

UNIVERSITATEA BABEȘ-BOLYAI CLUJ-NAPOCA
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ
SPECIALIZAREA INFORMATICĂ

LUCRARE DE LICENȚĂ
Aplicații web. Studiu de caz

Conducător științific:
Conf. Univ. Dr. Grigoreta COJOCAR

Absolvent:
Radu – Liviu MUNTEAN

2015

Cuprins

1. Introducere	2
2. Concepte de bază	3
2.1. World Wide Web	3
2.2. Document Object Model.....	4
2.3. Hyper Text Transfer Protocol	4
2.4. Modul de acces la resurse	7
3. Tehnologii client-side	9
3.1. HTML	9
3.2. CSS	10
3.3. JavaScript.....	14
3.4. Framework-ul Bootstrap	16
4. Tehnologii server-side	19
4.1. Java Servlets.....	19
4.2. Spring MVC.....	21
4.3. JavaServlet Pages.....	23
4.4 Arhitectura aplicației web	23
5. Protocolul WebSocket	25
6. Aplicația practică	29
7. Concluzii	37
Bibliografie	38

1. Introducere

Această lucrare are ca scop oferirea unui mod simplu și accesibil de a-ți îmbogăți cunoștințele în domeniul Sfintelor Scripturi și care poate fi folosită și în alte domenii. Bibliardar va fi un joc asemănător cu bine cunoscuta emisiunea televizată „Vrei să fii milionar?”, cu posibilitatea de a-l folosi pe oricare dintre următoarele dispozitive: desktop, tabletă și telefon mobil.

Lucrarea se constituie în șase capitole, excluzând introducerea, pe parcursul cărora se analizează avantajele și provocările apărute în procesul de implementare al proiectului.

În primul capitol sunt prezentate concepte de bază ale programării web. În al doilea capitol sunt descrise tehnologii client-side. În al treilea capitol sunt prezentate tehnologii server-side, folosite în aplicația practică. În al patrulea capitol sunt descrise avantajele și dezavantajele folosirii protocolului WebSocket. În al cincilea capitol este descris modul de utilizare și funcționare al aplicației practice.

Am ales această temă deoarece consider că un astfel de mod, mai interactiv, ar fi o nouă treaptă în educație stimulând generațiile tinere să se documenteze și cultiveze intelectual într-o formă mai atractivă.

2. Concepte de bază

2.1. World Wide Web

World Wide Web este un sistem format din clienți și servere distribuite. Serverul acceptă cereri din partea clientului pentru un anumit document. Apoi, acesta poate să returneze documentele cerute de client, dar poate și să genereze aceste fișiere. Mai mult, serverul poate primi cereri de stocare de documente.

Browserul este principalul mod prin care un client interacționează cu un server în World Wide Web. Acesta este responsabil de prezentarea și preluarea informațiilor primite de la server și de realizarea de cereri către server. Odată ce browserul primește un document, scris în HTML (HyperText Markup Language) sau în alt limbaj, acesta interpretează și afișează codul din document.

În World Wide Web toate informațiile sunt reprezentate ca documente. Există mai multe moduri prin care un document poate fi reprezentat. Unele sunt simple fișiere text ASCII, în timp ce altele sunt reprezentate ca o colecție de scripturi care sunt executate automat atunci când documentul este downloadat de către un browser.

Mai mult, un document poate conține referințe către alte documente. Aceste referințe sunt numite *hyperlinks*. Când un document este afișat într-un browser, *hyperlink*urile pot fi afișate explicit utilizatorului. Selectarea unui *hyperlink* de către un utilizator duce automat la crearea unei cereri adresată serverului pentru documentul referit de *hyperlink*. Apoi serverul returnează acel document, care este apoi interpretat și afișat de către browser. Noul document poate să înlocuiască documentul curent din browser, sau să fie afișat într-o altă fereastră.

De obicei, documentele din World Wide Web sunt prezentate ca fișiere HTML. Cu ajutorul HTML, documentul este structurat. Acesta este împărțit în secțiunea de heading și conținutul principal. Mai mult, HTML distinge mai multe elemente de structură, cum ar fi liste, tabele, sau forme. În afară de elementele structurale, HTML distinge mai multe elemente care indică browserului cum trebuie prezentat documentul.

