alert("Comentariu postat");
        });
    }
  }
});
</script>
```

➤ Trimitere e-mail

În cazul în care utilizatorul găsește interesant un anumit eveniment, acesta poate accesa butonul de trimitere e-mail, primind astfel pe e-mail informații despre evenimentul selectat. Pentru trimiterea e-mail-ului se folosește protocolul SMTP (Simple Mail Transfer Protocol) în cadrul căruia se specifică serverul gazda SMTP utilizat pentru trimiterea e-mail-ului, certificatul de autenticitate dacă este cerut de serverul SMTP, adresa de e-mail a expeditorului (proprietatea MailMessae.From), adresa de e-mail a destinatarului (proprietatea MaiMessage.To), subiectul mesajului (proprietatea MailMessage.Subject) și conținutul mesajului (proprietatea MailMessage.Body)

```
[HttpPost]
[ValidateAntiForgeryToken]
0 references
public ActionResult SendEmail(int id)
{
    Eveniment eveniment = db.Evenimente.Find(id);
    if (ModelState.IsValid)
    {
        var message = new MailMessage();
        message.To.Add(new MailAddress(Session["Email"].ToString())); // replace with valid value
        message.From = new MailAddress("popescu.sorin23@gmail.com"); // replace with valid value
        message.Subject = "Licenta Web App";
        message.Body = "Hello, ai ales sa mergi la evenimentul "+eveniment.NumeEvent + " care se desfasoara in data de "+ eveniment.DataEvent+". Descriere: " + eveniment.DescriereEvent;
        message.IsBodyHtml = true;

        using (var smtp = new SmtpClient())
        {
            var credential = new NetworkCredential
            {
                UserName = "popescu.sorin23@gmail.com", // replace with valid value
                Password = "****" // replace with valid value
            };
            smtp.UseDefaultCredentials = false;
            smtp.Credentials = credential;
            smtp.Host = "smtp.gmail.com";
            smtp.Port = 587;
            smtp.EnableSsl = true;
            smtp.Send(message);
        }

        return RedirectToAction("IndexUser");
    }
    return View();
}
```

➤ Trimitere și vizualizare mesaje în timp real

În cadrul acestei opțiuni, utilizatorul autentificat are posibilitatea de a trimite mesaje publice într-un grup de mesaje. Doar cei care sunt autentificați și se află în secțiunea de mesaje în același timp cu cel care trimite mesajul pot vedea mesajele trimise și pot răspunde la rândul lor. Prin această funcționalitate am folosit clasa ChatHub care derivă din clasa Microsoft.AspNet.SignalR.Hub. Derivarea din clasa Hub este o metoda utilă de a construi aplicații SignalR. În codul de chat, clienții apelează metoda ChatHub.Send pentru a trimite un mesaj nou. Hub-ul la rândul său trimite mesajul tuturor clienților apelând Clients.All.addNewMessageToPage. În fișierul de interfață cu utilizatorul codul realizează mai multe sarcini importante, cum ar fi: creează o referință către proxy-ul autogenerat pentru hub, declara o funcție pe care server-ul o poate apela pentru a împinge conținut către clienți și începe o conexiune pentru a trimite mesaje către hub.

```
0 references
public class ChatHub : Hub
{
    public void Send(string name, string message)
    {
        // Call the addNewMessageToPage method to update clients.
        Clients.All.addNewMessageToPage(name, message);
    }
}
```

Scriptul din interfața cu utilizatorul:

```
<script>
$(function () {
    // Reference the auto-generated proxy for the hub.
    var chat = $.connection.chatHub;
    // Create a function that the hub can call back to display messages.
    chat.client.addNewMessageToPage = function (name, message) {
        // Add the message to the page.
        $('#discussion').append('<li><strong>' + htmlEncode(name)
            + '</strong>: ' + htmlEncode(message) + '</li>');
    };
    // Get the user name and store it to prepend to messages.
    $('#displayname').val(prompt('Enter your name:', ''));
    // Set initial focus to message input box.
    $('#message').focus();
    // Start the connection.
    $.connection.hub.start().done(function () {
        $('#sendmessage').click(function () {
            // Call the Send method on the hub.
            chat.server.send($('#displayname').val(), $('#message').val());
            // Clear text box and reset focus for next comment.
            $('#message').val('').focus();
        });
    });
});
// This optional function html-encodes messages for display in the page.
function htmlEncode(value) {
    var encodedValue = $('<div />').text(value).html();
    return encodedValue;
}
}</script>
```

Modulul Utilizator neautentificat

➤ Vizualizare eveniment

În această opțiune, utilizatorul are posibilitatea de a vedea toate evenimentele în funcție de categoria aleasă și, totodată pot vedea informațiile și galeria foto a fiecărui eveniment în parte. Funcționalitatea acestei opțiuni a fost prezentată mai sus în cadrul ambelor module.

➤ Vizualizare istoric evenimente

Pentru această opțiune, utilizatorul intră în secțiunea de istoric și e capabil să vadă aceleași informații ca la evenimentele prezentate mai sus. Descrierea acestei funcționalități a fost detaliată în modulele anterioare, cele de Administrator și Utilizator autentificat.

➤ Vizualizare galerie foto

Atunci când utilizatorul intră în secțiunea de detalii a oricărui eveniment, are posibilitatea de a ajunge de acolo la galeria foto aferentă evenimentului respectiv. Procedul de afișare și vizualizare a galeriei a fost prezentat mai sus în ambele module, fiind detaliat în modulul Administrator.

➤ Vizualizare comentarii

Asemenea cazului de mai sus, când utilizatorul intră în secțiunea de detalii a unui eveniment, are posibilitatea de a vizualiza și comentariile lăsate de unii utilizatori care au participat la evenimentul respectiv. Implementarea acestei funcționalități a fost prezentată în modulul Administrator.

4.4 Structura bazei de date

Baza de date cuprinde o serie de tabele interconectate între ele, create în SQL Server Management, folosind limbajul SQL.

Tabela Users conține următoarele câmpuri: Id – cheia primară a tabelului, Nume – numele cu care se înregistrează utilizatorul, Email – e-mail-ul cu care se înregistrează și apoi se autentifică utilizatorul și Password – parola pe care o setează, iar apoi se autentifică cu ea utilizatorul. Tabela Eveniments are următoarele câmpuri: Id – cheia primară a tabelului, NumeEvent – denumirea evenimentului, DataEvent – data și ora când începe evenimentul, DescriereEvent – o descriere a evenimentului, precum și locul de desfășurare, dar și alte informații utile, PozeEvent – o fotografie sugestivă ca avatar pentru evenimentul respectiv, GenEvent – genul evenimentului, în funcție de care se stabilește în ce categorie va fi încadrat. Tabela Pozes cuprinde următoarele câmpuri: PozeId – cheia primară a tabelului, fiind în același timp și cheie străină, făcând legătura dintre această tabelă și tabela Eveniments, Poza1, Poza2, Poza3 și Poza4 sunt ultimele 4 câmpuri care conțin fiecare câte o cale spre o fotografie de pe dispozitiv care va fi încărcată de administrator, realizând, astfel, galeria foto. Tabela Comments conține următoarele câmpuri: CommentID – fiind cheia primară a tabelului, Body – conținutul comentariului, Name – numele persoanei care scrie comentariul, EvenimentId – Id-ul evenimentului la care se postează comentariul. Între tabelul Eveniments și Pozes există o relație de unu-la-unu, în sensul că, cheia primară de la tabela Pozes, PozeId, este cheie străină pentru tabelul Eveniments. Mai exact, unei înregistrări din tabela Eveniments îi corespunde o singură

înregistrare din tabela Pozes, adică unui eveniment îi va corespunde o galerie foto. Totodata, există tot o relație de unu-la-unu între tabelele Eveniments și Comments, adică cheia primară din tabelul Eveniments, EvenimentId, este cheie străină în tabelul Comments, mai exact, unui eveniment îi corespunde un set de comentarii.

