

CUPRINS

INTRODUCERE	3
CAPITOLUL I. Definirea și detaliile aplicației #LaOCarte	5
1.1. Definirea aplicațiilor web	5
1.2. Aplicații existente vs. aplicația propusă	6
1.3. Sisteme de recomandări	8
1.4. Rolurile și accesul în aplicație	9
CAPITOLUL II. Tehnologii utilizate în proiectarea și funcționalitatea aplicației ..	12
2.1. Programarea web utilizând ASP.NET	12
2.1.1. Introducere	
2.1.2. Mediul de dezvoltare	
2.1.3. Structura unei pagini în ASP.NET	
2.1.4. Securitatea	
2.1.5. Controale în ASP.NET	
2.1.6. Setări și configurări	
2.2. Limbajul C#	21
2.2.1. Introducere	
2.2.2. Avantaje	
2.2.3. Metode și librării folosite	
2.2.4. Clase în C#	
2.3. Baza de date	26
2.3.1. Introducere	
2.3.2. Schema relațională a bazei de date	
2.3.3. Extragerea datelor de pe site-uri	
2.3.4. Exemple de query-uri	
CAPITOLUL III. Tehnologii utilizate în design-ul aplicației	32

3.1. Introdúcere	32
3.2. HTML	34
3.3. CSS	34
3.3.1. Introdúcere	
3.3.2. LESS	
3.3.3. Bootstrap	
3.3.4. FontAwesome	
3.4. Javascript	41
3.4.1. Introdúcere	
3.4.2. jQuery	
CONCLUZII	47
BIBLIOGRAFIE	48
ANEXE	

INTRODUCERE

Prezenta lucrare de licență, își propune să aducă un nou concept de site web, românesc, pentru toate persoanele pasionate de cărți. Aceasta este o adaptare după binecunoscutul goodreads.com, adoptat într-o formulă locală, pentru România (site-ul fiind în limba română), iar limbajele folosite pentru crearea lui sunt în principal ASP.NET și C#, precum și câteva framework-uri pentru interfață (ex.: jQuery, Bootstrap, jCarousel, Bootstrap-Notify, jqCloud etc.).

Site-ul permite vizitatorilor (atât cei autentificați cât și cei neînregistrați) acces la un catalog online de peste 150 de cărți, la review-urile altor utilizatori precum și la un sistem de rating (disponibil în pagina de detaliu) pentru toate cărțile existente în baza de date. De asemenea, aceștia au posibilitatea creării unui profil personalizat, ceea ce le va permite la rândul lor să posteze recenzii și să voteze cărțile preferate. Totodată, utilizatorii înregistrați pot adăuga cărți în portofoliul personal după următoarele categorii:

- Cărți favorite
- Cărți citite
- Cărți de citit

Acestea se pot regăsi mai apoi în pagina de profil a utilizatorului împreună cu datele acestuia (nume utilizator și email) și o imagine de profil setată default care poate fi ulterior schimbată. În plus, utilizatorul are posibilitatea să-și schimbe datele folosite la crearea contului precum și parola folosită.

Pentru utilizatorii autentificați, deja “antrenați”, există posibilitatea recomandării de cărți, atât pe baza comportamentului acestuia cât și pe baza profilului unic deja creat.

Sistemul de recomandări va fi disponibil pe pagina principală împreună cu o secțiune de categorii și autori realizată într-un Tag Cloud.

De asemenea, există două tipuri de utilizatori autentificați. Sunt utilizatorii obișnuiți care au drepturile descrise mai sus și mai există admin-ul, care pe lângă specificațiile deja descrise,

poate modera comentariile altor utilizatori și poate adăuga cărți, genuri, edituri și autori noi în baza de date prin intermediul unui panou de control. Totodată acesta poate oferi altor utilizatori dreptul de admin.

Interfața se dorește a fi cât mai ușor de utilizat, iar design-ul acestuia, este unul cât mai simplu și modern, permițând atât utilizatorului obișnuit cât și persoanelor cu diferite dificultăți acces cât mai rapid și simplu la informațiile căutate. Printre principalele elemente din header-ul paginii sunt: meniul principal pentru navigarea spre celelalte pagini ale aplicației, butoanele de Login și Înregistrare precum și bara de căutare.

Spre deosebire de alte site-uri web similare, #LaOCarte, prezintă și stochează cărțile în baza de date ca entități unice, i.e. pentru fiecare carte se cunoaște titlul, autorul, descrierea, o poză de copertă și genul din care face parte. Editura și, respectiv codul ISBN, au fost lăsate deoparte iar editura a fost stocată separat în baza de date, deoarece se consideră că pentru utilizatori este mai important ce carte au citit sau doresc să citească mai departe decât de la ce editură este, evitând astfel duplicatele. Totuși, pentru evitarea oricăror neclarități, pe pagina de detaliu a fiecărei cărți se vor putea regăsi informații despre multiplele edituri (în cazul în care este cazul) unde a fost tipărită cartea precum și un link extern către adresa web a editurii.

Nu în ultimul rând, site-ul beneficiază de o pagină de contact unde orice utilizator poate trimite mail, având certitudinea că va primi și un răspuns la acesta, precum și o confirmare de trimitere.

CAPITOLUL I.

Noțiuni generale privind tehnologiile folosite

Definirea aplicațiilor web

Din punct de vedere tehnic, web-ul este un mediu programabil care permite personalizarea unei game largi și diverse de aplicații la nivel global, pentru milioane de utilizatori. Cele 2 componente ale unui site web modern sunt browser-ul și respectiv aplicația web, ambele disponibile pentru toată lumea, fără nici o cheltuială.

Browserele web sunt aplicații software pentru preluarea și prezentarea resurselor informaționale de pe internet (World Wide Web). O resursă de informații este identificată printr-un Uniform Resource Identifier (URI/URL), acesta putând fi o pagină web, o imagine, un video sau un fragment de conținut. Link-urile prezente în aceste resurse permite utilizatorilor navigarea în browser spre resursele aferente. Printre principalele și cele mai populare browsere se enumeră: Google Chrome, Mozilla Firefox, Internet Explorer, Opera și Safari.

Prin definiție, o aplicație web este un program de calculator ce rulează într-un browser web care permite vizitatorilor site-lui accesul și inserarea de informații într-o bază de date. Aplicațiile web interoghează server-ul (acesta cuprinzând baza de date) și produce în mod dinamic documentul web. Acesta este generat într-un format standard care permite afișarea acelui conținut în toate browserele (ex. HTML sau XHTML, CSS).

Câteva avantaje semnificative pentru construirea și menținerea unei aplicații web este că aceasta își îndeplinește scopul indiferent de sistemul de operare folosit, nu are aproape nici o cerință de instalare și pot fi implementate rapid fără nici un cost pentru utilizator.

De asemenea, website-urile moderne permit capturarea, procesarea, stocarea și transmiterea informațiilor private ale clienților (i.e. date personale, numerele de card, informații de securitate socială, etc.) pentru utilizare imediată și recurentă. Caracteristici precum pagini de login, mail, coșuri de cumpărături online etc. formează site-urile moderne și oferă

întreprinderilor mijloacele necesare pentru a comunica cu clienții. Pe măsură ce numărul firmelor care se bucură de avantajele afacerilor pe internet crește, și alte tehnologii legate de aceasta vor continua să se dezvolte.

Figura 1.1 de mai jos detaliază modelul unei aplicații web în trei straturi. Primul este în mod normal browser-ul web cu interfața utilizatorului. Al doilea este instrumentul de tehnologie dinamic care generează conținut și al treilea strat este cel de baze de date care depozitează conținutul (ex. articole, știri) și datele clienților (ex. nume de utilizator, parole, numere de securitate socială, etc.).

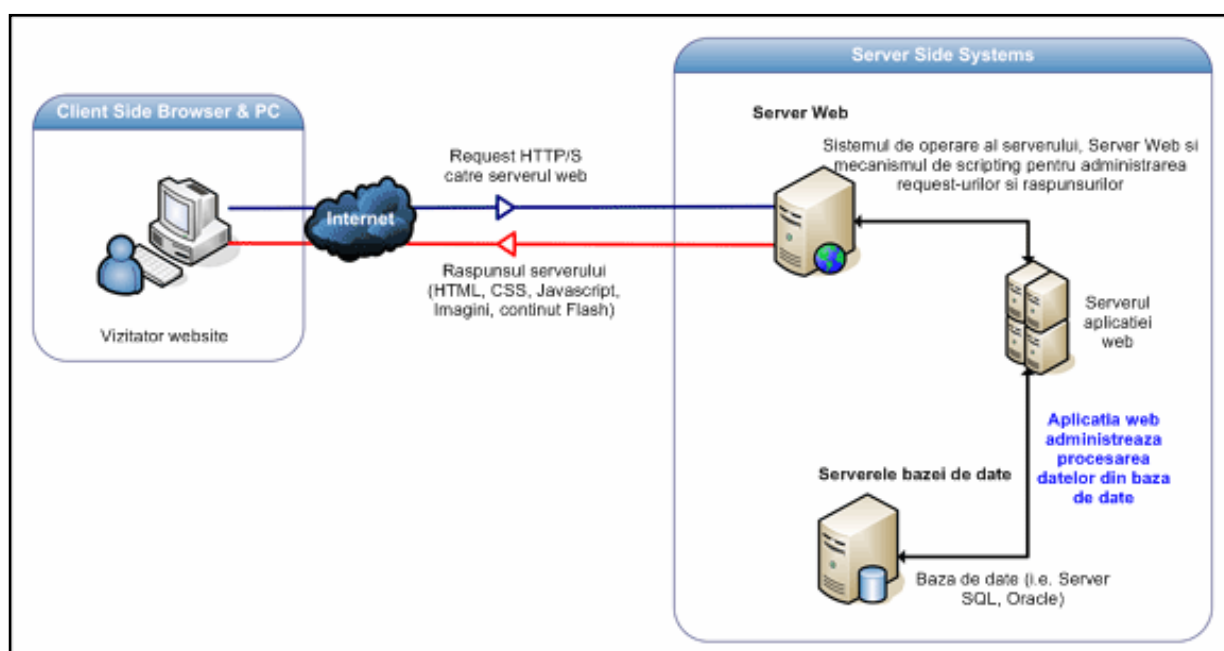


Figura 1.1. Modelul unei aplicații web

Aplicații existente vs. Aplicația propusă

În urma căutărilor pe internet putem împărți aplicațiile existente asemănătoare în două categorii:

1. Aplicațiile web românești
2. Aplicațiile web străine

Cele românești sunt reprezentate majoritar din bloguri personale ale iubitorilor de cărți unde se pot găsi prezentări și review-uri pentru cărțile citite de către proprietarul blogului. Pe lângă acestea, mai există site-urile editurilor și ale partenerilor (ex. elefant.ro), majoritatea lor fiind de tipul magazin online. Nu există nici un site web, românesc, de tip comunitate unde să se regăsească toate specificațiile aplicației descrise.

Printre aplicațiile străine, probabil cea mai cunoscută este goodreads, aceasta reprezentând și oarecum ca sursă de inspirație pentru #LaOCarte. Totuși, spre deosebire de aceasta, prezenta aplicație web își propune să ofere utilizatorilor o comunitate unică, oarecum simplificată, o bază de date cu cărți doar în limba română, fără duplicate, în funcție de editura la care au apărut și codul ISBN.

Există atât similitudini cât și diferențe între cele două aplicații, iar printre asemănări vom enumera:

- Amândouă permit crearea unui profil pentru fiecare utilizator și adăugarea cărților în diferite categorii (cărți de citit/want to read, cărți citite/read, etc.);
- Pe pagina de detaliu a fiecărei cărți se regăsesc atât detaliile despre cărți cât și secțiunea de comentarii precum și sistemul de rating;

Totuși spre deosebire de goodreads, utilizatorii care folosesc site-ul #LaOCarte nu au posibilitatea să vizualizeze profilul altor utilizatori, nu-i pot adăuga la prieteni (Add as Friend) și nici nu le pot vizualiza activitatea recentă pe pagina de Home. De asemenea, goodreads se încadrează într-o arie un pic mai largă, site-ul având și caracter de site de socializare, a.î. pe site se poate găsi și o secțiune separată de grupuri unde utilizatorii pot deveni membri, pot pune întrebări, pot trimite mesaje, etc.

Luând toate lucrurile în considerare, aplicația prezentată prezintă două avantaje față de goodreads datorită simplității sale (utilizatorii nu vor fi copleșiți de cantitatea de informație de pe site precum și de multiplele funcționalități) precum și faptul că aplicația este în limba română și nu mai există ceva asemănător în România.