Atunci când un document este parsat, el este stocat intern sub forma unui arbore cu rădăcină, numit „parse tree”. Intern, toate nodurile arborelui reprezintă un element din document. Mai mult, toate nodurile implementează o interfață standard care conține metode pentru accesarea conținutului său. Acest standard se numește DOM (Document Object Model).

2.2. Document Object Model

DOM (Document Object Model) este o interfață de programare folosit pentru documentele XML sau HTML. Cu ajutorul DOM este definit modul prin care un document este manipulat și accesat. Mai mult, acesta definește și structura logică a documentelor. DOM este împărțit în trei părți: Core, HTML și XML.

Core DOM este o interfață de bază, care oferă obiecte ce pot reprezenta orice document structurat. Această interfață conține un design minimal și compact pentru manipularea conținutului unui document (Robie).

În schimb, specificările DOM pentru HTML și XML oferă interfețe adiționale, care sunt folosite împreună cu Core DOM pentru a oferi un mod mai bun de accesare a documentelor.

Cu ajutorul DOM, se pot crea documente, cărora le pot fi accesate și modificate elementele din structură. Orice element dintr-un document XML sau HTML poate fi adăugat, șters sau modificat cu ajutorul DOM.

2.3. HyperText Transfer Protocol

HTTP (HyperText Transfer Protocol) este protocolul pe care se bazează toate comunicările din World Wide Web dintre clienți și servere. Unul din principalele avantaje ale comunicării prin HTTP este faptul că nu are un concept de conexiune deschisă și nu are nevoie ca serverul să mențină informații despre clienții săi (Chang).

HTTP este bazat pe protocolul de comunicare TCP. Astfel, atunci când un client face o cerere către un server, acesta deschide o conexiune TCP către server, care, apoi, este folosită pentru a primi răspuns. Deoarece HTTP folosește TCP ca protocol de comunicare, este garantat faptul că informațiile ajung corect și sigur la destinație. Astfel, HTTP nu este nevoit să aibă grijă de pierderi de cereri sau răspunsuri.

Protocolul HTTP are următoarele tipuri de mesaje:

- GET;
- POST;
- PUT;
- DELETE;
- HEAD;

Dintre acestea, GET și POST sunt cele mai folosite și sunt singurele metode care sunt suportate de către toate serverele web. Diferența principală dintre aceste două metode este modul prin care sunt reprezentați parametrii.

În cazul metodei GET, parametrii fac parte din URI (Uniform Resource Identifier). Astfel, este restricționat numărul parametrilor, deoarece URI este restricționat la un număr limitat de caractere. Prin această metodă pot fi trimise numai caractere de tip ASCII.

În cazul metodei POST, parametrii sunt incluși în headerul HTTP al cererii. Astfel, metoda POST nu are restricții din punctul de vedere al tipului sau al numărului parametrilor.

Dezavantaje ale metodei GET:

- parametrii, făcând parte din URI, pot fi ușor observați de către utilizatori, și, astfel, nu pot fi trimiși parametri care conțin informații importante, cum ar fi parole, astfel GET este mai puțin sigură.
- nu poate fi trimisă o cantitate mare de date prin parametrii.

Avantaje ale metodei GET:

- deoarece parametrii unui form sunt conținuți în URL, acest URL poate fi folosit în mod direct, fără a mai fi nevoie ca utilizatorii să treacă prin procesul de completare a form-ului.

Dezavantaje ale metodei POST:

- deoarece parametrii sunt incluși în corpul cererii, nu poate fi salvată completarea unui form.

Avantaje ale metodei POST:

- deoarece parametrii nu fac parte din URL, se pot transmite cantități mai mari de date.
- deoarece parametrii nu fac parte din URL, se pot trimite parametrii care conțin informații importante.