Structura bazei de date este prezentată mai jos (figura 4.5).

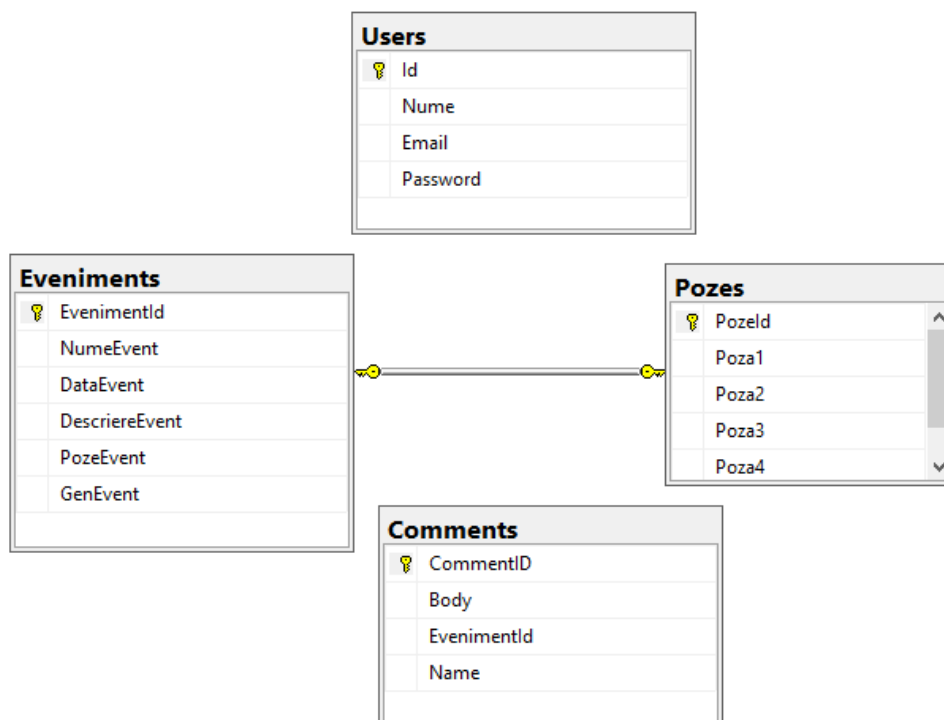


Figura 4.5 – Structura bazei de date

După cum am menționat și mai sus, baza de date conține mai multe tabele. Prima tabela, tabela Users, conține id-ul, de tip integer, care este cheia primară a tablei. Următorul câmp, Nume, de tip nvarchar, reprezintă numele ales de utilizator când își creează un cont. Câmpul Email, tot de tip nvarchar, folosit pentru a stoca adresa de e-mail furnizată de utilizator la crearea contului. Iar ultimul câmp, Password, de tip nvarchar, fiind utilizat pentru stocarea parolei alese de utilizator, cu care se va autentifica în partea de login. Structura tablei este prezentată mai jos (figura 4.6).


Users			
	Column Name	Data Type	Allow Nulls
	Id	int	<input type="checkbox"/>
	Nume	nvarchar(MAX)	<input checked="" type="checkbox"/>
	Email	nvarchar(MAX)	<input checked="" type="checkbox"/>
	Password	nvarchar(MAX)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Figura 4.6 – Structura tabelii Users

În cadrul tabelii Eveniments, s-au folosit mai multe câmpuri, după cum s-a prezentat mai sus, și anume, câmpul EvenimentId, de tip integer, folosit ca și cheie primară a tabelii. NumeEvent de tip nvarchar, în cadrul căruia se salvează denumirea fiecărui eveniment, apoi, câmpul DataEvent, de tip datetime, în el se înregistrează data și ora la care va începe fiecare eveniment. Câmpul Descriere, de tip nvarchar, va cuprinde informații adiționale legate de evenimente, cum ar fi o mică descriere a lor, însoțită de locul desfășurării, dar și alte informații relevante. Următorul câmp, PozeEvent, de tip nvarchar, reprezentând fotografia cu care se identifică evenimentul, în el se va salva calea din computerul de pe care administratorul va încărca fotografia, iar ultimul câmp, câmpul GenEvent, tot de tip nvarchar, în el salvându-se genul evenimentului creat, în funcție de el, repartizându-se evenimentul în categoria specifică. Structura tabelii Eveniments este redată mai jos (figura 4.7).


Eveniments *			
	Column Name	Data Type	Allow Nulls
	EvenimentId	int	<input type="checkbox"/>
	NumeEvent	nvarchar(MAX)	<input type="checkbox"/>
	DataEvent	datetime	<input type="checkbox"/>
	DescriereEvent	nvarchar(MAX)	<input type="checkbox"/>
	PozeEvent	nvarchar(MAX)	<input type="checkbox"/>
	GenEvent	nvarchar(MAX)	<input type="checkbox"/>
			<input type="checkbox"/>

Figura 4.7 – Structura tabelii Eveniments

Pentru tabela Pozes, cum s-a menționat și mai sus, s-a folosit câmpul PozelId, de tip integer, reprezentând, pe de o parte, cheia primară a tabelului, iar pe de altă parte, în același timp, cheia străină din tabela Eveniments în tabela Pozes. În acest caz, s-a folosit o relaționare unu-la-unu între tabela Eveniments și tabela Pozes, în sensul că, unui eveniment îi corespunde un singur set de poze, adică o galerie. Câmpurile Poza1, Poza2, Poza3, Poza4 sunt toate de același tip, adică nvarchar, iar în ele se salvează câte o cale spre o fotografie din calculatorul de pe care administratorul încarcă fotografiile în galeria foto. Structura tabelului Pozes este prezentată mai jos (figura 4.8).


Pozes			
	Column Name	Data Type	Allow Nulls
	PozelId	int	<input type="checkbox"/>
	Poza1	nvarchar(MAX)	<input checked="" type="checkbox"/>
	Poza2	nvarchar(MAX)	<input checked="" type="checkbox"/>
	Poza3	nvarchar(MAX)	<input checked="" type="checkbox"/>
	Poza4	nvarchar(MAX)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Figura 4.8 – Structura tabelului Pozes

În cadrul tabelului Comments, s-a utilizat câmpul CommentID, de tip integer, fiind cheia primară a tabelului. Câmpul Body, de tip ntext, a fost utilizat pentru a se stoca fiecare comentariu în parte, lăsat de utilizatori. Următorul, câmpul Name, de tip nvarchar, a fost folosit pentru a se salva în el numele utilizatorilor care au lăsat comentarii la evenimente, iar câmpul EvenimentId, de tip integer, reprezintă cheia străină din tabelul Eveniments. Aceasta realizează o relaționare de unu-la-unu cu tabela Eveniments, adică, mai exact, unui eveniment i se atribuie un singur set de comentarii. Structura tabelului Comments se poate vedea mai jos (figura 4.9).