Sisteme de recomandări

Un sistem de recomandări reprezintă o un sistem de filtrare de informații care caută și anticipează rezultatele căutării utilizatorului pe baza preferințelor ulterioare ale acestuia. Sistemele de recomandări au reușit să schimbe modul de comunicare al aplicațiilor web cu utilizatorii acesteia, prin creșterea interacțiunii și pentru a oferi o experiență mai bogată. Spre deosebire de experiența statică, în care utilizatorii caută singuri produsele, sistemul de recomandări încearcă în mod autonom să identifice recomandări pentru utilizatorii individuali pe baza achizițiilor și căutărilor anterioare, precum și pe baza comportamentului altor utilizatori.

Există două abordări de bază în implementarea unui sistem de recomandări:

- Filtrare colaborativă – se bazează pe modelul de comportament anterior al utilizatorilor. Aceasta se poate construi atât pe comportamentul unui singur utilizator cât și pe comportamentul altor utilizatori care au trăsături similare. Cel din urmă se bazează pe o colaborare automată a mai multor utilizatori și se formează grupuri filtrându-se preferințele similare ale acestora.
- Filtrare pe bază de conținut - Aceasta construiește profilul de preferințe pe baza preferințelor conținutului pe care utilizatorul le-a plăcut sau nu. Un exemplu al acestei abordări reprezintă folosirea informațiilor de navigare și a istoricului.

O exemplificare a abordărilor descrise mai sus se poate observa în figurile 1.2 și 1.3.

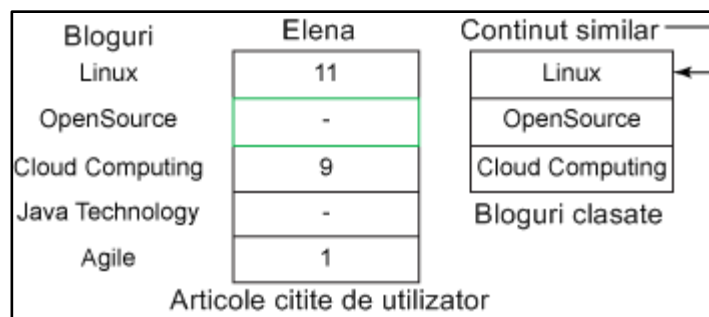


Figure 1.2. Filtrarea pe bază de conținut

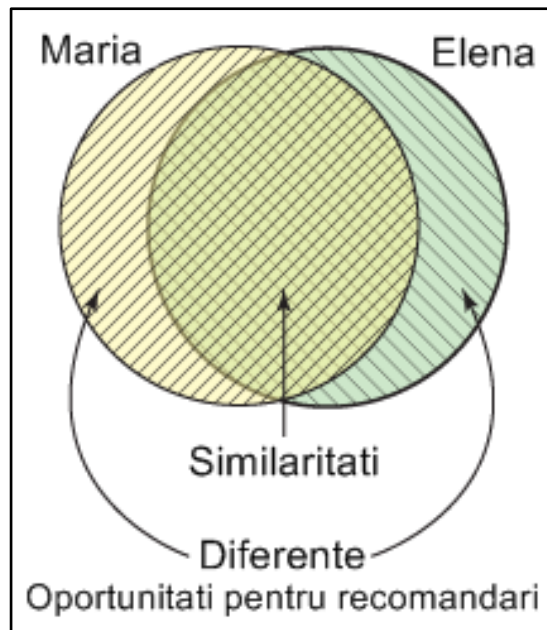


Figure 1.3. Filtrarea colaborativă

În cazul aplicației #LaOCarte, sistemul de recomandări existent pe pagina principală a aplicației se bazează pe filtrarea bazată pe conținut. În momentul creării profilului, utilizatorul are posibilitatea să marcheze cărțile preferate. În momentul în care există cel puțin o carte adăugată, sistemul de recomandări va calcula câte cărți din fiecare categorie există printre cărțile adăugate de utilizator la favorite. Pe baza acestui calcul, se vor putea recomanda cărți noi din categoriile cu rangul cel mai mare, ordinea lor făcându-se pe baza rating-ului cărților individuale în mod descrescător.

Rolurile și accesul în aplicație

Pentru început orice utilizator este un vizitator neînregistrat ce are acces atât pe pagina principală a aplicației precum și pe paginile de detaliu ale cărților și să vizualizeze rating-ul și comentariile fiecărei cărți. Totuși acesta nu are mai multe drepturi, nu poate vota la rândul său cărțile și nici nu poate posta comentarii sau să adauge cărțile în diferite categorii (de citit, citite și favorite).

În afară de utilizatorii neînregistrați, mai există cei care au profil deja creat și aceștia pot fi împărțiți în două în funcție de rolurile pe care le au:

1. Utilizatori normali – Users
2. Utilizatori cu drepturi de admin -Admin

Pentru oricare din cele 2 tipuri de utilizatori este necesară înregistrarea unde se cere:

- Nume de utilizator
- Adresă de email
- Parolă

După pasul de înregistrare, acesta se poate loga în aplicație și va avea toate drepturile descrise mai sus. În afară de acestea, utilizatorul va avea și o pagină de profil unde își poate schimba poza setată default și va putea vizualiza cărțile adăgate la diferitele categorii menționate. Aceasta se poate observa în Figura 1.4.

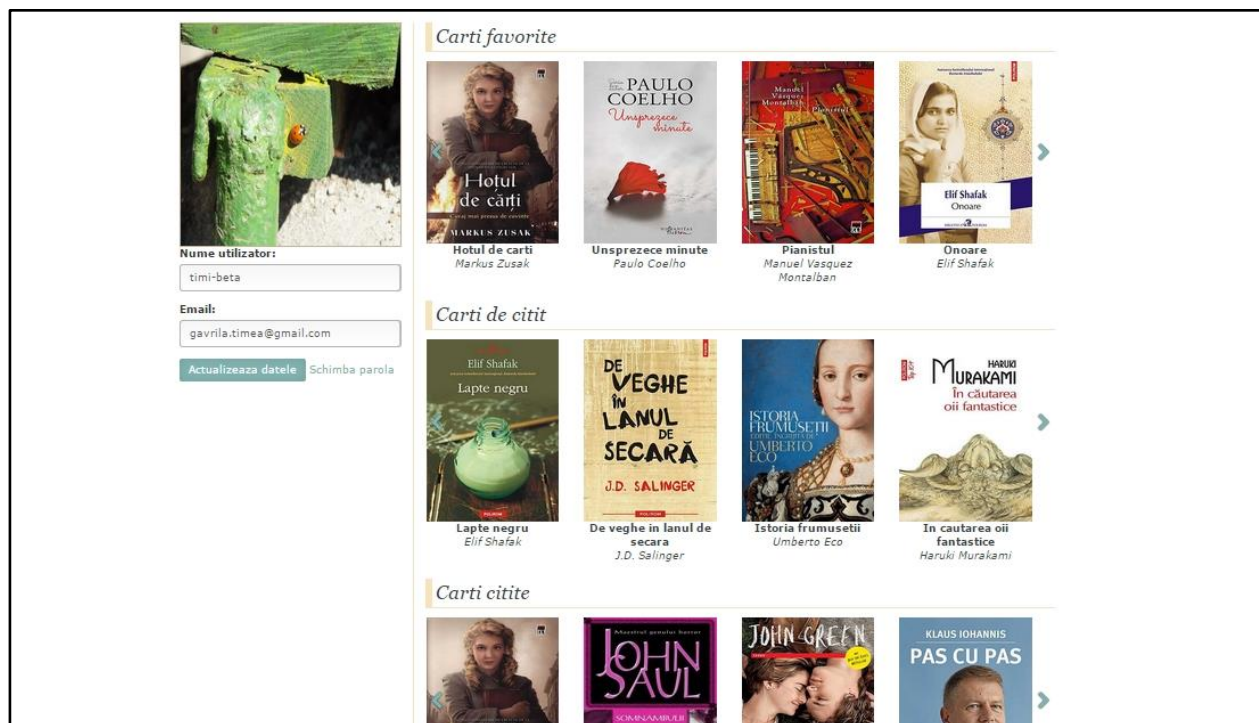


Figure 1.4. Pagina de profil a unui utilizator

Așa cum s-a menționat mai devreme, mai există și utilizatori de tipul Admin, care beneficiază de un tablou de admin din care:

- Pot vedea o listă cu toți utilizatorii site-ului și le pot acorda și acestora drepturi de

Admin;

- Pot modera comentariile publicate pe paginile de detalii ale cărților;
- Pot insera cărți, și date noi în baza de date;

CAPITOLUL II.

Tehnologii utilizate în proiectarea și funcționalitatea aplicației

Programarea web utilizând ASP.NET

Introducere

ASP.NET este un framework open source creat de Microsoft, special pentru dezvoltarea aplicațiilor dinamice și a serviciilor web. Acesta este succesorul ASP-ului (Active Server Pages) introdus pentru prima dată în 1998 ca și prim program de tip server-side al aplicațiilor. În plus față de acesta, ASP.NET beneficiază de avantajele platformei de dezvoltare .NET și de instrumentele oferite de aplicația “Visual Studio .NET”.

Acesta suportă trei modele de dezvoltare diferite:

1. Web Pages (Modele Single Pages, similare cu PHP-ul și clasicul ASP);

Clasicul ASP este un limbaj de scripting în care atât codul din spate, cât și limbajul de markup stau în același fișier, iar fiecare dintre aceste fișiere corespund unei pagini web. Aceasta metodă a fost foarte populară la vremea ei și multe site-uri au fost create astfel. Totuși, cu trecerea timpului, s-a dorit o mai bună separare a codului, caracteristici noi precum și posibilitatea reutilizării codului prin evitarea duplicării.

2. Web Forms (ASP.NET tradițional despre care vom vorbi în continuare);

Ca și în ASP, aplicațiile web realizate cu ASP.NET se bazează pe o abordare unde fiecare pagină este reprezentată sub forma unui fișier fizic (numic formular web). Spre deosebire de o pagină ce folosește ASP, o pagină *Web Forms* oferă unele îmbunătățiri, cum ar fi separarea codului și al limbajului de markup prin divizarea conținutului web în două fișiere diferite: unul pentru limbajul de markup și unul pentru codul C#. Această nouă metodă a fost folosită foarte

mulți ani și continuă să fie alegerea multor dezvoltatori web datorită facilităților pe care le oferă.

3. MVC (Model View Controller);

Spre deosebire de ASP.NET Web Forms care a fost introdus ca un înlocuitor pentru predecesorul său, ASP, ASP.NET MVC abandonează arhitectura folosită până acum și aceasta este înlocuită cu arhitectura Model-View-Controller (MVC – Model-Vizualizare-Controlor).

ASP.NET a introdus o mulțime de noi caracteristici pentru dezvoltatori, incluzând cod server-side compilat, tehnică folosită pentru a separa logica serverului de interfața utilizatorului, un model extensibil de control pe partea de server, o metodă ușor de utilizat de implementare și suport pentru formularele de validare, atât pe partea de interfață cât și server-side.

Ca și avantaje ale utilizării ASP.NET, se numără:

- Oferă un model de programare orientată pe obiecte, acesta conținând un set larg de componente, bazate pe XML;
- Are performanță ridicată deoarece rulează cod compilat. Codul sursă este împărțit de obicei în 2 fișiere, unul pentru codul executabil și unul pentru conținutul paginii (textul din pagina și cod HTML);
- .NET este compatibil cu mai multe limbaje de programare diferite, cele mai utilizate fiind Visual Basic și C#.

Mediul de dezvoltare

Microsoft Visual Studio este un IDE (Integrated Development Environment) creat de Microsoft folosit pentru dezvoltarea programelor pe desktop, a website-urilor, a aplicațiilor și serviciilor web. Acesta suportă diferite limbaje de programare și permit editarea codului pentru aproape orice limbaj. Printre cele integrate în Visual Studio se regăsesc: C, C++, C++/CLI, VB.NET, C# și F#. Totodată acesta suportă XML/XSLT, HTML/XHTML, Javascript și CSS.

Structura unei pagini în ASP.NET

Un atribut al unei aplicații web bine concepută îl reprezintă aspectul consistent al site-ului. Fiecare pagină ar trebui să aibă, pe lângă conținutul său personalizat, o formatare comună pentru toate paginile a.î. să se păstreze aceeași temă. Tema aplicației web poate fi realizată prin păstrarea header-ului și footer-ului pe toate paginile precum și folosirea acelorași stiluri și layout. Un exemplu standard de structură a unei pagini ASP.NET se poate vedea în figura de mai jos.