Un răspuns HTTP este format dintr-o secțiune header și un conținut. Secțiunea de header conține informații legate de modul în care browserul trebuie să interpreteze acest conținut. Un răspuns HTTP conține și un cod care reprezintă modul în care serverul a răspuns la cererea trimisă. Cele mai întâlnite coduri sunt:

- „200 OK “ indică faptul că cererea s-a executat cu succes;
- „304 Not Modified ” indică faptul că resursa asupra căreia s-a făcut cererea nu s-a modificat față de ultima oară când s-a făcut cererea și că browserul trebuia să încarce acea resursă din cache. Acest cod poate să apară numai în cazul în care browserul face o cerere de tip GET;
- „404 Not Found ” indică faptul că serverul nu poate găsi resursa cerută;
- „401 Authorization Required ” indică faptul că resursa cerută este protejată și are nevoie de o autorizare pentru a putea fi accesată;
- „403 Forbidden ” cererea este înțeleasă, dar a fost refuzată sau accesul nu este permis.

- „ 500 Internal Error ” indică faptul că serverul a întâlnit o problemă în timp ce procesa cererea.

Protocolul HTTPS (HTTP Secure) este o versiune sigură a protocolului HTTP. Acesta este rezultatul adăugării capabilităților protocolului SSL (Secure Sockets Layer) la protocolul standard HTTP. În cazul protocolului HTTPS, protocolul conținut în URL va fi de tip „https://”.

2.4. Modul de acces la resurse

Un URI (Uniform Resource Identifier) este format din secvențe de caractere, cu ajutorul cărora este identificată orice resursă prin locație sau nume sau ambele în Internet.

Un URL (Uniform Resource Locator) este format din secvențe de caractere (litere, cifre și caractere speciale) care reprezintă locația unei resurse aflate în World Wide Web. Un URL este un URI care, în plus, față de a identifica o resursă, prezintă și modul prin care trebuie accesată acea resursă.

Forma semantică a unui URL este:

<protocol>://<user>:<password>@<host>:<port>/<url-path>

Aceasta conține:

- <protocol> reprezintă modul în care browserul trebuie să comunice cu serverul web atunci când primește sau trimite date.
- <host> reprezintă un nume de domeniu unic, care reprezintă un mod unic prin care se identifică un site web.
- <port> reprezintă portul pe care serverul așteaptă să primească cereri. Dacă nu este specificat, atunci portul va fi setat automat ca 80 dacă protocolul este HTTP și 443 dacă protocolul este HTTPS
- <url-path> reprezintă calea locală a unei resurse de pe server

Mai mult, dacă URL-ul este folosit prin protocolul HTTP prin metoda GET, atunci acesta mai poate conține, după <url-path>, și unul sau mai mulți parametrii despărțiți de caracterul “&”.

Un URN (Uniform Resource Name) este un URI care are structura schemei URN:

$$\langle \text{URN} \rangle ::= \text{"urn:"} \langle \text{NID} \rangle \text{":"} \langle \text{NSS} \rangle$$

În acest caz NID este Namespace Identifier, iar NSS este Namespace Specific String.

3. Tehnologii client-side

3.1. HTML

HTML (HyperText Markup Language) este principalul mod prin care documentele sunt prezentate în World Wide Web. Acesta este format din două părți: conținut efectiv și un set de instrucțiuni care indică calculatorului modul în care trebuie să afișeze acest conținut (Brooks, 2007).

Un document HTML simplu este format din cel puțin patru seturi de elemente:

```
<html> ... </html>

<head> ... </head>

<title> ... </title>

<body> ... </body>
```

Aceste elemente definesc părțile esențiale ale unui document HTML: documentul propriu-zis, o secțiune de heading, un titlu, și corpul său.

Fiecare element este definit de două tag-uri, unul de început și unul de sfârșit. Aceste tag-uri sunt întotdeauna în paranteze unghiulare. În plus, tag-urilor de sfârșit li se adaugă la început caracterul „/”. Însă, există și elemente care au un singur tag care se închide prin caracterele „/>”.