Comments			
	Column Name	Data Type	Allow Nulls
	CommentID	int	<input type="checkbox"/>
	Body	ntext	<input checked="" type="checkbox"/>
	EvenimentId	int	<input type="checkbox"/>
	Name	nvarchar(MAX)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Figura 4.9 – Structura tabelului Comments

CAPITOLUL 5

UTILIZAREA APLICAȚIEI

La accesarea aplicației, în primă fază, utilizatorilor, indiferent de tipul lor, le este afișată pagina principală de mai jos, figura 5.1:

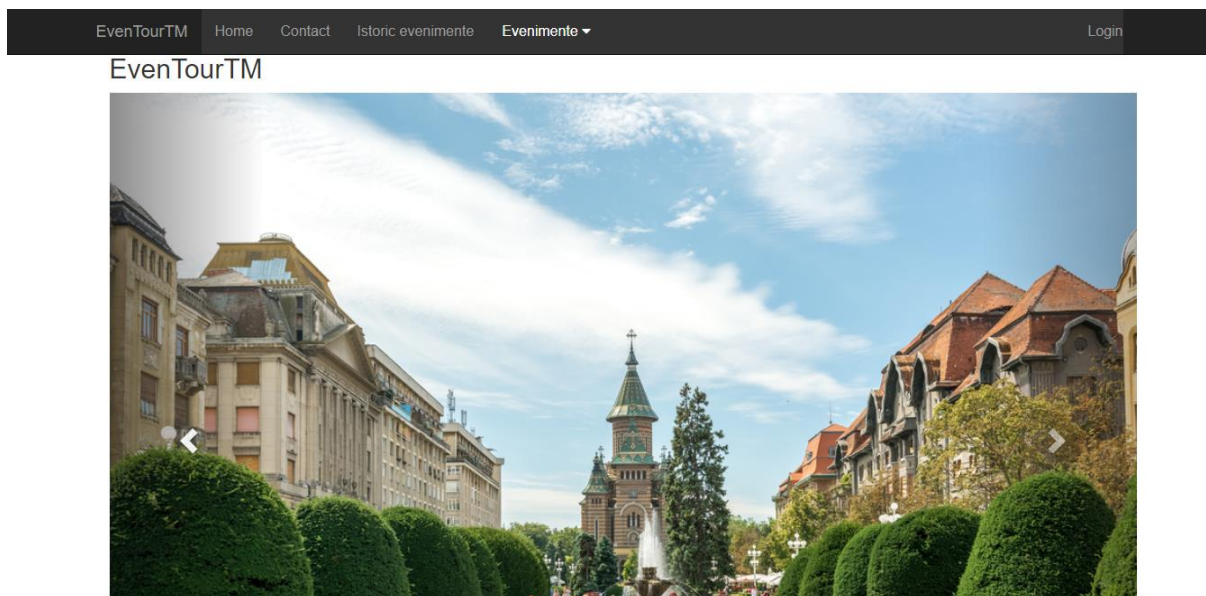


Figura 5.1 – Pagina principală a aplicației

De pe pagina principală, utilizatorul poate naviga către pagina de login a aplicației, pagina care arată la fel pentru toate tipurile de utilizatori și este redată mai jos, figura 5.2.

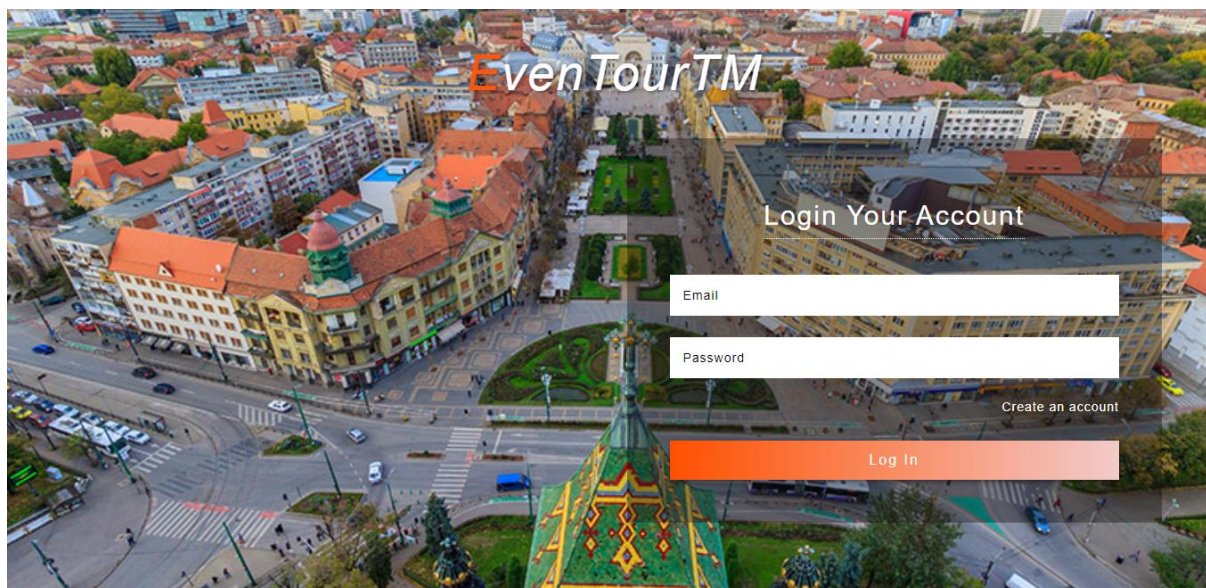


Figura 5.2 – Pagina de Login a aplicației

În continuare vom vorbi despre modul de utilizare al aplicației în cadrul fiecărui tip de utilizator.

Modulul Utilizator autentificat

În cadrul acestui modul, utilizatorii cu un cont în cadrul aplicației au parte de o serie de funcționalități în plus față de cei neautentificați.

După ce au ajuns în cadrul paginii de login prezentată mai sus, aceștia se autentifică cu datele lor personale, iar dacă nu au cont, își pot crea unul, accesând link-ul de creare cont care îi va direcționa către pagina de înregistrare (figura 5.3).