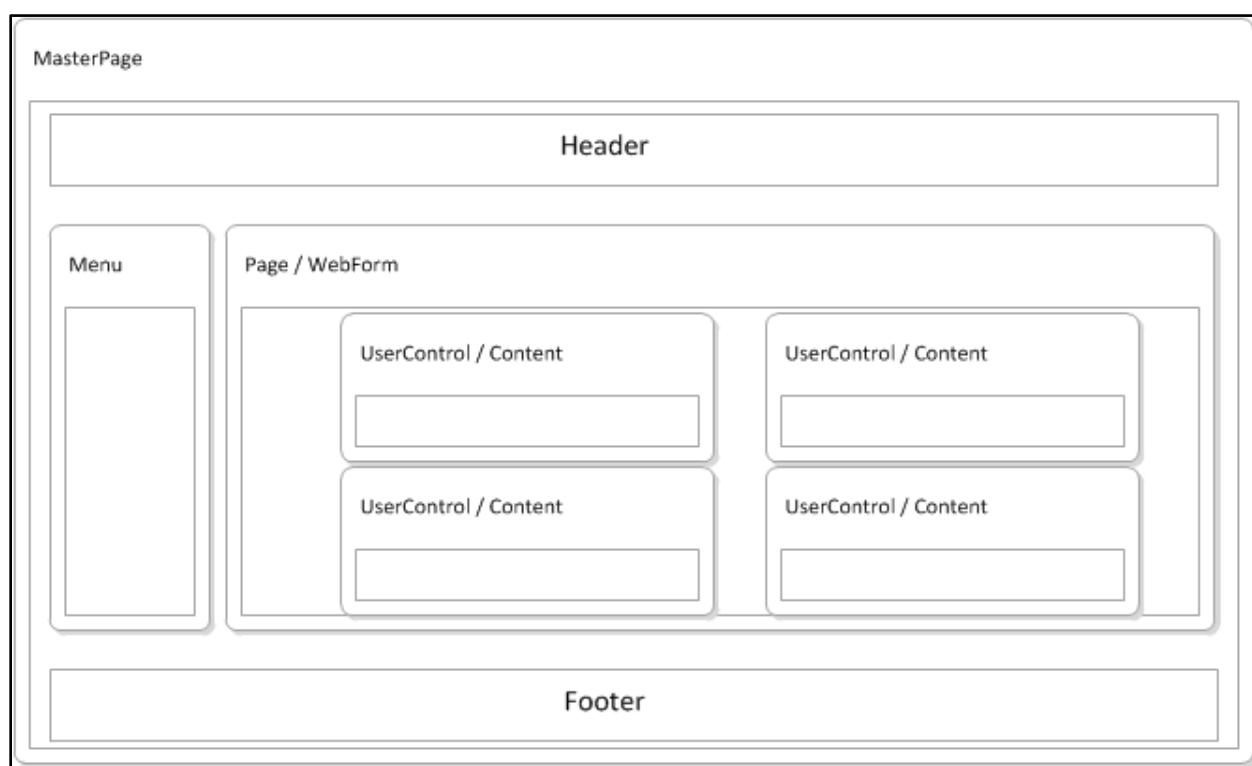


Figure 2.1. Structura unei pagini ASP.NET

Păstrarea aceleiași teme precum și repetarea header-ului și footer-ului pe toate paginilor se poate realiza printr-o varietate de tehnici. O abordare simplă ar fi copierea codului comun pe toate paginile web. Totuși această metodă prezintă o serie de dezavantaje printre care, cel mai important ar fi ca în cazul unei modificări al părții comune, această editare trebuie realizată pe toate paginile aplicației. Totuși, începând cu versiunea 2.0 a ASP.NET-ului s-a introdus noțiunea

de MasterPage.

Un MasterPage este un tip special de pagina care definește atât marcajul comun pentru site cât și definirea markup-ului pentru conținutul personalizat. Acestea sunt definite de către ContentPlaceHolder, care specifică pur și simplu poziția în ierarhia MasterPage-ului, unde va fi amplasat conținutul paginilor (WebForms). Printre avantajele folosirii unui MasterPage se pot aminti:

1. Permite centralizarea funcționalității comune a paginilor a.î. actualizările se pot face într-un singur loc.
2. Controlul asupra aspectului paginii finale prin accesul la partea de randare a codului.
3. Ușurează crearea unui set de coduri și controale care aplică rezultatele la un set de pagini. De exemplu, se poate crea un meniu care să se regăsească pe toate paginile.
4. Oferă un model de obiect care permite modelarea MasterPage-ului din interiorul paginilor individuale.

În aplicația propusă, există și un MasterPage care este format din header-ul și footer-ul paginilor. În header se găsesc: logo-ul (se poate regăsi în Anexa 3), butoanele de login și de înregistrare (la care se adaugă și modalul de login), butoanele de social media (Facebook Twitter, Google+, Pinterest, Instagram), bara de căutare și meniul.

În afară de MasterPage și conținutul specific fiecărei pagini mai există UserControls, care permit crearea și personalizarea anumitor părți și funcționalități pentru aplicație și inserarea acestora unde avem nevoie pe paginile site-ului. Un exemplu de UserControl este crearea unui Newsletter pe care vrem să-l afișăm pe mai multe pagini.

Securitatea

ASP.NET în colaborare cu Microsoft Internet Information Services (IIS) poate autentifica datele de login ale utilizatorului (numele de utilizator și parola) folosind una din următoarele metode de autentificare:

- Autentificare Windows;

- Autentificare prin intermediul form-urilor - se crează o pagină de login și se gestionează autentificarea în aplicație;
- Autentificare de tip Client Certificate.

ASP.NET controlează accesul la informațiile site-ului comparând credențialele de autentificare, sau reprezentarea acestora, cu drepturile pentru fișierele de sistem (NTFS) sau cu un fișier XML care listează utilizatori autorizați sau rolurile și grupurile autorizate.

După cum se poate observa în Figura 2.2. toți clienții comunică cu aplicația web prin intermediul IIS-ului. Microsoft Internet Information Services autentifică cererea, dacă este necesar și apoi localizează resursele solicitate (un exemplu ar fi pagina de profil a utilizatorului). În cazul în care clientul este autorizat, resursa este disponibilă.

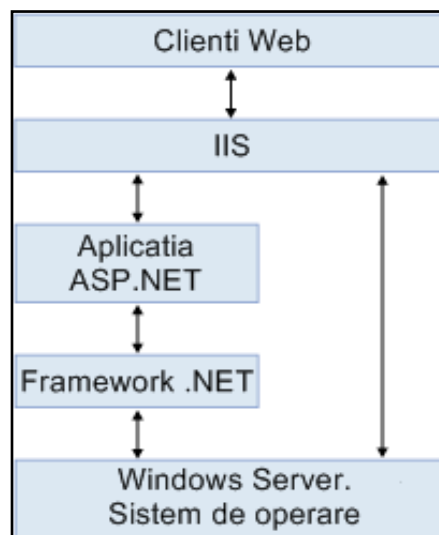


Figura 2.2. Arhitectura ASP.NET

Când o aplicație web rulează, aceasta poate folosi caracteristicile de securitate build-in ale ASP.NET-ului. În plus, aceasta poate folosi caracteristicile de securitate ale framework-ului .NET.

Controale în ASP.NET

ASP.NET ne pune la dispoziție o colecție de controale (obiecte) care se execută atunci când se solicită pagina, generând limbajul de markup pentru ca browser-ul să-l poată citi. Multe dintre controalele de server web se aseamănă cu elemente HTML cum ar fi butoanele și input-urile. De asemenea, cu ajutorul acestora se mai pot genera liste de elemente repetitive, care sunt legate la baza de date. Controalele de server sunt de trei tipuri:

1. HTML Server Controls - tag-urile HTML tradiționale
2. Web Server Controls – tag-urile ASP.NET
3. Validation Server Controls – folosite pentru validările input-urilor

Toate controalele de server trebuie să regăsească în interiorul tag-ului <form> și, pentru accesarea și procesarea lor de către server aibă atributul `runat="server"`.

În figura de mai jos se poate observa diferența între un tag traditional de HTML și același tag, dar în ASP.NET (Web Server Control).

Tag HTML	Tag ASP.NET
<code><button type="button" class="button" onclick="functieApelata()">Apasa!</button></code>	<code><form runat="server"> <asp:Button id="button1" Text="Apasa!" runat="server" OnClick="submit" CssClass="button"/> </form></code>

Figura 2.3. Tag HTML vs. Tag ASP

Un tip de controale speciale în ASP.NET sunt cele care fac legătura cu baza de date printr-o comandă SQL sau cu un fișier XML și afișează rezultatele în pagină (ex.: GridView, DataGrid, Datalist, ListView, Repeater). În timp ce Repeater-ul oferă control absolut asupra a ceea ce se generează în pagină, Datalist-ul, de exemplu, generează un tabel în jurul conținutului.

Datorită libertății oferite, în aplicație s-a folosit preponderent Repeater-ul. Un exemplu de cum arată un Repeater legat la o baza de date se poate vedea în Figura 2.4.

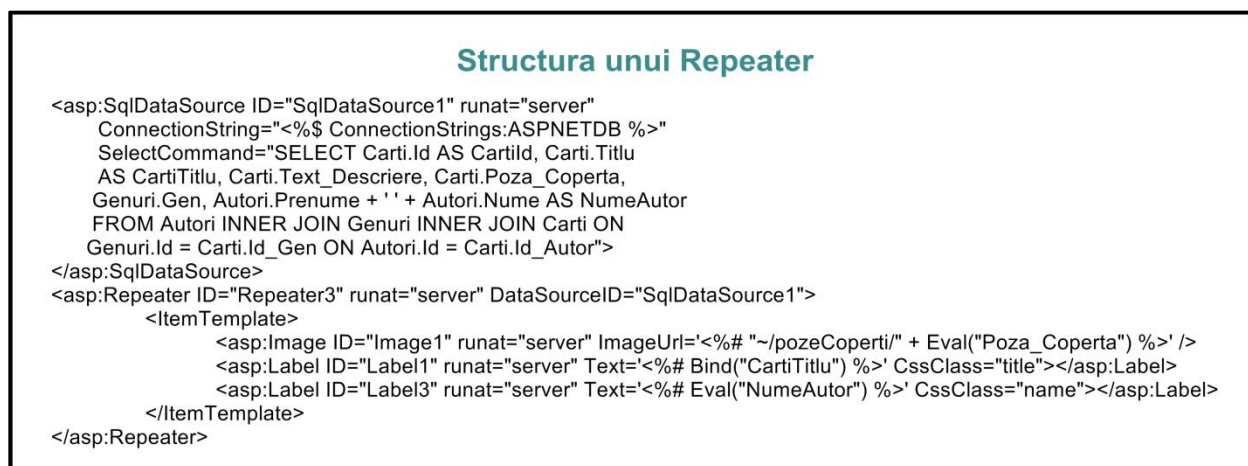


Figura 2.4. Structura unui Repeater

Un alt tip de control în ASP.NET este UpdatePanel. Acesta este folosit pentru actualizarea diferitelor părți din pagină, împiedicând astfel refresh-ul întregii pagini. Datorită faptului că pentru partea de login s-a utilizat un modal din Bootstrap, la apăsarea butonului de logare se actualizează pagina iar modalul dispărea automat. În cazul în care autentificarea utilizatorului nu era cu succes, mesajul de eroare, fiind tot în interiorul modalului, nu se putea vizualiza decât după apăsarea butonului de Login din header-ul paginii. Totuși, plasând UpdatePanel-ul chiar în afara tag-ului de login și în interiorul modalului, pagina rămâne aceeași, actualizându-se doar conținutul din interiorul acestuia. Codul împreună cu design-ul paginii (cu modalul de login) se pot regăsi în figurile de mai jos.