În afară de elemente definite prin tag-uri, un document HTML mai trebuie să conțină și un element DOCTYPE pus la începutul documentului, care indică browserelor tipul acestui document, mai exact versiunea acestuia, pentru ca acesta să poată interpreta corect documentul. De exemplu, elementul DOCTYPE care indică o versiune HTML 4.0 Transitional a documentului arată astfel:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

Elementul DOCTYPE nu trebuie, însă, specificat manual de fiecare dată, deoarece majoritatea mediilor de programare îl includ automat.

Începând cu versiunea HTML5, sintaxa elementului DOCTYPE a fost simplificată. Astfel, pentru versiunea HTML 5 sintaxa acestuia este:

```
<!doctype html>
```

3.2. CSS

În timp ce HTML este un limbaj structural ce reprezintă nivelul de conținut al unui document, CSS (Cascading Style Sheet) reprezintă nivelul de prezentare a acestui document.

Deși anumite versiuni de XHTML și HTML conțin elemente de prezentare al documentului, cum ar fi sau <i>, folosirea acestora nu este recomandată, deoarece, nivelul de prezentare al documentului este amestecat cu nivelul de conținut al acestuia, eliminând, astfel, avantajele pe care le oferă separarea acestora (Meyer, 2007).

În afară de separarea dintre nivelul de conținut și cel de prezentare, alte avantaje ale folosirii CSS sunt:

- Reutilizarea codului, deoarece un fișier CSS poate fi reutilizat în mai multe fișiere HTML.
- Deoarece fișierul CSS este salvat de browser în cache-ul local după primul request, acesta trebuie downloadat o singură dată, o dată pentru fiecare fișier HTML. Astfel, se economisește spațiul de bandwidth.
- Nivelul de prezentare este mai ușor de întreținut de către programatori.

Există trei moduri prin care se pot specifica reguli CSS de stilizare pentru elemente într-un fișier HTML:

- cu ajutorul elementului

```
<link rel="stylesheet" type="text/css" href="/style.css" media="screen">
```

ce trebuie folosit în interiorul elementului „head”, astfel este invocat un fișier extern CSS ce conține reguli de stilizare.

- Cu ajutorul elementului <style>, ce trebuie să se afle în interiorul elementului <head>, prin care se definesc în mod direct elementele de stilizare. În acest caz regulile de stilizare se aplică numai documentului în care au fost definite. Exemplu:

```
<style type="text/css" media="screen, projection">
```

```
h1 { color: #777; }
```

```
h2 { font-weight: bold; }
```

```
</style>
```

- Folosind stiluri inline. Acest lucru se realizează adăugând atributul „style” unui element în care sunt specificate una sau mai multe elemente de stilizare separate prin „;”. În acest caz regulile de stilizare se aplică numai elementului în care au fost specificate.

Reguli CSS

Regulile într-un fișier CSS reprezintă formulări care indică browserului modul în care un element dintr-un fișier HTML trebuie afișat. O regulă este formată dintr-un selector și un set de proprietăți de stilizare.

Selectoarele sunt structuri care sunt folosite pentru a selecta unul sau mai multe elemente, pentru ca apoi acestor elemente să li se poată aplica proprietățile de stilizare.

Structura regulilor este:

```
Selector  
{  
    Proprietate1:valoare1;  
    Proprietate2:valoare2;  
    .....  
}
```

Tipuri de selectoare

Există mai multe tipuri de selectoare:

- Selectoare de element
- Selectoare de clase
- Selectoare de ID
- Selectoare de grup
- Selectorul universal

Selectorul de element este cel mai folosit selector. Acest tip de selector este format din nume de elemente dintr-un document HTML. Astfel, regulile atribuite acestui selector vor fi aplicate tuturor elementelor cu acel nume din document.

De exemplu:

```
h1 { color: red; }
```

indică faptul că culoarea textului din fiecare element h1 din document este roșie.

Selectorul de clasă este folosit pentru a aplica un set de reguli mai multor elemente, care pot fi de tipuri diferite între ele, dar care sunt setate la o clasă comună prin atributul „class”. De exemplu, considerând elementele:

```
<h1 class="class1">Heading</h1>
```

```
<p class="class1">Paragraph</p>
```

Putem specifica culoarea textului ambelor elemente prin regula

```
.class1{ color: red; }
```

Spre deosebire de selectorul de clasă, selectorul de ID aplică proprietățile unui element unic care are setat un atribut „id” după care este identificat.