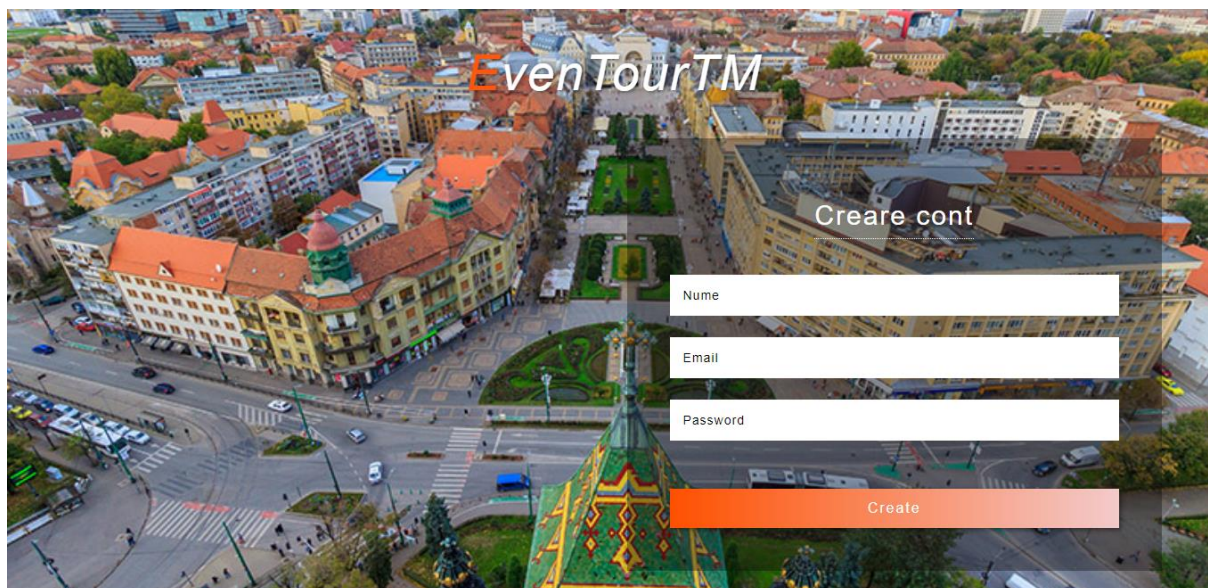


Figura 5.3 – Pagina de Creare cont a aplicației

După ce aceștia își vor crea contul propriu, vor trebui să se autentifice pe pagina de Login, prezentată în figura 5.2, cu datele furnizate la crearea contului. Dacă autentificarea se realizează cu succes, aceștia vor fi redirecționați către pagina principală aferentă utilizatorilor autentificați, unde pe aceasta va apărea, în plus, un tab denumit Chat care va permite schimbul de mesaje în timp real cu alți utilizatori autentificați activi pe Chat (figura 5.4).

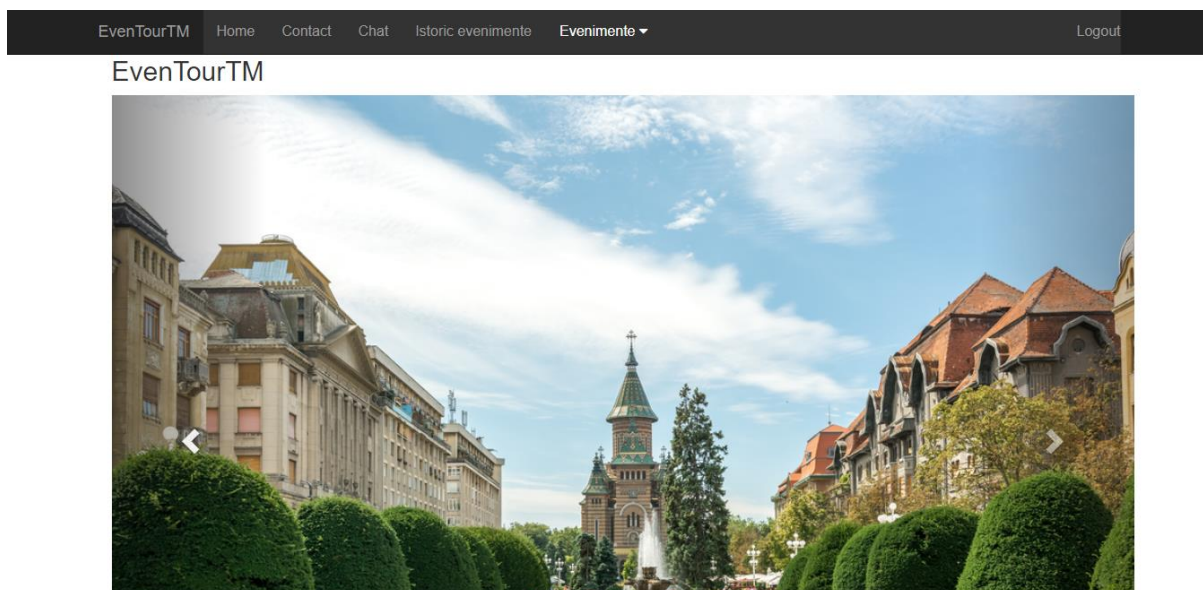


Figura 5.4 – Pagina principală a aplicației pentru cei autentificați

În cadrul paginii principale, utilizatorii au posibilitatea de a alege ce categorie de evenimente doresc să vadă, iar în cadrul fiecăreia se afișează evenimentele aferente (figura 5.5).

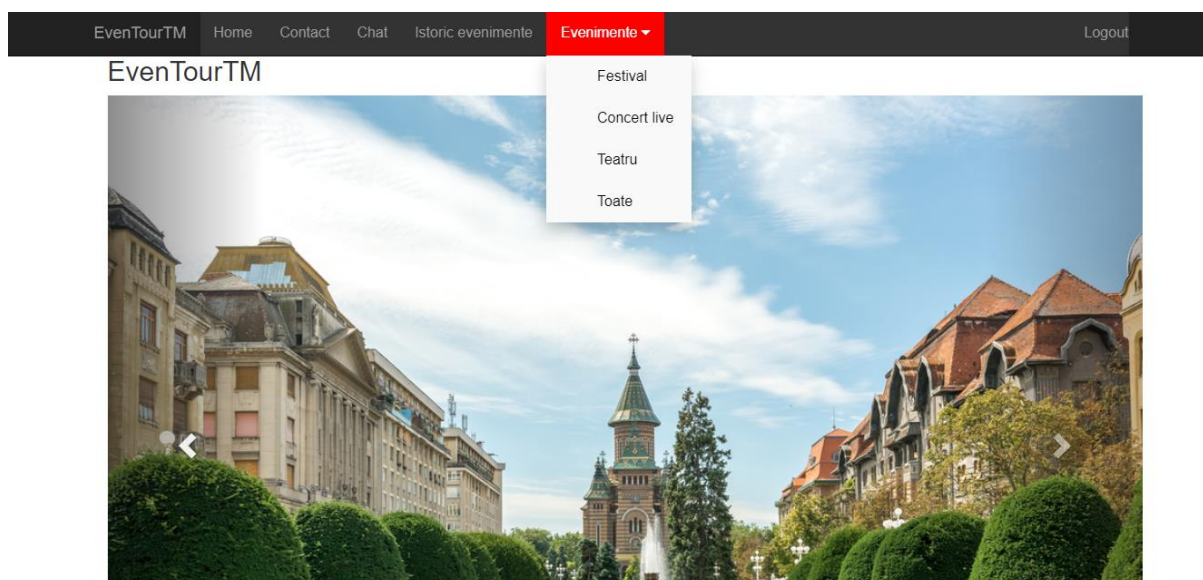


Figura 5.5 – Lista categoriilor disponibile

La selectarea unei categorii, aceștia vor fi direcționați pe pagina care cuprinde toate evenimentele din categoria respectivă. Pe acea pagină va fi afișată o bară de cautare a evenimentelor după denumire, vor fi afișate informațiile de bază ale evenimentelor respective, plus un link în care sunt prezentate mai multe detalii despre fiecare eveniment, dar și un buton pe care îl pot selecta pentru a putea fi înștiințați pe e-mail în legătura cu evenimentul ales (figura 5.6).