Modalul de login folosind UpdatePanel

```
<div class="modal-body">
  <asp:ScriptManager ID="ScriptManager1" runat="server"></asp:ScriptManager>
  <asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
      <asp:Login ID="Login" runat="server"
        LoginButtonText="Intra in cont"
        PasswordLabelText="Parola:"
        UserNameLabelText="Nume utilizator:"
        RememberMeText="Tine minte" TitleText=""
        FailureText="Login nereusit. Va rugam incercati din nou."
        CssClass="login-form-table">
        <LoginButtonStyle CssClass="btn primary-button" />
      </asp:Login>
    </ContentTemplate>
  </asp:UpdatePanel>
</div>
```

Figura 2.5. Folosirea UpdatePanel în interiorul modal-ului de Login



Figura 2.6. Modalul de Login

Setări și configurări

În ASP.NET acestea se realizează prin intermediul fișierului Web.config. Acesta conține setările principale de configurare pentru o aplicație ASP.NET. Fișierul este un document XML ce controlează modul de încărcare, de configurare de securitate, de limbă și setările de compilare. Web.config poate conține de asemenea, elemente specifice de aplicare cum ar fi șiruri de conexiune la baza de date. Un astfel de șir stă de obicei în interiorul tag-urilor <connectionStrings></connectionStrings> așa cum se poate observa în figura de mai jos.

```
<connectionStrings>
  <add name="ASPNETDB"
    connectionString="Data Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\ASPNETDB.MDF;Integrated Security=True;User Instance=True"
    providerName="System.Data.SqlClient" />
</connectionStrings>
```

Figure 2.7. Șirul de conexiune la baza de date

În afară de acesta se mai pot seta sau dezactiva condiții pentru partea de înregistrare a utilizatorilor. Un exemplu specific pentru aplicația #LaOCarte, așa cum se și exemplifică în figura de mai jos, este scoaterea întrebării de securitate care vine setată ca default true în ASP.NET precum și scoaterea condiției ca parola să conțină minim un caracter non-alfanumeric.

Setari privind inregistrarea utilizatorilor

```
<membership defaultProvider="SqlProvider">
  <providers>
    <clear/>
    <add name="SqlProvider" type="System.Web.Security.SqlMembershipProvider"
      connectionStringName="ASPNETDB" enablePasswordRetrieval="false"
      enablePasswordReset="true" requiresQuestionAndAnswer="false" requiresUniqueEmail="true"
      maxInvalidPasswordAttempts="5" minRequiredPasswordLength="6"
      minRequiredNonalphanumericCharacters="0" passwordAttemptWindow="10" applicationName="/" />
  </providers>
</membership>
```

Figure 2.8. Setări în Web.config

Limbajul C#

Introducere

C# este un limbaj de programare orientat pe obiect, imperativ, declarativ și generic, care permite printre altele atât dezvoltarea de aplicații Windows cât și aplicații de tip client-server și baze de date. Acesta a fost dezvoltat de Microsoft, în interiorul framework-ului .NET.

Sintaxa limbajului este în același timp expresivă și simplă. Pentru dezvoltatorii deja familiarizați cu C, C++ sau Java, C# este cu atât mai ușor de învățat cu cât sintaxa simplifică mult complexitatea C++-ului, oferă caracteristici precum variabile de tip null, enumerații, acces direct la memorie și multe altele care nu se regăsesc în Java. Totodată, C# suporta metode și tipuri generice, care ofera siguranță și sporește performanța.

Ca un limbaj de programare orientat pe obiect, C# permite încapsularea, moștenirea și polimorfismul. Toate variabilele și metodele (incluzând și metoda Main), sunt încapsulate în definițiile claselor. În plus, C# faciliteaza dezvoltarea componentelor software prin mai mulți constructori inovatori, cum ar fi:

1. Semnături de metode încapsulate numite, care permit notificări pentru evenimentele type-safe.
2. Proprietăți care permit accesul la variabilele private.
3. Atribute care oferă date declarative despre tipurile folosite la momentul execuției.
4. Documentație sub forma de comentarii inline pentru XML.
5. Language-Integrated Query (LINQ) – permite interogări pe o varietate de surse de date.

Avantaje

Principalul avantaj în folosirea unui limbaj de programare precum C#, spre deosebire de altele, este faptul că acesta rulează pe CLR (Common Language Runtime), o mașină virtuală, componentă a framework-ului .NET care gestionează executarea programelor. Aceasta permite integrarea mai ușoară a altor componente scrise în alte limbaje (în special a celor compatibile cu CLR). Totodată, prin C# avem acces la toate bibliotecile .NET-ului, biblioteci ce sunt destul de

extinse iar implementarea acestora se realizează destul de ușor.

Metode și librării folosite

1. Legarea datelor obținute din baza de date în controale din ASP.NET (ex. Repeater)

Când avem nevoie de anumite condiții pentru query-ul de SQL, nu mai este suficient doar SelectCommand-ul din ASP.NET. Un exemplu ar fi pagina de detalii a cărților unde pentru fiecare carte ne trebuie id-ul ei (care este transmis prin URL). În cazul acesta popularea Repeater-ului cu date se realizează din C# așa cum se poate observa în Figura 2.9.

Popularea controalelor cu date din C#

```
string q = Request.Params["q"];
if (q != null)
{
    try
    {
        q = Server.UrlDecode(q);
        SqlDataSource1.SelectCommand = "SELECT Carti.Id AS CartiId,
        Carti.Titlu AS CartiTitlu, Carti.Poza_Coperta, Carti.Text_Descriere,
        Genuri.Gen, Autori.Prenume + ' ' + Autori.Nume AS NumeAutor,
        ROUND(AVG(ISNULL(NoteDateCartilor.Nota, 0)), 0) AS MedieNote
        FROM Carti INNER JOIN Genuri ON Carti.Id_Gen = Genuri.Id
        INNER JOIN Autori ON Carti.Id_Autor = Autori.Id
        LEFT JOIN NoteDateCartilor ON Carti.Id = NoteDateCartilor.Id_Carte
        WHERE Carti.Id = @q
        GROUP BY Carti.Id, Carti.Titlu, Carti.Poza_Coperta, Carti.Text_Descriere,
        Genuri.Gen, Autori.Prenume + ' ' + Autori.Nume";
        SqlDataSource1.SelectParameters.Clear();
        SqlDataSource1.SelectParameters.Add("q", q);
        SqlDataSource1.DataBind();
    }
    catch (Exception err)
    {
        Response.Write(err);
    }
}
```

Figura 2.9. Popularea controalelor din C#

2. Trimiterea mail-urilor

Framework-ul Microsoft .NET oferă, printre altele, un namespace, System.Net care permite implementarea protocoalelor de internet folosite de aplicații pentru trimiterea și primirea de date de pe internet. Unul dintre acestea este SMTP (Simple Mail Transfer Protocol), folosit pentru

trimiterea de e-mail-uri din C#. Acesta inițializează clasa SmtplibClient și îi atribuie un Host și un Port. Următoarea figură arată cum se realizează o funcție ce poate trimite e-mail-uri din C# folosind o adresă de Gmail. Funcția se apelează doar la apăsarea butonului de trimitere și doar în cazul în care adresa de email din formularul de contact este una validă.

Trimiterea e-mail-urilor în C#

```
using System.Net;
protected void SendMail()
{
    var fromAddress = "timeagavrilă.contact@gmail.com";
    var toAddress = YourEmail.Text.ToString();
    const string fromPassword = "Parola123";
    string body = "Nume: " + YourName.Text + "\n";
    body += "Email: " + YourEmail.Text + "\n";
    body += "Subiect: " + YourSubject.Text + "\n";
    body += "Mesaj/Întrebare: \n" + Comments.Text + "\n";
    var smtp = new System.Net.Mail.SmtpClient();
    {
        smtp.Host = "smtp.gmail.com";
        smtp.Port = 587;
        smtp.EnableSsl = true;
        smtp.DeliveryMethod = System.Net.Mail.SmtpDeliveryMethod.Network;
        smtp.Credentials = new NetworkCredential(fromAddress, fromPassword);
        smtp.Timeout = 20000;
    }
    string sub = "Formular contact";
    smtp.Send(fromAddress, toAddress, sub, body);
}
```

Figura 2.10. Funcția pentru trimiterea mail-urilor din C#

3. Păstrarea informațiilor unei pagini web după refresh

În urma apăsării unui buton din pagină, care are cod de executat în C#, de cele mai multe ori se execută automat un refresh. În urma acestuia, chiar dacă se încerca afișarea unui mesaj de succes sau de eroare, acesta era executat înaintea refresh-ului paginii iar pentru utilizator era imposibil să vizualizeze rezultatele acțiunii sale. Totuși, această mică dificultate poate fi depășită cu ajutorul cookie-urilor.

Cookie-urile reprezintă mici bucăți de informație trimise de către site și reținute în browser-ul web. De fiecare dată când se realizează un refresh la site, browser-ul trimite cookie-ul înapoi

către server pentru a-l informa de activitățile anterioare ale utilizatorului. Există două tipuri de cookies, cele de sesiune și cele persistente. Primele sunt create temporar și sunt valabile doar cât timp utilizatorul se află pe site. Cel de-al doilea tip de cookie, cele persistente rămân stocate în subfolder-ele browser-ului și se activează în momentul în care utilizatorul se întoarce înapoi pe site.

În pagina de detaliu a fost necesară crearea unui cookie în momentul în care un utilizator autentificat adaugă o carte nouă la favorite. Fiind o acțiune cu succes, trebuie să înștiințăm utilizatorul printr-un mesaj, după refresh-ul paginii. Aceasta a fost realizată cu ajutorul unui cookie, setat în C# așa cum se poate observa în figura de mai jos. Imediat după afișarea mesajului de succes sau în cazul unei erori, cookie-ul trebuie șters a.î. acesta să nu apeleze funcția la fiecare refresh al paginii.

Folosirea cookie-urilor in C#

```
protected void Page_Load(object sender, EventArgs e)
{
    try
    {
        if (Session["status"] != null && Session["status"].ToString().Equals("1"))
        {
            Page.ClientScript.RegisterStartupScript(this.GetType(), "ErrorFunction", "errorMessages('Mesaj succes','success');", true);
            Session.Remove("status");
        }
    }
    catch (Exception err)
    {
    }
}
/*...*/
if (success)
{
    Session["status"] = "1";
    Response.Redirect(Request.RawUrl, false);
}
else
{
    Page.ClientScript.RegisterStartupScript(this.GetType(), "ErrorFunction", "errorMessages('Mesaj eroare!','danger');", true);
    Session.Remove("status");
}
```

Figur 2.11. Setarea și folosirea unui cookie în C#

4. Apelarea unei funcții Javascript din C#

Așa cum se poate observa în figura de mai sus, pentru mesajele de eroare sau de succes este

necesară o funcție Javascript (funcție ce transformă mesajele în blocuri dinamice folosind un plugin de jQuery). Aceasta se poate realiza cu ajutorul metodei Page.ClientScript.RegisterStartupScript.

5. Implementarea filtrelor de căutare

Având o bază de date destul de extinsă, pe lângă bara de search, a fost nevoie de implementarea unor filtre cu ajutorul cărora utilizatorul să găsească informațiile căutate cât mai ușor. Acestea sunt în funcție de genul cărții și de autor și sunt implementate în C# sub forma unor funcții apelate la fiecare checkbox apăsăat în pagină. În funcție de opțiunile selectate, acestea sunt adăugate query-ului de SQL și se reconstruiește astfel Repeater-ul cu condițiile noi adăugate. Exemplul pentru selectarea genurilor cărților se poate găsi în figura de mai jos.

Funcțiile apelate pentru filtrele de cautare

```
String defaultSql = "SELECT Carti.Id AS CartId, Carti.Titlu AS CartiTitlu,
                    Carti.Text_Descriere, Carti.Poza_Coperta, Genuri.Gen,
                    Autori.Prenume + ' ' + Autori.Nume AS NumeAutor
                    FROM Autori INNER JOIN Genuri INNER JOIN Carti ON
                    Genuri.Id = Carti.Id_Gen ON Autori.Id = Carti.Id_Autor WHERE 1=1";

static String genSql = "";
static String autorSql = "";
protected void runFilter() {
    SqlDataSource1.SelectCommand = defaultSql + genSql + autorSql;
    SqlDataSource1.SelectParameters.Clear();
    SqlDataSource1.DataBind();
}

protected void Select_Gen(object sender, EventArgs e) {
    string genSelected = "";
    if (Repeater2.Items.Count > 0) {
        for (int count = 0; count < Repeater2.Items.Count; count++) {
            CheckBox genuri = (CheckBox)Repeater2.Items[count].FindControl("Genuri");
            if (genuri.Checked) {
                if (genSelected != "") {
                    genSelected = genSelected + ", ";
                }
                genSelected = genSelected + "" + genuri.Text + "";
            }
        }
    }
    if (genSelected != "") {
        genSql = "AND Genuri.Gen IN (" + genSelected + ")";
    }
    else {
        genSql = "";
    }
    runFilter();
}
```

Figura 2.12. Funcția pentru filtrarea genului cărților

Clase în C#

C# fiind limbaj de programare orientat pe obiect, este posibilă definirea și folosirea claselor. În primul rând o clasă reprezintă un grup de metode și variabile legate. O clasă descrie unul sau mai multe obiecte care pot fi declarate printr-un set uniform de date și cod (funcționalitate). În C# se pot defini mai multe categorii particulare de date și membri care conțin cod, printre care se regăsesc: constante, variabile, proprietăți, constructori, destructori, metode, evenimente, etc.