De exemplu, considerând elementele:

```
<h1 id="class1">Heading</h1>
```

```
<p id=" class2">Paragraph</p>
```

și regula

```
#class2{ color: red; }
```

Atunci numai culoarea textului din elementul „p” va deveni roșie.

Selectorul universal este folosit pentru a selecta toate elementele din document. Acesta este definit prin intermediul caracterului „*”. De exemplu, aplicând regula:

```
{ color: red; }
```

Culoarea tuturor textelor din toate elementele documentului devine roșie.

3.3. JavaScript

JavaScript este un limbaj de scripting orientat pe obiect care are o sintaxă de bază asemănătoare limbajului C. Acesta este folosit în aplicațiile web pentru a crea pagini web dinamice. Mai mult, acesta este folosit în aplicațiile client-side pentru a oferi un mod de comunicare a clientului cu serverul (Flanagan, 2002).

Deoarece JavaScript este un limbaj client-side, fișierele JavaScript sunt downloadate de către browser înainte ca codul acestuia să poată fi executat.

JavaScript este un limbaj care poate să reacționeze la anumite evenimente declanșate de către utilizator, cu ajutorul unor funcții. Cele mai importante și folosite astfel de funcții sunt:

- `onClick`, executată atunci când utilizatorul apasă un anumit element
- `onKeyDown`, executată atunci când utilizatorul apasă un buton
- `onChange`, executată atunci când conținutul unui element s-a modificat

În JavaScript o variabilă este declarată cu ajutorul cuvântului rezervat „`var`”. O astfel de variabilă nu poate să fie un cuvânt rezervat și nu poate să înceapă cu un număr sau unul din caracterele „`$`” sau „`_`”.

Folosirea limbajului JavaScript se face prin intermediul elementului `<script>`. Acest element poate să conțină instrucțiuni de cod JavaScript sau poate să conțină un link către un fișier JavaScript aflat pe server (Saternos, 2014).

În continuare sunt prezentate mai multe tipuri de instrucțiuni și operatori JavaScript:

Instrucțiunea IF

În JavaScript instrucțiunea IF are următoarea sintaxă

```
if (expresie) { // set de instrucțiuni }
```

Această instrucțiune evaluează expresia, iar dacă rezultatul evaluării este valoarea booleană `true`, atunci setul de instrucțiuni este executat.

Operatorii ternari

În JavaScript operatorii ternari reprezintă o prescurtare a instrucțiunilor if/else. Aceștia au sintaxa:

```
var variabilă=(condiție de testat) ? (condiție adevărată) : (condiție falsă);
```

Astfel, dacă condiția de testat este adevărată, atunci variabilei îi este atribuită valoarea variabilei sau a expresiei reprezentată de „(condiție adevărată)”. Dacă, însă, rezultatul condiției de testat este fals, atunci variabilei îi este atribuită valoarea variabilei sau a expresiei reprezentată de „(condiție falsă)”.

Instrucțiunea FOR

În JavaScript instrucțiunea FOR are sintaxa:

```
for (var i={valoare_inițială} ; ({expresie}) ; {atribuire} )  
{  
    //set de instrucțiuni  
}
```

Această instrucțiune atribuie variabilei i o valoare inițială, iar apoi execută setul de instrucțiuni de atâtea ori cât timp atribuirea este făcută pentru fiecare ciclu de instrucțiuni executat, iar expresia are valoarea true.

Instrucțiunea WHILE

În JavaScript instrucțiunea WHILE are sintaxa:

```
while ({condiție})  
{  
    //set de instrucțiuni  
}
```

Prin instrucțiunea WHILE, setul de instrucțiuni este executat cât timp valoarea condiției este true.

3.4. Framework-ul Bootstrap

Bootstrap este un framework front-end, dezvoltat inițial de Twitter și ajuns acum la versiunea 3, permite realizarea de site-uri web responsive, care se adaptează la orice soluție de dispozitiv: desktop, tablete și telefoane mobile.