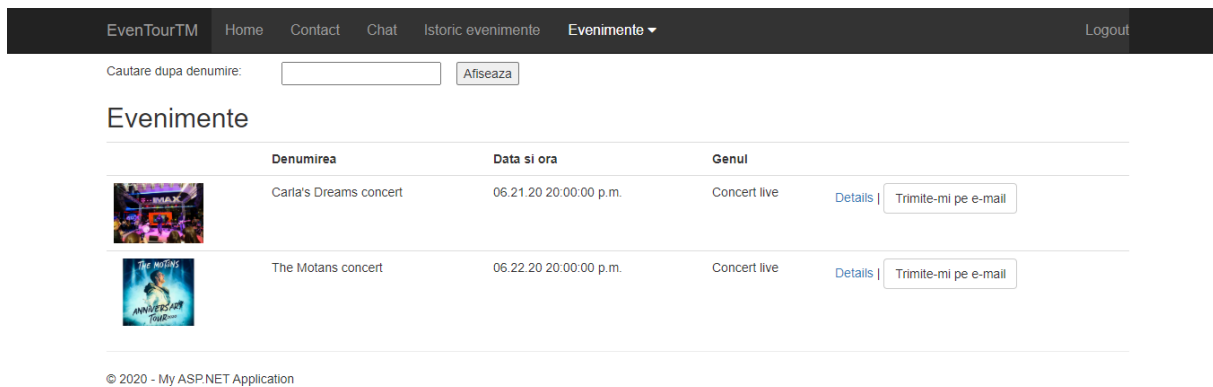


Figura 5.6 – Lista evenimentelor dintr-o anumită categorie

La vizualizarea detaliilor legate de un eveniment, se va deschide o pagină unde vor fi afișate informații suplimentare, plus o secțiune de comentarii în care pot vizualiza comentariile existente, lăsate de alți utilizatori, dar pot adăuga și ei, la rândul lor, comentarii (figura 5.7).



Figura 5.7 – Pagina de detalii a unui eveniment

După ce au ajuns pe această pagină și vor selecta link-ul de „Galerie foto”, utilizatorii vor fi redirecționați către galeria foto a evenimentului respectiv, o serie de imagini derulându-se într-un slideshow (figura 5.8).

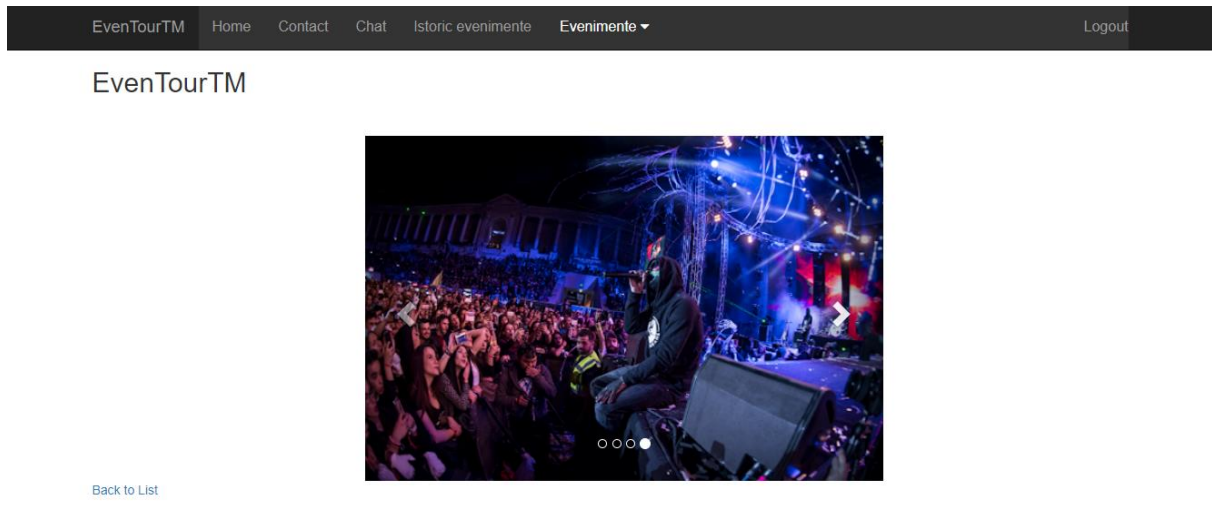


Figura 5.8 – Galeria foto a unui eveniment

În cazul în care utilizatorii vor selecta tab-ul de „Istoric evenimente”, vor fi direcționați către o pagină cu toate evenimentele care s-au desfășurat între timp. Totodată, dacă accesează link-ul de detalii, vor avea acces la alte informații suplimentare despre fiecare eveniment în parte, la secțiunea de comentarii unde pot vizualiza și adăuga și ei comentarii la rândul lor, dar și la galeria foto a evenimentelor, exact ca la secțiunea evenimentelor care se vor desfășura în viitor (figura 5.9).



Evenimente				
	Denumirea	Data si ora	Genul	
	Timforalls	06.04.20 10:00:00 a.m.	Festival	Details
	Bani din cer - piesa de teatru	05.29.20 20:00:00 p.m.	Teatru	Details

Figura 5.9 – Lista evenimentelor care s-au desfășurat deja

Pe lângă acestea, utilizatorii mai au la dispoziție funcția de mesagerie de grup în timp real, adică după ce selectează tab-ul „Chat”, li se va deschide o fereastră în care pot schimba mesaje cu utilizatorii care sunt activi în momentul acela și folosesc și ei acea funcție (figura 5.10).

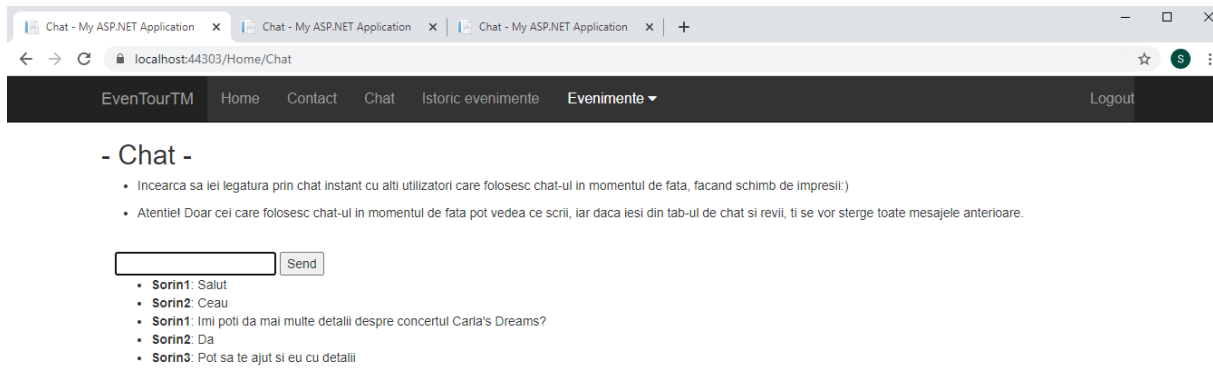


Figura 5.10 – Interfața de mesagerie a aplicației/forum

Modulul Utilizator neautentificat

Spre deosebire de utilizatorii autentificați, cei neautentificați nu mai ajung pe pagina de login sau pe cea de creare cont, ci navighează pe pagina principală, în primă fază (figura 5.11).