O astfel de clasă s-a folosit în aplicație pentru a defini într-un singur loc conexiunea la baza de date, urmând ca în proiect să se folosească o instanțiere a acestei clase. Detaliile acestei clase precum și modul în care a fost apelată se pot observa în Figura 2.13.

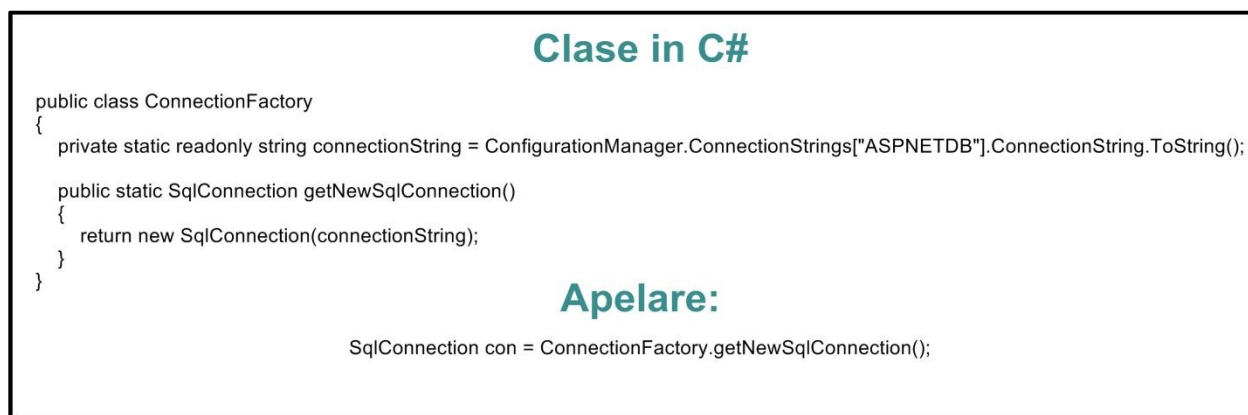


Figure 2.13. Definirea unei clase în C#

Baza de date

Introducere

O bază de date reprezintă o colecție de informații organizate a.î. un program pe calculator poate selecta rapid datele necesare. Aceasta mai poate fi numită și sistem de evidență electronic.

Un sistem de gestiune al bazelor de date (DBMS) este o aplicație software care interacționează cu un utilizator sau cu alte aplicații precum și cu baza de date în sine pentru

captarea și analizarea datelor. Aceste sisteme sunt proiectate pentru a permite crearea, definirea, interogarea, actualizarea și administrarea bazelor de date. Printre cele mai cunoscute DBMS-uri se enumără: MySQL, Microsoft SQL Server, Oracle, PostgreSQL, Sysbase și IBM DB2.

Schema relațională a bazei de date

Primul lucru care se face în procesul creării unei baze de date este realizarea unui model de date conceptual care reflectă structura informațiilor ce vor fi stocate în baza de date. Totodată modelul trebuie să determine structura logică și determinarea modului în care datele vor fi stocate, organizate și manipulate. Cel mai popular exemplu de model de bază de date este cel relațional care utilizează un format tabel (un exemplu mai clar se poate găsi în Anexa 1 pentru schema bazei de date ASP.NET Membership).

O schemă a bazei de date definește entitățile și relațiile lor. Aceasta trebuie să conțină detalii descriptive despre baza de date, care se pot fi descrise cu ajutorul diagramelor schemă. O schemă poate fi separată în 2 mari categorii:

1. Schema fizică a bazei de date – aceasta este alcătuită din date stocate efectiv precum și forma de memorare a acestora (i.e. fișiere, indici, etc.). Aceasta definește modul în care datele vor fi stocate într-un depozit secundar.
2. Schema logică a bazei de date – Aceasta definește toate constrângerile logice care trebuie aplicate pe datele stocate. Definește tabele, vizualizări și constrângeri de integritate.

În afară de shema ASP.NET Membership și în funcție de cerințele aplicației, a fost creată încă o schemă, în continuarea primei (deoarece cele două sunt legate prin tabelul de utilizatori – aspnet_Users). Aceasta se poate găsi în Anexa 2.

Extragerea datelor de pe site-uri

Baza de date a aplicației web reprezintă o parte din baza de date a site-ului elefant¹.

¹ <http://www.elefant.ro/>

Extragerea datelor a fost realizată cu ajutorul programului import.io prin antrenarea acestuia pe mai multe pagini individuale de unde s-au selectat câmpurile dorite (titlu, autor, gen, descriere și poză) urmând ca mai apoi acesta să navigheze singur pe restul site-ului și să culegă informațiile dorite. Totuși, datele extrase sunt într-o formă nestructurată iar câmpurile din tabel nu prezintă relații între ele. Din acest motiv, datele au fost preluate individual și introduse din panoul de admin al aplicației. În Figura 2.14. și 2.15. se poate observa modalitatea de extragere a darelor cu import.io în forma lor inițială, precum și panoul de introducere a datelor din aplicația web.

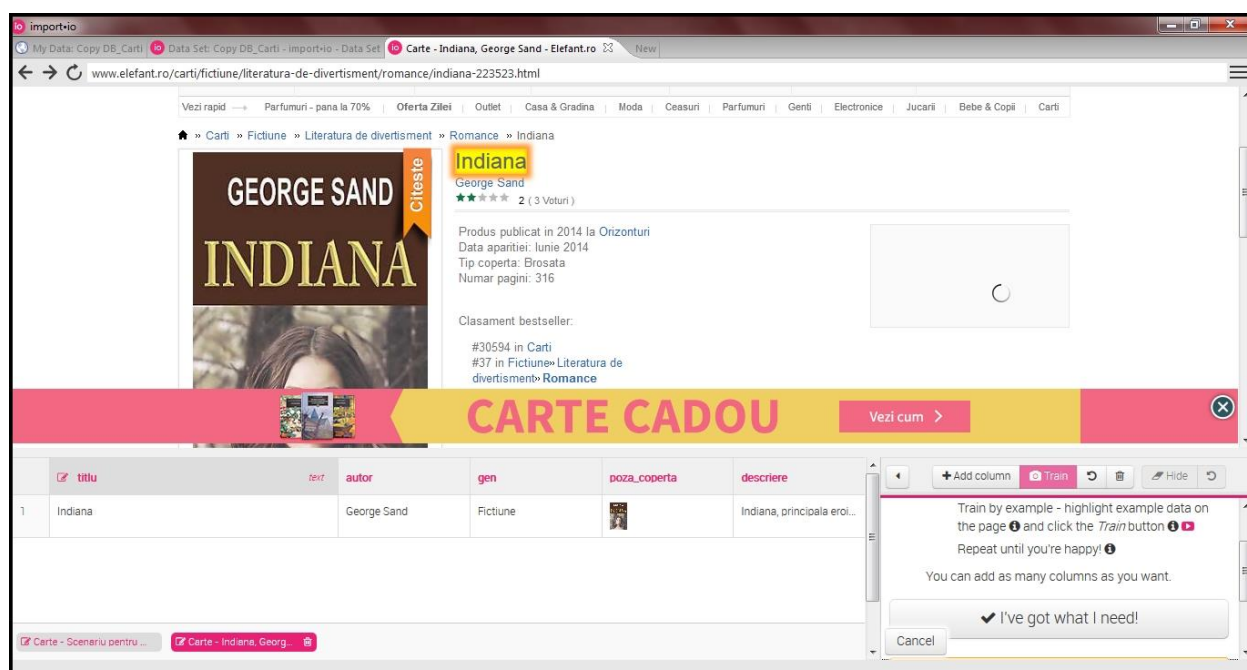


Figura 2.14. Extragerea datelor cu import.io

The screenshot shows the #LAOCARTE web application interface. At the top, there is a navigation bar with the site logo, social media icons, and links for Login and Inregistrare. Below this is a search bar with the placeholder text 'Autor / Titlu / Gen' and a 'Cauta' button. The main content area features a sidebar menu on the left with options: 'Adauga Editura', 'Adauga Autor', 'Adauga Gen', 'Adauga Carte', and 'Adauga relatie carte-editura'. The central form is titled 'Inserare al datelor în baza de date' and contains the following fields: 'Titlu' (text input), 'Autor' (dropdown menu with 'John Saul' selected), 'Gen' (dropdown menu with 'Fictiune' selected), 'Descriere:' (text area), and 'Poza Coperta' (file upload button labeled 'Choose File' with the text 'No file chosen'). A green 'Insereaza' button is at the bottom of the form. The footer contains the copyright notice: '© 2014 - 2015 Gavrilă Timea-Maria. Toate drepturile rezervate.'

Figura 2.14. Panoul de inserare al datelor în baza de date

Exemple de query-uri

Pentru pagina de detalii a fiecărei cărți, string-urile de SQL au fost create din C# datorită faptului că parametrul pentru identificarea fiecărei cărți (id-ul) a fost trimis prin url. Pentru această pagină au fost nevoie de trei SQL-uri separate:

1. Pentru detaliile despre carte (titlu, autor, gen, descriere, poza de copertă, rating-ul);
2. Pentru comentariile despre carte (textul, numele utilizatorului, data la care a fost postat);
3. Pentru sugestii de alte cărți din aceeași categorie (titlu, autor, poza de copertă).

Pentru primul query, obținerea mediei de rating a fost obținută cu ajutorul unui LEFT JOIN, astfel:

```
ROUND(AVG(ISNULL(NoteDateCartilor.Nota, 0)), 0)
```

Datorită faptului că nu toate cărțile au deja rating, ISNULL, transformă null-ul în zero, pentru cărțile care nu nici o notă acordată. AVG face media dintre notele existente și pentru că avem nevoie de un număr întreg, la final se folosește ROUND, care, așa cum îi spune și numele, rotunjește numărul primit ca prim parametru, la numărul de zecimale din al doilea parametru (în cazul de față, acesta este zero).

Tot pe pagina de detalii a fiecărei cărți, în secțiunea de comentarii, se afișează numele

utilizatorului, textul, data la care a fost postat comentariul și poza utilizatorului în cazul în care aceasta există (în caz contrar, la returnarea unui rezultat null, se va apela o funcție ce va înlocui null-ul cu numele pozei default). În figura de mai jos se poate vedea query-ul pentru secțiunea de comentarii postate. Parametrul @q este mai târziu atribuit în C#, id-ului cărți (id preluat din link-ul paginii).

Query-uri la Baza de Date

```
SELECT Comentarii.Id, Comentarii.Comentariu_Text, Comentarii.Data,  
Comentarii.Id_Carte, aspnet_Users.UserName, PozeUseri.Poza_User  
  
FROM aspnet_Users  
INNER JOIN Comentarii ON aspnet_Users.UserId = Comentarii.Id_User  
LEFT OUTER JOIN PozeUseri ON Comentarii.Id_User = PozeUseri.Id_User  
  
WHERE Comentarii.Id_Carte = @q
```

Figura 1.16. Query pentru afisarea comentariilor unei cărți

Un alt exemplu deosebit de query este cel pentru homepage unde s-a construit sistemul de recomandări. În cazul utilizatorilor autentificați care au cărți în secțiunea de favorite, se vor selecta top 6 cărți din baza de date, Query-ul trebuie construit a.î. să găsească cărți pe care utilizatorul curent nu le are deja la favorite, dar să fie din categoria predominantă a raftului său de cărți, iar acestea vor fi ordonate descrescător după rating-ul primit de alți utilizatori.

Query-ul arată astfel:

Query sistem de recomandari

```
SELECT TOP 6 Carti.Id AS CartId, Carti.Titlu AS CartiTitlu, Carti.Poza_Coperta, Carti.Text_Descriere, Genuri.Gen,
    Autori.Prenume + ' ' + Autori.Nume AS NumeAutor, ROUND(AVG(ISNULL(NoteDateCartilor.Nota, 0)), 0) AS MedieNote
FROM Carti
INNER JOIN Genuri ON Carti.Id_Gen = Genuri.Id
INNER JOIN Autori ON Carti.Id_Autor = Autori.Id
LEFT JOIN NoteDateCartilor ON Carti.Id = NoteDateCartilor.Id_Carte
WHERE Genuri.Gen = @gen
AND NOT EXISTS (SELECT CartiFavorite.Id_Carte
    FROM CartiFavorite
    WHERE CartiFavorite.Id_Carte = Carti.Id
    AND CartiFavorite.Id_User = @user)
GROUP BY Carti.Id, Carti.Titlu, Carti.Poza_Coperta, Carti.Text_Descriere, Genuri.Gen, Autori.Prenume + ' ' + Autori.Nume
ORDER BY MedieNote DESC
```

Figure 3.17. Query pentru sistemul de recomandări

CAPITOLUL III.