Este în momentul de față cel mai utilizat framework pentru dezvoltarea interfețelor web devenind foarte rapid standardul în crearea template-urilor pentru principalele sisteme CMS (Content Management System) cum sunt WordPress și Joomla.

Framework-urile de tip Bootstrap se folosesc de către acei web-designeri și dezvoltatori care cunosc bine măcar limbajul HTML și CSS și au nevoie de o bază solidă pentru începerea construirii proiectului.

Putem spune deci, că Bootstrap este un instrument utilizat pentru a gestiona cât mai bine faza inițială a unui proiect deoarece putem conta pe o serie de componente care pot fi reutilizate și personalizate oferindu-ne o bază solidă de pornire a proiectelor noastre.

Această funcție este sugerată și de numele framework-ului, termenul „bootstrap” însumând procesele necesare pentru pornirea computerelor la fel cum framework-ul Bootstrap ne pune la dispoziție instrumentele pentru pornirea proiectului nostru web.

În plus, mai oferă încă o facilitare, care nu este deloc ușor de trecut cu vederea: plaja de compatibilitate cu marile web browsere este foarte bună.

La baza dezvoltării continue a framework-ului Twitter Bootstrap este pre-procesorul LESS, ales datorită vitezei de compilare a codului (de 6 ori mai rapid ca SASS), și a eleganței și utilizării JavaScript-ului. Ca rezultat, oferă unui dezvoltator posibilitatea ajustării design-ului prin definirea sau setarea unui set de variabile sau parametrii și recompilarea surselor LESS rezultând un nou set de fișiere de stil (CSS).

LESS, nu este singurul pre-procesor de CSS disponibil, existând în momentul de față mai multe pre-procesoare printre care s-au remarcat printre dezvoltatorii de front-end: LESS, SASS (Syntactically Awesome Stylesheets) și SCSS. Folosirea acestora aduce o valoare adăugată oricărui framework de front-end. Însă modul lor de funcționare este asemănător, fiecare

încercând să ofere dezvoltatorului ușurință în a scrie un cod de calitate, și de a dezvolta un produs finit.

Aceste pre-procesoare au apărut ca o necesitate în urma limitărilor pe care le implică folosirea stilurilor CSS (lipsa variabilelor, și a facilității de re folosire a stilurilor în mai multe selecții CSS).

Deși la prima vedere, folosirea acestor pre-procesoare ar putea fi considerată ca o muncă adițională în realizarea unei aplicații sau pagini web, una dintre avantajele majore este câștigarea de timp în dezvoltare, alături de un cod mai curat (DRY = Don't Repeat Yourself), un fișier de CSS mai curat și ușor de întreținut.

Font-uri pentru reprezentarea icoanelor

O altă tendință este folosirea unui font în reprezentarea icoanelor pentru anumite acțiuni și reprezentarea de informații auxiliare. Unul dintre acestea: FontAwesome, este chiar definit ca fiind un font iconic dezvoltat pentru a fi folosit împreună cu framework-ul Twitter Bootstrap. Dar nu este singurul, el fiind complementat de câteva font-uri din aceeași categorie.

Un exemplu este Socialico, care oferă un set de caractere destinat reprezentării iconițelor celor mai populare rețele de socializare.



Figura 1 - Socialico

E o metodă nouă de prezentare a informațiilor de tip icoane, care vine și cu multe avantaje precum: scalabilitatea reprezentării fără pierderea calității (font-ul fiind reprezentat vectorial), o amprentă mai mică (dimensiune mai mică a fișierelor), mai puțină muncă de Photoshop și bineînțeles de creare sau modificare stiluri CSS.

4. Tehnologii server-side

4.1. Java Servlets

Java Servlets reprezintă tehnologia Java folosită pentru crearea aplicațiilor web server-side. Java Servlets este parte din Java Enterprise Edition (Java EE).

Un servlet este o clasă instanțiată în mod dinamic, pentru a răspunde cererilor primite de server de la un client. Acesta rulează în totalitate în Java Virtual Machine (JVM).