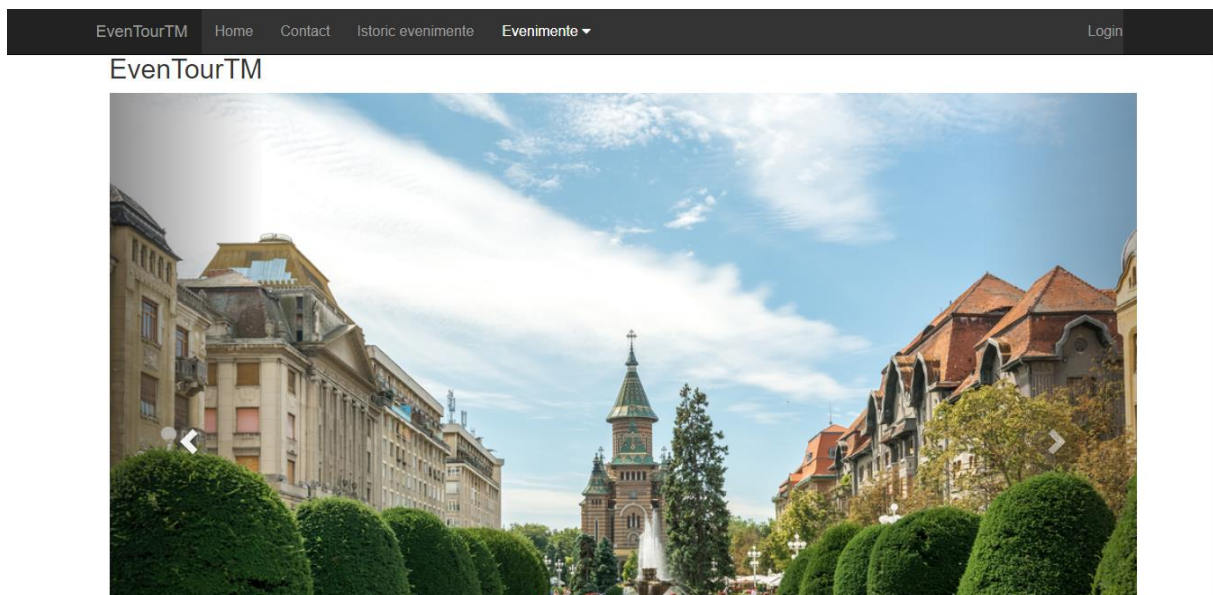


Figura 5.11 – Pagina principală pentru cei neautentificați

În cazul în care utilizatorul selectează o anumită categorie din cele enumerate, i se va afișa o listă cu evenimentele disponibile și o casetă în care le va putea căuta după denumirea lor, în schimb, față de interfața utilizatorului autentificat, el nu va mai avea acces la butonul de e-mail (figura 5.12).

EvenTourTM

Home

Contact

Istoric evenimente

Evenimente

Login

Cautare dupa denumire:

Afiseaza

Evenimente



	Denumirea	Data si ora	Genul	
	Carla's Dreams concert	06.21.20 20:00:00 p.m.	Concert live	Details
	The Motans concert	06.22.20 20:00:00 p.m.	Concert live	Details

Figura 5.12 – Lista evenimentelor pentru cei neautentificați

La selectarea link-ului de detalii, acestora li se va deschide o pagină cu informații suplimentare despre fiecare eveniment, dar și secțiunea de comentarii pe care o pot vizualiza, însă, față de utilizatorii autentificați, nu vor putea adăuga și ei comentarii la evenimente. Iar dacă vor accesa link-ul de galerie foto, vor putea vedea aceeași galerie pe care o văd și utilizatorii autentificați, interfață prezentată mai sus (figura 5.13).


EvenTourTM Home Contact Istoric evenimente Evenimente ▾ Login				
 <p>Galerie foto</p>				
Denumirea	Carla's Dreams concert			
Data si ora	06.21.20 20:00:00 p.m.			
Descriere	<p>Primul turneu național Carla's Dreams are loc în această vară. Trupa va susține un concert la Sala Olimpia din Timișoara pe 21 iunie 2020, de la ora 20:00. În fiecare an, Carla's Dreams susține un concert în fața a peste 5000 de fani la Arenele Romane și umple toate sălile pe unde cântă în țară, și afară. Trupa a acumulat peste 1 milion de abonați pe propriul canal de YouTube unde are peste 530 000 000 de milioane de vizualizări.</p>			
Genul	Concert live			
<h3>Comentarii</h3> <p>1 comments</p> <hr/> <p>SorinPopescu Foarte bine organizat concertul!</p> <hr/> <p>Back to List</p>				

Figura 5.13 – Pagina de Detalii eveniment pentru cei neautentificați

Secțiunea de Istoric evenimente este în același stil cu cea de la modulul utilizatorului autentificat, interfață prezentată în acel modul, și se pot vizualiza aceleași informații ca la evenimentele care se vor desfășura în viitor.

Modulul Administrator

La fel ca și în cadrul modului Utilizator autentificat, administratorul, după ce ajunge pe pagina principală a aplicației, intră pe pagina de autentificare, unde se autentifică cu un user și o parolă primite în prealabil, iar după aceea, este redirecționat către noua pagină principală dedicată lui, pagină ilustrată mai jos. În cadrul acestei pagini, administratorul are în bara de navigare acces atât la ce pot vizualiza utilizatorii cu cont, cât și la lista cu utilizatorii cu cont pe aplicație, dar și la secțiunea de galerie foto, într-un format diferit față de utilizatori, sub formă de listă (figura 5.14).

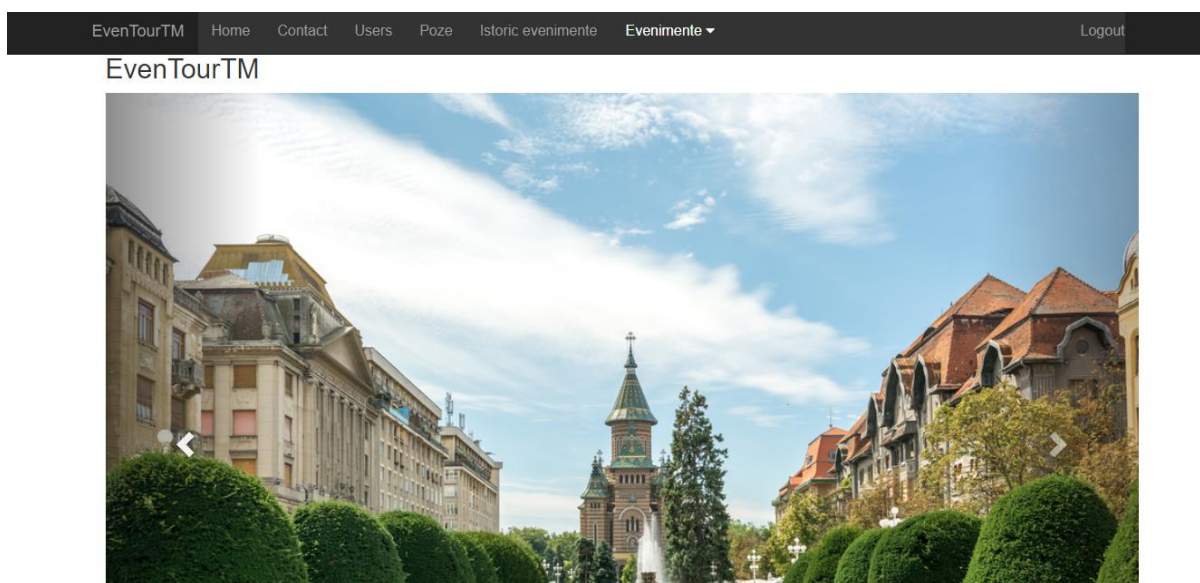


Figura 5.14 – Pagina principală pentru administrator

La selectarea unei categorii, administratorul va avea acces la lista cu evenimentele aferente acelei categorii, însă, pe lângă vizualizarea pe care o are asemenea utilizatorului, mai are acces la editarea, ștergerea și crearea unui eveniment. La categoria aleasă mai figurează doar un eveniment, deoarece ora de început a celui alt a trecut, astfel încât a fost mutat la secțiunea de Istoric evenimente (figura 5.15).