Tehnologii utilizate în design-ul aplicației

Introducere

Web design-ul cuprinde mai multe aptitudini și discipline în realizarea și întreținerea unei aplicații web. Termenul de web design este în mod normal folosit pentru descrierea procesului de colectare de idei, precum și aranjarea estetică și organizarea acestora (proiectarea design-ului front-end), ghidat de anumite principii și pentru un anumit scop. Acesta este un proces de creație cu intenția de a prezenta conținutul unei pagini web într-un mod cât mai inovator și atractiv.

În web design se utilizează mai multe categorii de elemente vizuale cheie pentru proiectarea paginii:

- Layout-ul – Acesta este modul în care conținutul paginii (textul, imaginile, reclamele, etc.) este aranjat. Un obiectiv cheie al acestuia este să ajute utilizatorul să găsească informațiile necesare dintr-o privire. Layout-ul presupune menținerea echilibrului, consistența și integritatea de proiectare. Pentru crearea unui prototip de layout, s-a folosit în realizarea aplicației câteva mockup-uri create cu ajutorul Balsamiq-ului². Acestea se pot regăsi în Anexa 3.
- Culorile – Alegerea unei teme de culori depinde de scopul aplicației și de publicul țintă. Acesta poate fi atât simplu alb-negru cât și colorat, transmițând astfel ce fel de brand se ascunde în spatele unui site web. Paleta de culori a fost preluată de pe site-ul COLOURlovers³, dar modificată puțin pentru aplicația prezentă a.î. orice tip de utilizator care folosește aplicația să se regăsească (paleta originală de culori precum și cea actualizată și folosită pe site se poate găsi în Anexa 4).
- Grafica – Aceasta poate include logo-uri, poze, imagini (de tipul clipart sau icons),

² <https://balsamiq.com/>

³ <http://www.colourlovers.com/>

toate dezvoltând aspectul design-ului web. Totuși, pentru a fi cât mai user-friendly, acestea trebuie plasate în mod corespunzător în conținutul paginii a.î. să nu aglomereze pagina sau să facă încărcarea acesteia prea lentă. De asemenea, acestea trebuie să se potrivească și cu tema de culori aleasă anterior. Pentru grafica aplicației a fost creat în Photoshop un logo personalizat care poate fi găsit în Anexa 5.

- Font-uri – Utilizarea diferitelor font-uri poate contribui la îmbunătățirea aspectului site-ului. Cele mai multe browsere web pot citi un anumit număr de font-uri cunoscute ca font-uri "web-safe" iar web designeri lucrează în general cu acest grup.
- Conținutul – Conținutul și design-ul paginii pot lucra împreună pentru consolidarea mesajului site-ului prin aspect și text. Textul trebuie să fie întodeauna relevant și util a.î. să nu încurce un cititor și să găsească ușor informația căutată pentru a menține utilizatorul pe site. Conținutul trebuie, de asemenea, să fie optimizat pentru motoarele de căutare, să aibă lungimea corespunzătoare și să încorporeze cuvinte cheie relevante.

În afară de elementele de bază ale unei aplicații web care o fac interesantă și convingătoare din punct de vedere vizual, un site trebuie să ia în considerare utilizatorul final, cel care va folosi aplicația. Ușurința cu care se poate folosi aceasta se poate realiza acordând atenție următorilor factori:

- Navigarea – Arhitectura site-ului, meniurile și butoanele de navigare, trebuie create cu atenție. Scopul acestora este să ajute utilizatorul să se deplaseze prin paginile site-ului fără probleme în găsirea eficientă a informațiilor de care are nevoie.
- Compatibilitatea – Abilitatea design-ului paginii de a se comporta la fel pe diferite browsere și sisteme de operare cu scopul de a spori vizualizările acesteia.
- Multimedia – Prin proiectarea video-urilor sau a stimulilor audio se pot ajuta utilizatorii să înțeleagă mai ușor și mai rapid informațiile prezentate. Acest lucru poate încuraja vizitatorii să petreacă mai mult timp pe pagină.
- Interactiv – Încurajarea utilizatorilor să participe activ și să se implice prin secțiunile de comentarii sau a sondajelor de opinie. Transformare utilizatorilor din simpli vizitatori în clienți oferindu-le posibilitatea înscrierii la newslettere cu adresa de e-mail.

HTML

HyperText Markup Language (HTML) este un limbaj de markup standard folosit pentru crearea paginilor web. Acesta este format din elemente specifice HTML-ului, numite tag-uri, care sunt interpretate de către browser pentru afișarea conținutului unei pagini. Tag-urile sunt scrise între paranteze unghiulare, iar majoritatea lor vin în pereche (ex. `<html> </html>`). Totuși există și câteva excepții, caz în care tag-urile sunt self-closing (ex. ``, `<input>`, `
`).

Principalele tag-uri HTML pe care ar trebui să le conțină orice pagină web sunt:

- `<html> </html>`
- `<head> </head>`
- `<title> </title>`
- `<body> </body>`

HTML5 reprezintă ultima versiune standardizată a HTML-ului lansat în octombrie 2014 și aprobată de World Wide Web Consortium (W3C). Scopul acestuia a fost îmbunătățirea limbajului, asigurând suport pentru multimedia și, în același timp, să îl mențină ușor de înțeles pentru oameni și ușor de interpretat pentru calculatoare și dispozitive (browsere web). Acesta a adăugat noi tag-uri, atât pentru a îmbogăți conținutul semantic al paginilor (ex. `<article>`, `<header>`, `<footer>`, `<aside>`, `<nav>`, `<figure>`) cât și pentru manipularea mai ușoară a conținutului media și graficii (ex. `<video>`, `<audio>`, `<canvas>`, `<object>`).

CSS

Introducere

Cascading Style Sheets (CSS) este un limbaj de stiluri folosit pentru descrierea aspectului și formătărilor unui document web. Acesta este folosit pentru crearea unor pagini web captivante, pentru interfața utilizatorului atât pentru aplicații web cât și pentru aplicații mobile. CSS-ul este conceput în primul rând pentru a permite separarea conținutului documentului de prezentarea

acestui, incluzând elemente care permit aranjarea aspectului, schimbarea culorilor și a fonturilor. Această separare poate îmbunătăți accesul la conținut, oferă mai multă flexibilitate și control și permite accesul mai multor pagini HTML la aceeași formatare prin același fișier .css reducând astfel complexitatea și repetiția în conținutul structural.

LESS

Less este un pre-procesor de CSS care extinde limbajul adăugând noi caracteristici precum variabile, mixing-uri, calcule matematice și funcții. Totodată, spre deosebire de CSS acesta permite scrierea codului sub formă arborescentă. Un exemplu despre cum funcționează mai exact Less-ul se poate vedea în Figura 3.1. unde se regăsește o parte din codul scris pentru icoanele de social media din header-ul paginii aplicației.

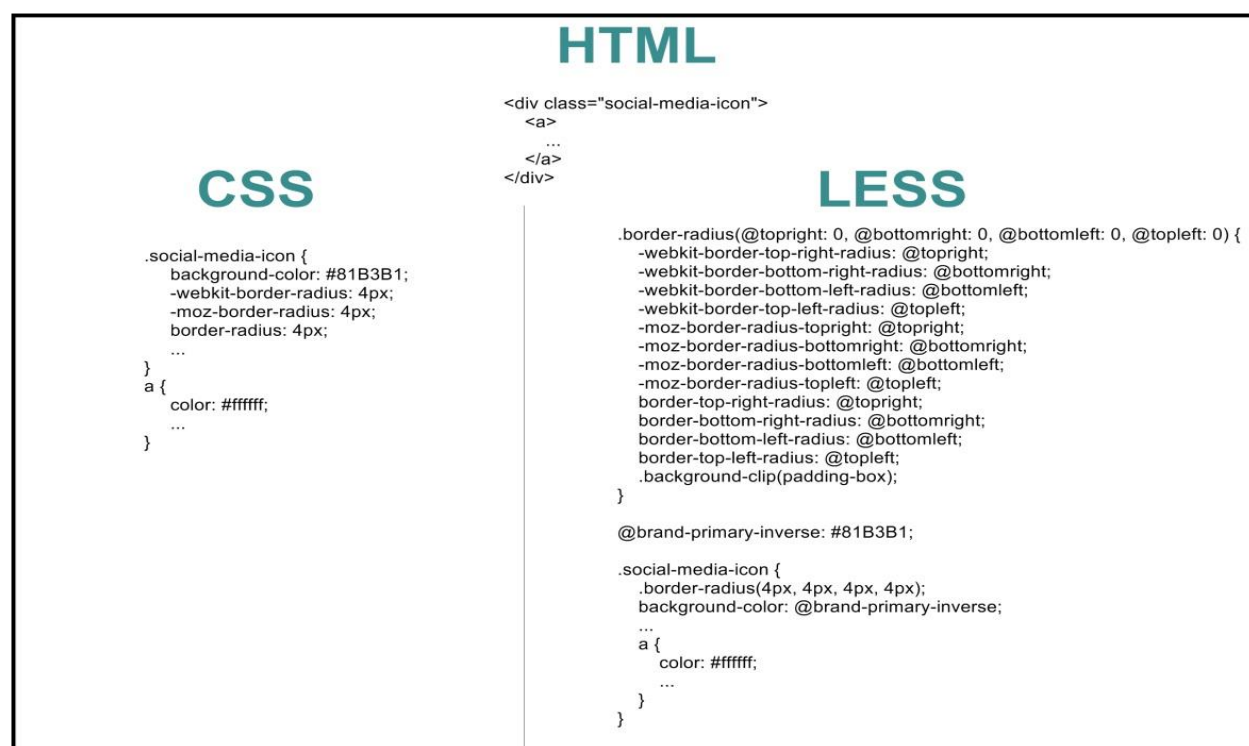


Figura 2.1. CSS vs. LESS

Deși la prima vedere pare că este mai mult de scris în LESS spre deosebire de CSS, pe

parcursul dezvoltării aplicației funcțiile create se pot refolosi pe orice clase, id-uri sau tag-uri avem nevoie. De asemenea, există fișiere întregi cu funcții deja create, acestea putând fi folosite doar apelându-le cu parametri de care avem nevoie fără a fi nevoie de să le mai construim noi.

Pentru instalarea acestuia este necesară instalarea Node.js⁴. Cu ajutorul lui, din linia de comandă, rulăm comenzile pentru instalarea pachetelor de Grunt, de obicei în fișierele de stil ale proiectului (în cazul aplicației acestea sunt instalate în folderul Style):

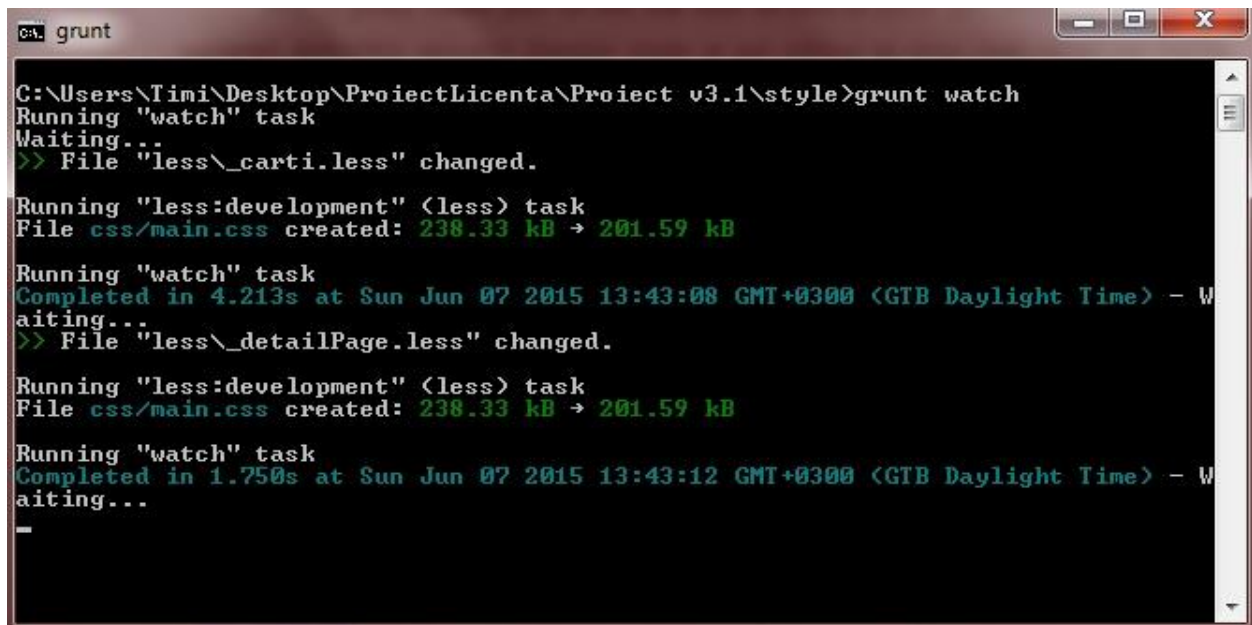
```
$ npm instal -g grunt
```

```
$ npm instal -g grunt-cli
```

Grunt-ul reprezintă un sistem de build care, printre altele, poate fi configurat să detecteze orice schimbări din fișierele de tip “.less” și să realizeze ulterior operațiuni definite de utilizator, respectiv, compilarea fișierelor LESS în CSS. Aceste operațiuni sunt definite mai clar în fișierul Gruntfile.js.