Față de alte tehnologii server-side, Java Servlets are anumite avantaje, cum ar fi:

- Eficiența, prin faptul că servleturile sunt instanțiate numai o dată, atunci când sunt încărcate prima dată de serverul web acest lucru fiind mai eficient decât instanțierea de fiecare dată atunci când serverul primește o cerere (Goodwill, 2001);
- Persistența, dată de faptul că servleturile pot să își păstreze starea între cereri. Odată încărcate în memorie, acestea așteaptă cereri.

Arhitectura Java Servlet de bază este dată de interfața `javax.servlet.Servlet` definită în pachetul `javax.servlet`. Printre metodele definite de această interfață, cele mai importante sunt: `init()`, `service()` și `destroy()`.

Metoda `init()` este apelată o singură dată pentru fiecare servlet, de către server, imediat după ce servletul a fost inițializat. Această metodă crează și inițializează resursele pe care le va folosi atunci când va răspunde cererilor.

Metoda `service()` poate fi apelată numai după apelarea metodei `init()`. Aceasta este apelată de fiecare dată când clientul trimite o cerere.

Metoda `destroy()` este apelată pentru fiecare resursă creată cu ajutorul metodei `init()` înainte de distrugerea unei instanțe. Această metodă poate fi folosită la închiderea conexiunilor la baza de date, dar și pentru salvarea anumitor informații

Preluarea datelor trimise de client

În cazul servleturilor metodele folosite pentru preluarea datelor trimise de client ca parametri sunt aceleași în cazul în care acestea au fost trimise ca cereri POST sau GET.

Cele trei metode folosite pentru preluarea parametrilor sunt:

```
public Enumeration ServletRequest.getParameterNames();  
public String ServletRequest.getParameter(String name);  
public String[] ServletRequest.getParameterValues(String name);
```

Metoda `getParameterNames()` returnează numele parametrilor cererii în forma unei enumerări de șiruri de caractere.

Metoda `getParameter(String name)` returnează valoarea parametrului cererii specificat prin numele său, dat ca parametru în metodă. Metoda returnează null dacă parametrul specificat nu există.

Metoda `getParameterValues(String name)` returnează valorile parametrului cererii specificat prin numele său, dat ca parametru în metodă, sub forma unui șir de strings.

Întreținerea sesiunii

Java Servlets conține metode proprii cu ajutorul cărora se pot întreține sesiunile. Aceste metode sunt oferite prin intermediul obiectului `HttpSession`. Aceste metode sunt:

```
public void setAttribute(String name, Object value)
public Object getAttribute(String name)
public String[] getAttributeNames()
public void removeAttribute(String name)
```

Cu ajutorul metodei `setAttribute(String name, Object value)` se poate stoca pe sesiune un obiect format dintr-un nume și o valoare.

Metoda `getAttribute(String name)` returnează obiectul din sesiune al cărui nume corespunde parametrului dat funcției.

Metoda `String[] getAttributeNames()` returnează un șir format din numele tuturor obiectelor salvate pe sesiune.

Metoda `removeAttribute(String name)` șterge obiectul din sesiune al cărui nume corespunde parametrului dat funcției.

4.2. Spring MVC

Spring MVC (model-view-controller) este un framework open source creat peste arhitectura Java Servlets.

Framework-ul Spring MVC este bazat pe cereri pe care le întreține cu ajutorul servletului `DispatcherServlet`. Acest servlet acționează ca un Controller de legătură care, după ce analizează cererea retrimite, la rândul său, cererile primite altor clase Controller (Johnson). Aceste clase decid apoi modul în care să răspundă cererii. Acest lucru este ilustrat în figura 2.