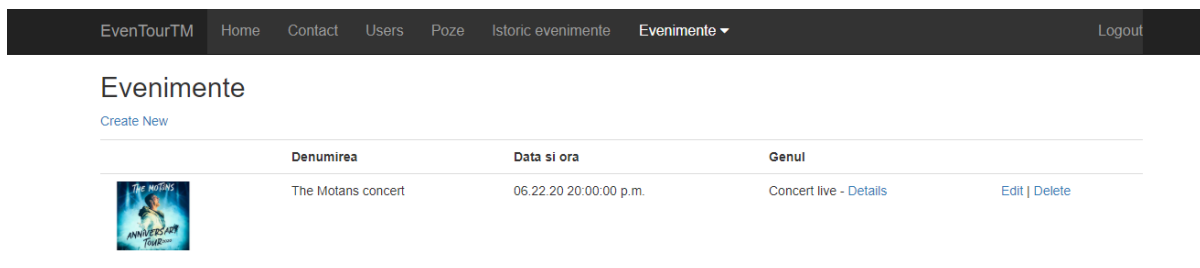


Figura 5.15 – Lista cu evenimente aferentă administratorului

Pagina detaliilor legate de evenimente este aceeași cu cea din cadrul modulului utilizatorului neautentificat, în sensul ca administratorul poate doar vedea comentariile, nu poate adăuga și el. Dacă intră la secțiunea de Create new, administratorul va putea adăuga un eveniment nou în categoria în care se află, prin completarea tuturor câmpurilor aferente cu date valide (figura 5.16).

EvenTourTM Home Contact Users Poze Istoric evenimente **Evenimente** Logout

Create

Eveniment

Denumirea

Data si ora

Descriere

Galerie foto Nu ai ales niciun fișier

Genul

[Back to List](#)

Figura 5.16 – Pagina de creare de eveniment nou

În cadrul secțiunii de editare, administratorul are posibilitatea de a edita informațiile fiecarui eveniment în parte, dacă este necesar acest lucru (figura 5.17).

EvenTourTM Home Contact Users Poze Istoric evenimente **Evenimente** Logout

Edit

Eveniment

Denumirea

Data si ora

Descriere

Galerie foto 22_the_mota...versar.jpg

Genul

[Back to List](#)

Figura 5.17 – Pagina de editare a unui eveniment

La selectarea opțiunii de ștergere, administratorul aplicației poate șterge un anumit eveniment aflat în lista evenimentelor afișate, din diverse motive, afișându-i-se o pagină în care sunt prezentate toate informațiile evenimentului respectiv, dar și un buton de confirmare pentru

ștergerea definitivă a evenimentului. La ștergerea evenimentului din listă, se șterge automat și galeria foto aferentă lui (figura 5.18).



Figura 5.18 – Pagina de ștergere a unui eveniment

La selectarea opțiunii de Istoric evenimente din bara de navigare, administratorului i se afișează, ca oricărui utilizator, toate evenimentele care s-au desfășurat deja, în plus, are la dispoziție opțiuni de editare și ștergere pentru fiecare eveniment în parte. Pentru fiecare dintre aceste opțiuni, interfața este aceeași cu cea prezentată mai sus la opțiunile de editare și ștergere (figura 5.19).

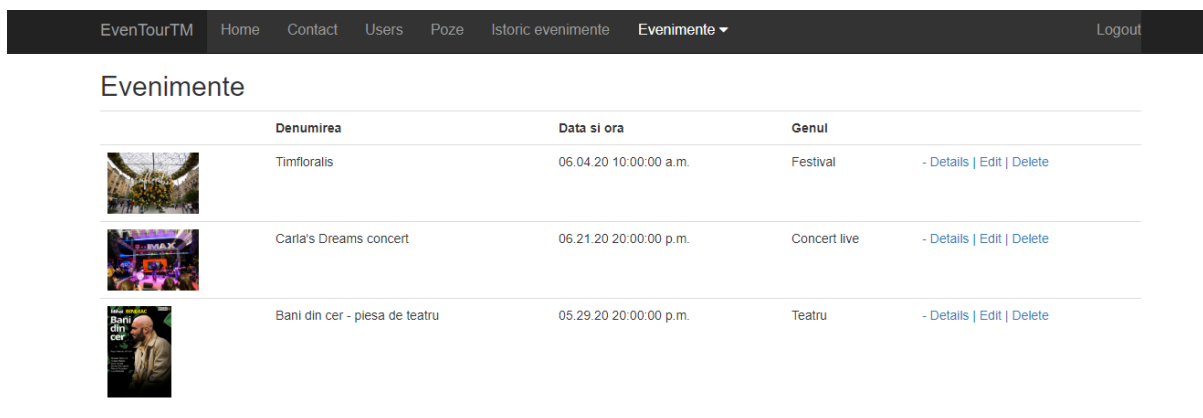


Figura 5.19 – Pagina Istoric evenimente pentru administrator

În cadrul secțiunii Poze din bara de navigare, administratorul are acces la o listă cu calea din calculatorul de pe care lucrează pentru fiecare poză din galeria foto a fiecărui eveniment. În cadrul acestei pagini, el are acces la opțiunile de creare de galerie foto, dacă a fost adăugat un nou eveniment, de detalii, de editare date, adică înlocuire fotografii și de ștergere galerie (figura 5.20).

EvenTourTM	Home	Contact	Users	Poze	Istoric evenimente	Evenimente ▾	Logout
Galerii foto							
Create New							
Denumirea	Poza1	Poza2	Poza3	Poza4			
Timfloralis	/images/timflo1.jpg	/images/timflo2.png	/images/timflo3.jpg	/images/timflo4.jpg - Details	Edit	Delete	
Carla's Dreams concert	/images/carlas1.png	/images/carlas2.jpg	/images/carlas3.jpg	/images/carlas4.png - Details	Edit	Delete	
The Motans concert	/images/motans1.jpg	/images/motans2.jpg	/images/motans3.jpg	/images/motans4.jpg - Details	Edit	Delete	
Bani din cer - piesa de teatru	/images/bani1.jpg	/images/bani2.jpg	/images/bani3.jpg	/images/bani4.jpg - Details	Edit	Delete	

Figura 5.20 – Pagina de galerii foto

La selectarea opțiunii de detalii se afișează o pagină cu galeria foto într-un slideshow aferentă evenimentului ales, asemenea cu ce văd ceilalți utilizatori. Opțiunea de creare de galerie foto nouă este necesară atunci când se adaugă un eveniment nou în lista de evenimente și căruia trebuie să i se asigneze o galerie foto nou creată și ea (figura 5.21).