După realizarea pașilor de mai sus se poate rula din nou cmd-ul cu comanda “grunt watch” și se poate începe scrierea fișierelor LESS. După cum se poate observa în Figura 3.2., prin rularea comenzii și la salvarea fișierelor se crează noul fișier CSS (main.css), actualizat cu noile modificări. În cazul în care există cod LESS scris greșit acesta le va semnala și codul nu va fi compilat.

⁴ <https://nodejs.org/>



```
C:\Users\Timi\Desktop\ProiectLicenta\Proiect v3.1\style>grunt watch
Running "watch" task
Waiting...
>> File "less\_carti.less" changed.
Running "less:development" <less> task
File css/main.css created: 238.33 kB → 201.59 kB
Running "watch" task
Completed in 4.213s at Sun Jun 07 2015 13:43:08 GMT+0300 (GTB Daylight Time) - W
aiting...
>> File "less\_detailPage.less" changed.
Running "less:development" <less> task
File css/main.css created: 238.33 kB → 201.59 kB
Running "watch" task
Completed in 1.750s at Sun Jun 07 2015 13:43:12 GMT+0300 (GTB Daylight Time) - W
aiting...
-
```

Figura 3.2. Rularea Grunt-ului

Bootstrap

Inițial creat de către doi angajați la Twitter (un designer și un dezvoltator), Bootstrap a devenit rapid unul dintre cele mai populare framework-uri de front-end, open source. Numit inițial Twitter Blueprint, acesta este o combinație de HTML, CSS și Javascript care ajută la crearea componentelor interfeței utilizatorului.

Printre avantajele folosirii Bootstrap-ului se enumără:

1. Se poate personaliza – După preferințe, se poate adapta în conformitate cu specificațiile proiectului. Dezvoltatorii au capacitatea de a alege caracteristicile necesare, iar restul poate fi ignorat.
2. Este consistent – Acesta a fost fondat după acest principiu, permițând mai multor dezvoltatori care lucrează la același proiect să aibă aceleași rezultate, indiferent de platforma folosită iar interfața finală să arate la fel pe toate browserele.
3. Asigură suport – Bootstrap are o comunitate imensă în spatele lui a.î. pentru orice fel de probleme există soluții. În plus, acesta este în curs de actualizare continuă și în același timp, dezvoltat și găzduit pe GitHub.

4. Este responsive – Indiferent de dispozitivul folosit (de la desktop, la iPad și mobile), Bootstrap-ul se adapteaza automat în funcție de platformă.

Fiind un framework de tip mobile first, Bootstrap (în special Bootstrap 3) se bazează pe un sistem fluid de grid-uri care pot ajunge până la 12 coloane, pe măsură ce dimensiunea ecranului se mărește. Acesta include clase predefinite pentru un acces cât mai ușor la layout-ul paginii precum și mixin-uri pentru generarea layout-urilor semantice.

Grid-urile sunt formate printr-o serie de rânduri și coloane unde este inclus conținutul.

- Rândurile sunt incluse în clasele “.container” sau “.container-fluid”;
- Rândurile sunt folosite pentru crearea grupurilor orizontale de coloane;
- În interiorul coloanelor se plasează conținutul, iar acestea trebuie să stea în interiorul rândurilor
- Există clase predefinite pentru rânduri “.row” și, respectiv, coloane “.col-md-*” (unde, în loc de “*” se pune numărul de coloane ce se alocă din cele 12 existente pe un rând).

Media queries se folosesc în fișierele LESS pentru determinarea punctelor de “rupere” (breakpoints) în sistemul de grid-uri. Un exemplu mai sugestiv se poate observa în Figura 3.3.

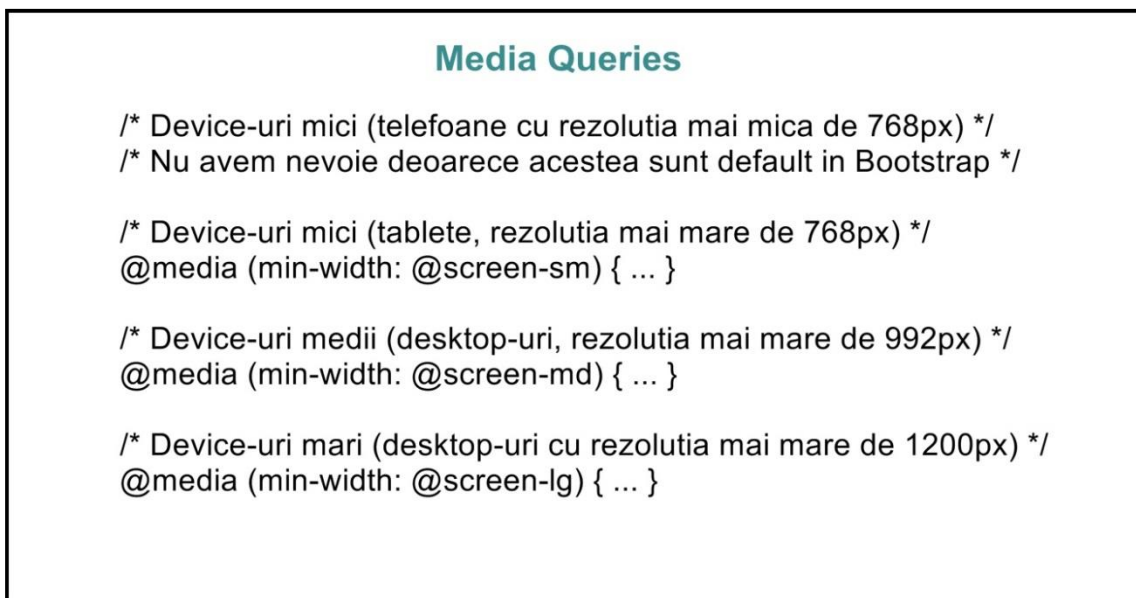


Figura 3.3. Media Queries pe diferite rezoluții de ecran

Folosit împreună cu LESS, design-ul paginilor este contruit de la început responsive a.î. pagina va arăta similar atât pe desktop cât și pe tablete și telefoane mobile. Un exemplu ar fi header-ul paginii care se adaptează în funcție de rezoluția ecranului. Diferențele rezultate se pot observa în figurile de mai jos.



Figura 3.4. Header la rezoluție medie-mare



Figura 3.5. Header la rezoluție mică cu meniul închis

Font Awesome

Font awesome reprezintă un set de instrumente de font-uri și imagini bazate pe CSS și LESS. Așa cum se poate observa în Figura 3.3 acestea pot fi folosite atât în codul HTML cât și în LESS în pseudo-elementele `:before` și `:after`. Exemplul de mai jos descrie construirea

imaginilor de social media din header-ul paginii aplicației. În stânga este partea de HTML, cum de fapt au fost și create elementele, iar în partea dreaptă se poate observa o alternativă, realizată în LESS.

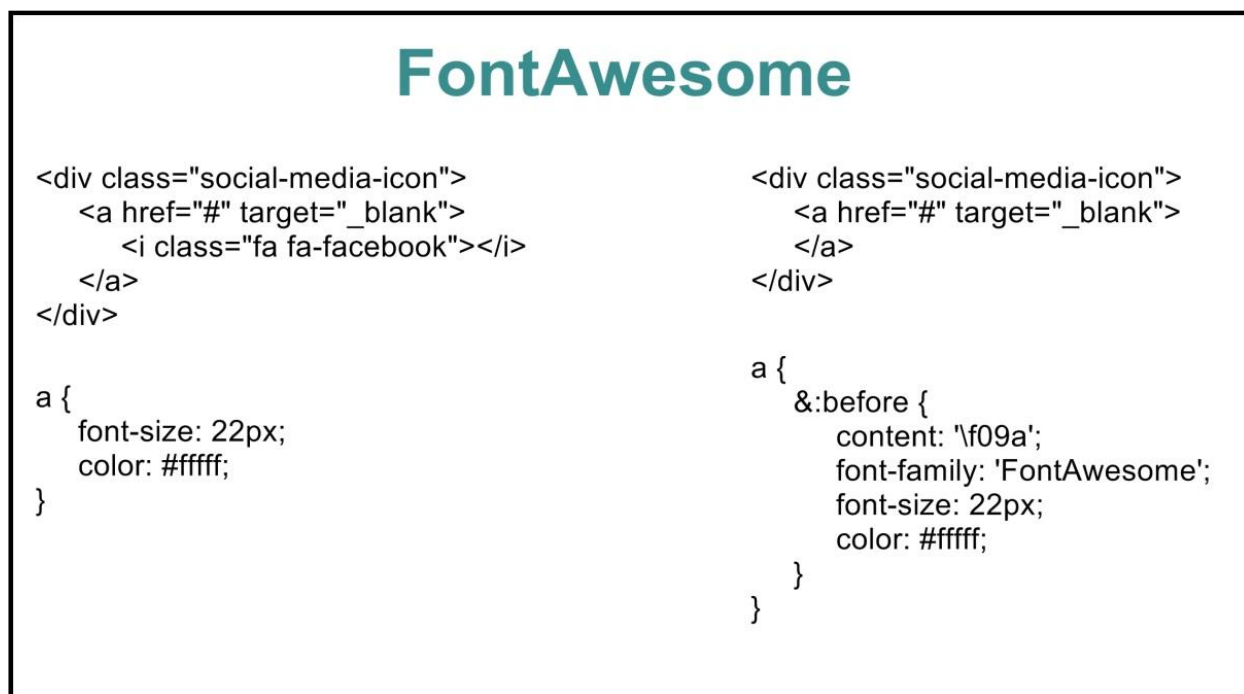


Figura 3.6. Folosirea imaginilor FontAwesome in HTML vs. CSS/LESS

De asemenea, au mai fost folosite imagini în crearea unui aspect plăcut și compact al site-ului la modalele de login și cel pentru schimbarea parolei precum și pentru pagina de înregistrare. Cele trei iconițe folosite se pot observa în figura următoare.

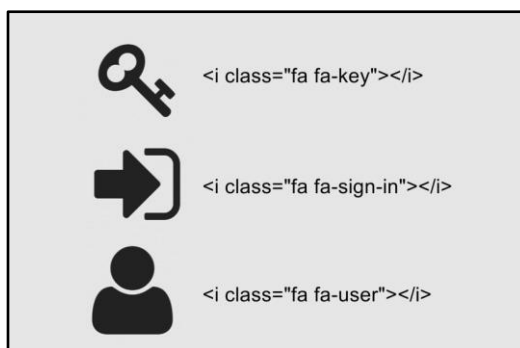


Figure 4.7. Imagini FontAwesome

JavaScript

Introducere

JavaScript (adesea prescurtat JS), este un limbaj de programare dinamic. Acesta a fost creat în 10 zile în mai 1995 de către Brendan Eich (Netscape). Numit inițial Mocha, a fost lansat în septembrie 1995 sub numele oficial LiveScript, redenumit apoi în decembrie, același an, JavaScript. Este cel mai des folosit în aplicațiile web, ceea ce permite rularea codului de către browser. JavaScript este clasificat ca fiind un limbaj de programare imperativ, orientat pe obiect.

În ciuda similarităților de sintaxa și a numelui, între JavaScript și Java nu există absolut nici o legătură.

JQuery

JQuery este o librărie open-source Javascript creată pentru a simplifica și a îmbunătăți codul rulat de tip client-side, pentru ușurarea proceselor precum traversarea DOM-ului (Document Object Model) în HTML, pentru crearea de animații, tratarea evenimentelor și pentru a dezvoltarea aplicațiilor folosind cereri tip AJAX.

În realizarea aplicației s-au folosit diferite framework-uri jQuery pentru a permite o utilizare mai ușoară a aplicației:

1. FontAwesome Star Rating

Pluggin-ul a fost folosit în pagina de detaliu a cărților pentru afișarea rating-ului fiecărei cărți (o notă între 1 și 5 sub forma unor imagini de stele colorate din setul de instrumente FontAwesome) care, de asemenea, permite și utilizatorului autentificat să voteze o carte (prin trecerea mouse-ului peste stele). În caz că un utilizator neautentificat încearcă să voteze o carte, va fi afișat un mesaj de eroare.

Pentru folosirea acestuia, este necesar un simplu container care să aibă id-ul “star” și proprietatea “data-rating” în care va primi media notelor rotunjită.

Container-ul pentru StarRating

```
<div id="stars" class="starr" data-rating='<%# Eval("MedieNote") %>'></div>
```

Figure 3.8. Folosirea StarRating-ului

2. jCarousel

Acesta este folosit pentru controlarea listelor de elemente pe orizontală sau pe verticală și oferă un set complet și flexibil de instrumente pentru navigarea în orice conținut bazat pe HTML într-un mod de carusel. Acesta a fost folosit atât în pagina de detaliu al fiecărei cărți, unde caruselul afișează alte 10 rezultate din aceeași categorie, cât și în pagina de profil a utilizatorului unde există 3 carusele pentru fiecare din categoriile: cărți favorite, cărți de citit și cărți citite.

În funcție de rezoluția ecranului (aceasta depinzând de dispozitivul folosit), caruselul se comportă responsive și poate afișa de la maximum de cărți (toate cele 10) până la o singură carte sau două. În cazul în care nu sunt afișate toate cărțile pentru vizualizarea acestora sunt necesare cele două săgeți din dreapta și din stânga container-ului. Detaliile acestui aspect se pot observa în figurile de mai jos.



Figure 3.9. Carusel la rezoluție mare

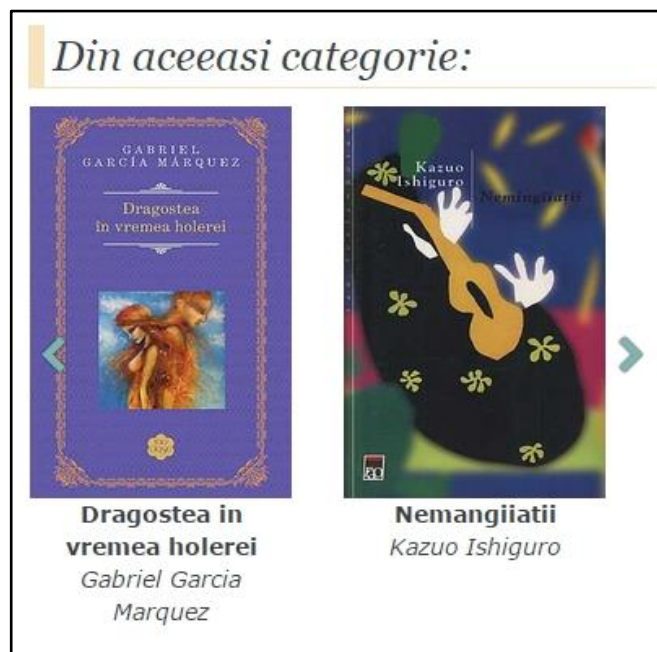


Figure 3.10. Carusel la rezoluție mica

3. Bootstrap Notify

Pornind de la blocurile de mesaje standard tip “alert” din Bootstrap, acest plugin le transformă în notificări dinamice. Acestea au fost folosite pentru trimiterea mesajelor de succes sau de eroare către utilizator. Apelarea acestora se face în C#, apelând o funcție ce primește doi parametri. Primul îl reprezintă mesajul text ce trebuie afișat (acesta diferă de la un eveniment la altul) și tipul de mesaj (poate fi de succes sau de eroare). În funcție de cele două tipuri, blocurile afișate în pagină au culori diferite (verde pentru succes și roșu pentru erori). Un exemplu de blog se poate vedea în figura 3.11.

De asemenea, plugin-ul este ușor customizabil, permițând dezvoltatorului să afișeze mesajele în aproape orice modalitate posibilă (începând de la poziția unde va fi amplasat în fereastra de browser, până la timpul afișării acestuia). Pentru aplicație s-au folosit câteva setări de bază ce se pot regăsi în codul din Figura 3.12.



Figure 3.11. Afişarea unui mesaj de succes

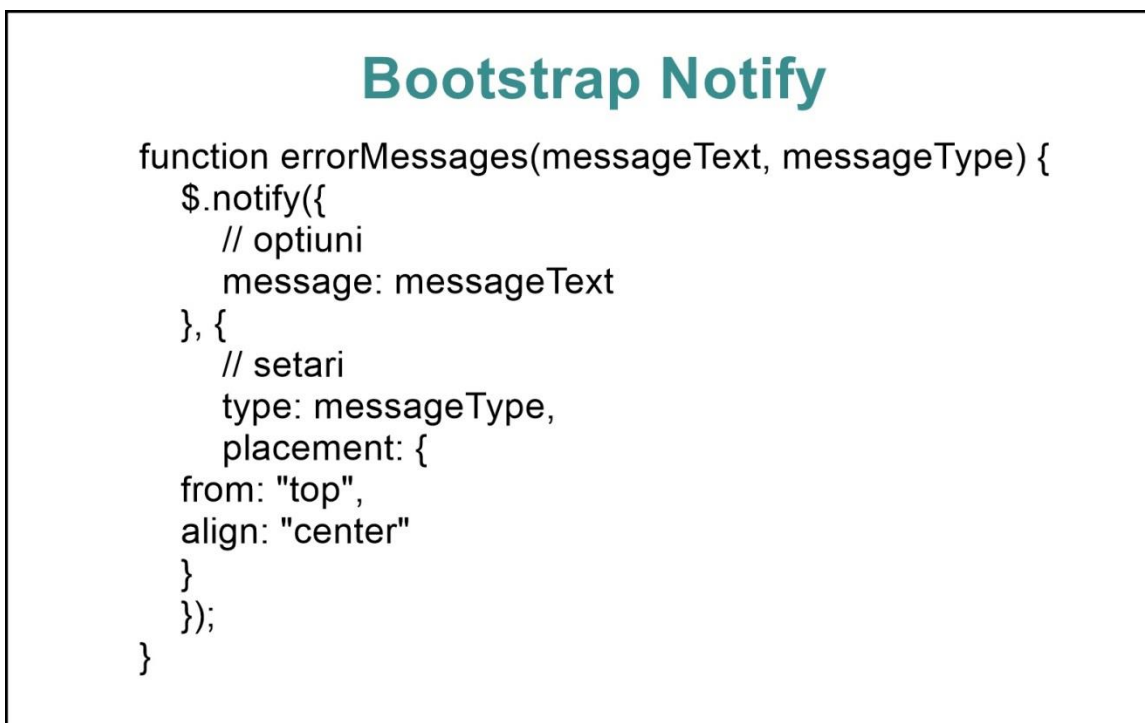


Figura 3.12. Funcția de afișare a erorilor din Bootstrap Notify

4. jQCloud

Acesta este un plugin de jQuery cu ajutorul căruia se pot construi ”word clouds” sau ”tag clouds”. Acestea se creează dinamic doar din HTML și CSS. Tag cloud-ul a fost creat pe pagina principală a aplicației, tag-urile însumând atât categoriile de cărți existente, cât și numele de autori din baza de date. Dimensiunea font-ului pentru tag-uri este dată de numărul de cărți din categoria respectivă din baza de date, respectiv numărul de cărți scrise de autor (acestea putând fi ulterior modificate din CSS).

Pentru apelarea funcției este necesar minim un parametru, acesta fiind obiectul care ține datele afișate (numele autorului sau categoria, numărul de apariții în baza de date și link-ul către pagina de rezultate cu parametrul selectat). În afară de acesta, plugin-ul se mai poate personaliza cu diferitele opțiuni existente, precum culori, dimensiuni, formă, autoResize, etc. În Figura 3.8. se regăsește obiectul creat dinamic cu date din baza de date precum și funcția apelată în momentul deschiderii paginii.

Obiectul creat si apelarea functiei de creare a Tag Cloud-ului

```
[[ { "text" : "Arta si Design",
    "weight" : 8,
    "link" : "Search.aspx?q=Arta si Design"},

  { "text" : "Biografii si Memorii",
    "weight" : 19,
    "link" : "Search.aspx?q=Biografii si Memorii"}, ...]

$(document).ready(function () {

    var inputString = $(".object").val();

    var object = JSON.parse(inputString);

    $('#tagCloud').jQCloud(object, {
        colors: ["#413E4A", "#81B3B1"],
        autoResize: true
    });
});
```

Figura 3.12. Apelarea plugin-ului care creează Tag Cloud-ul

Nu stii ce sa mai cauti? Te ajutam noi!

Figure 3.13. Tag Cloud format

CONCLUZII

Scopul lucrării de față este de a realiza o incursiune în domeniul informaticii și se bazează pe studierea și aprofundarea tehnologiilor prezentat, de la backend, C#, ASP.NET, baze de date, până la interfața aplicației HTML, CS și Javascript (jQuery). Toate acestea ajută la construirea unei aplicații de evidență a cărților, o bibliotecă virtuală pentru utilizatori.

Ideea generală a aplicației este crearea unei comunități unice, locale, doar pe România, în jurul cărților și, totodată, permiterea utilizatorilor să-și creeze un sistem de organizare (o bibliotecă online) unde își pot adăuga cărțile pe diferite categorii. Utilizatorii au posibilitatea să-și creeze un profil și să-l personalizeze după bunul plac. Design-ul aplicației se dorește a fi cât mai user-friendly, facilitând accesul la informațiile căutate în cel mai ușor mod. De asemenea, aplicația este responsive, asta însemnând că indiferent de dispozitivul folosit (de la desktop până la tablete și telefoane mobile) site-ul va arăta similar, potrivit pentru fiecare rezoluție de ecran.

Lucrarea este structurată în trei capitole. În primul capitol este realizată o scurtă prezentare despre aplicațiile web în general și avantajele folosirii acestora. De asemenea, s-a făcut o comparație între #LaOCarte și alte site-uri similare, printre care blogurile și site-urile editurilor românești, precum și binecunoscutul goodreads. Tot în primul capitol s-a făcut o trecere în revistă a ceea ce înseamnă sistem de recomandări și cum s-a implementat un astfel de sistem strict pe aplicația web prezentată aici precum și rolurile utilizatorilor cu exemplificări pentru toate cele trei tipuri (utilizator neautentificat, utilizatori înregistrați – Admin și User).

În al doilea capitol s-au analizat tehnologiile folosite în partea din spate (back-end), pentru funcționalitatea aplicației cu exemple concrete pentru ceea ce s-a folosit pe site-ul #LaOCarte. În consecință, platforma ASP.NET împreună cu C# are foarte multe de oferit dezvoltatorilor iar comunitatea formată în jurul acestora și documentațiile existente au soluții implementate pentru absolut orice probleme ar exista.

Al treilea capitol a pus accentul pe interfața și design-ul aplicației (front-end). Site-ul reușește să fie accesibil oricui și are un design modern și atractiv. Acesta dispune de o structură bine organizată, facilitând stilizarea cât mai ușoară a elementelor cu ajutorul compilatorului

LESS și a diferitelor framework-uri.

În concluzie, dezvoltarea aplicației #LaOCarte a fost în același timp o provocare și o plăcere. S-a lucrat cu spor și drag la ea și pe viitor se dorește personalizarea și dezvoltarea continuă a acesteia.

BIBLIOGRAFIE

1. Sitepoint 2010, The Principles of Beautiful Web Design by Jason Beaird, Canada
2. *** <http://www.asp.net/>
3. *** <http://www.codeproject.com/>
4. *** <http://bootsnipp.com/>
5. *** <http://www.acunetix.com/>
6. *** <https://msdn.microsoft.com/>
7. *** <http://stackoverflow.com/>
8. *** <https://webthinker.wordpress.com/>
9. *** <http://lesscss.org/>
10. *** <http://getbootstrap.com/>
11. *** <http://fortawesome.github.io/Font-Awesome/>
12. *** <http://www.webopedia.com/>
13. *** <https://scotch.io/>
14. *** <https://www.treefrog.ca/>
15. *** <http://sorgalla.com/jcarousel/>
16. *** <http://mistic100.github.io/jQCloud/>
17. *** <http://www.w3schools.com/>
18. *** <http://www.allaboutcookies.org/>
19. *** <http://geekswithblogs.net/>
20. *** <http://en.wikipedia.org/wiki/Wiki>
21. *** <http://www.tutorialspoint.com/>
22. *** <http://csharp.net-tutorials.com/>

23. *** <http://www.ibm.com/>
24. *** <https://www.coursera.org/>