EvenTourTM	Home	Contact	Users	Poze	Istoric evenimente	Evenimente ▾	Logout
Create							
Poze							
Eveniment	Timfloralis ▾						
Poza1	Alege fișierul Nu ai ales niciun fișier						
Poza2	Alege fișierul Nu ai ales niciun fișier						
Poza3	Alege fișierul Nu ai ales niciun fișier						
Poza4	Alege fișierul Nu ai ales niciun fișier						
Create							
Back to List							

Figura 5.21 – Pagina de creare a unei noi galerii foto

În cadrul secțiunii de editare, administratorul are posibilitatea de a edita fotografiile deja existente în galeria foto pentru un anumit eveniment, în sensul că poate schimba fotografiile pe care le dorește (figura 5.22).

Figura 5.22 – Pagina de editare a galeriei foto

În cazul în care administratorul alege opțiunea de ștergere a galeriei foto a unui eveniment, acestuia i se va deschide o pagină în care i se vor afișa informațiile aferente galeriei foto și un buton de confirmare a ștergerii galeriei (figura 5.23).

Figura 5.23 – Pagina de ștergere a unei galerii foto

Atunci când selectează butonul de utilizatori din bara de navigare, administratorul are acces la lista cu utilizatorii care au cont în aplicație, împreună cu datele lor furnizate la crearea contului (figura 5.24).

EvenTourTM Home Contact Users Poze Istoric evenimente Evenimente Logout		
Utilizatori		
Nume	Email	Password
Sorin	sorin@email.com	parola1
Administrator	admin	admin
Sorin Popescu1	popescu_sorin23@yahoo.com	parola2

Figura 5.24 – Pagina cu utilizatorii care au cont în aplicație

CAPITOLUL 6

CONCLUZII

6.1 Privire de ansamblu asupra aplicației

Aplicația „EvenTourTM” este utilă pentru a gestiona mai ușor toate evenimentele care se desfășoară în Timișoara, dar și pentru a le promova mult mai ușor prin intermediul internetului, a platformei web, cetățenilor orașului, dar și celor din alte orașe care doresc să participe la un anumit eveniment desfășurat aici, oferind un acces mult mai rapid către informațiile aferente evenimentelor de interes pentru ei. Elementul de noutate constă în utilizarea unei mesagerii în timp real în cadrul aplicației, asemenea unui forum, în care utilizatorii care folosesc în același timp funcția de mesagerie, pot schimba mesaje între ei. Un alt element important al aplicației îl reprezintă o interfață destul de interactivă cu utilizatorul, dar și aducerea informațiilor de interes pentru cetățeni cât mai aproape de aceștia, eliminând oricare alte mijloace de informare mai puțin comode decât acesta, care nu sunt poate la fel de veridice ca acesta și care necesita deplasarea fizică pentru obținerea lor.

Aplicația prezintă trei module destinate: administratorului, utilizatorului autentificat, utilizatorului neautentificat. Datele utilizatorilor autentificați sunt confidențiale față de restul utilizatorilor sau vizitatorilor, ei folosindu-le în scop personal pentru a-și crea propria experiență în aplicație, datele fiind vizibile doar pentru administrator.

6.2 Direcții de dezvoltare

O primă direcție de dezvoltare ar putea fi adăugarea opțiunii de introducere a oricărui oraș din România, astfel încât reprezentanții orașelor respective să aibă acces la aplicație printr-un nume de utilizator și o parolă de administrator specifice orașului lor și să poată adăuga, edita și șterge evenimente doar în cadrul orașului respectiv și, totodată, putând vizualiza doar utilizatorii arondați aceluia oraș.

O altă direcție de dezvoltare ar reprezenta adăugarea opțiunii de Hartă prin Google Maps în cadrul fiecărui eveniment în parte. Astfel, toți utilizatorii cărora le-ar pare interesant un anumit eveniment, dar ar întâmpina dificultăți în a ajunge la adresa respectivă, ar putea utiliza cu încredere aplicația de Hărți. Prin aceasta, utilizatorii au posibilitatea de a ajunge într-un mod mai rapid și mai comod către evenimentul favorit, chiar și în condițiile în care nu locuiesc în orașul respectiv.

BIBLIOGRAFIE

- [1] <https://www.atelierdeweb.ro/aplicatii-web/>
- [2] <https://www.iqads.ro/articol/42259/cele-mai-populare-aplicatii-in-organizarea-de-evenimente>
- [3] Hans-Petter Halvorsen, *Introduction to Visual Studio and C#*, Editura Telemark University College, Norway, 2014
- [4] <https://docs.microsoft.com/en-us/dotnet/framework/get-started/overview>
- [5] https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm
- [6] Nimit Joshi, *Programming ASP .NET MVC*, Editura C# Corner, New York, 2013
- [7] Adam Freeman, *Pro ASP.NET MVC 5*, Editura Apress, New York, 2013
- [8] Sergey Barskiy, *Code-First Development with Entity Framework*, Editura Packt Publishing, Birmingham, 2015
- [9] Tom Dykstra, Rick Anderson, *Getting Started with Entity Framework 6 Code First using MVC 5*, Editura Apress, New York, 2014
- [10] <https://www.entityframeworktutorial.net/what-is-entityframework.aspx>
- [11] Paolo Pialorsi, Marco Russo, *Programming Microsoft® LINQ in Microsoft .NET Framework 4*, Editura O'Reilly Media, Inc., California, 2010
- [12] https://en.wikipedia.org/wiki/Microsoft_SQL_Server
- [13] Stacia Varga, Denny Cherry, Joseph D'Antoni, *Introducing Microsoft SQL Server 2016*, Washington, 2016
- [14] https://www.tutorialspoint.com/ms_sql_server/ms_sql_server_overview.htm
- [15] <https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/>
- [16] John Sharp, *Microsoft Visual C# Step by Step, Ninth Edition*, Editura Pearson Education, New York, 2018
- [17] Abolrous Sam A., *Learn C# Includes the C# 3.0 Features*, Editura Wordware Publishing, Texas, 2008
- [18] https://www.tutorialspoint.com/asp.net_mvc/asp.net_mvc_razor.htm
- [19] <https://www.tutorialspoint.com/bootstrap/index.htm>
- [20] Aravind Shenoy, Ulrich Sossou, *Learning Bootstrap*, Editura Packt Publishing, Birmingham, 2014
- [21] https://en.wikipedia.org/wiki/Microsoft_Visual_Studio

- [22] <https://pusher.com/tutorials/realtime-comments-aspnet>
- [23] <https://www.geeksforgeeks.org/software-engineering-sdlc-v-model/>
- [24] <https://prbeta.ro/blog/diagrama-gantt/>