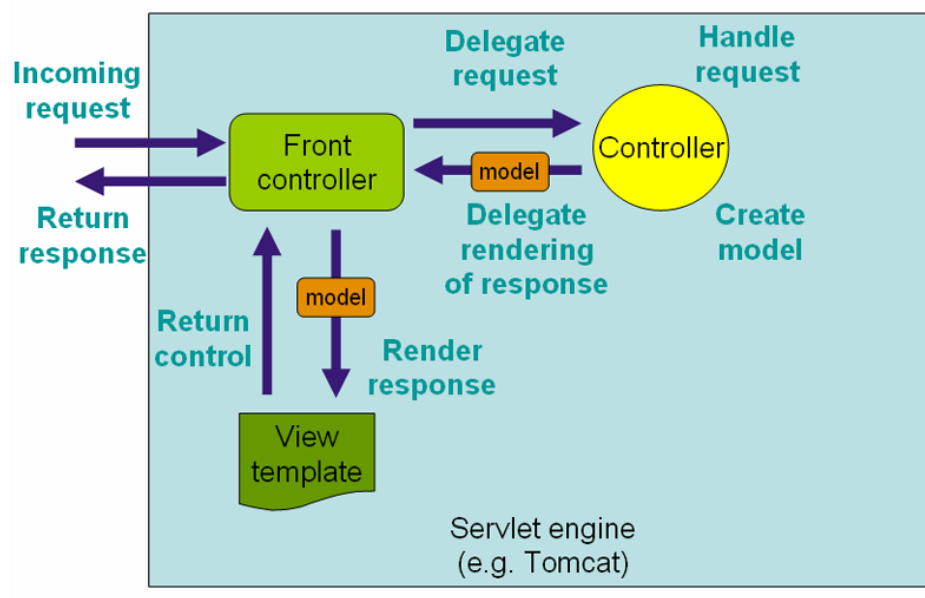


Figura 2 - MVC

Față de alte tehnologii server-side, Spring MVC oferă o separare mai clară între modelele, view-urile și controlerele aplicației (Deinum, 2012).

Inversion of Control

Una din principalele caracteristici ale framework-ului Spring îl reprezintă principiul Inversion of Control (IoC), numit și Dependency Injection (DI).

4.3. JavaServlet Pages

Un fișier JavaServlet Pages (JSP) reprezintă o clasă servlet. Acesta este format din cod Java încorporat într-un fișier de tip HTML. Atunci când se face o cerere pentru prima dată asupra unui fișier JSP, acesta este translatat într-un servlet care este apoi compilat și executat. Dacă asupra acestui fișier se mai fac alte cereri, serverul execută servletul corespunzător acestui fișier.

De obicei, fișierele JSP sunt folosite într-o aplicație web la crearea unor pagini HTML dinamice. Astfel, pagina dinamică corespunzătoare unui fișier JSP este reprezentată de codul HTML din acest fișier, căreia, prin aplicarea de cod Java, îi sunt generate în mod dinamic valori, afișate apoi de codul HTML (Murach, 2008).

De obicei, într-o aplicație model-view-controller (MVC) fișierele JSP sunt folosite la partea de „view”. Acestea sunt invocate de către un servlet, sau de către o clasă controller în cazul folosirii framework-ului Spring MVC.

4.4 Arhitectura aplicației web

Arhitectura unei aplicații web este formată din trei niveluri:

- Nivelul de reguli de control
- Nivelul de prezentare
- Nivelul de persistență

Nivelul de reguli de control este reprezentat de servleturi, sau de clase controller, care controlează fluxul de cereri asupra aplicației. Aceste servleturi sau clase pot la rândul lor apela clase care reprezintă nivelul de acces la baza de date din aplicație, cu ajutorul cărora pot manipula această bază de date. Mai mult, ele sunt responsabile și de invocarea fișierelor ce reprezintă nivelul de prezentare al aplicației.

Nivelul de prezentare este reprezentat de fișiere JSP și HTML. Acestea sunt invocate de către o anumită clasă controller atunci când aceasta primește cereri de la un client pentru unul din aceste fișiere.

Nivelul de persistență este reprezentat de clase folosite la prelucrarea unor sisteme de fișiere, cum ar fi fișiere XML, sau binare, sau baze de date folosite la salvarea de informații necesare pentru aplicație.

5. Protocolul WebSocket

Până la apariția protocolului de comunicare WebSocket, crearea de aplicații web care necesitau comunicări bidirecționale între client și server presupuneau utilizarea în exces a XMLHttpRequest. Acest lucru face comunicările bidirecționale mult prea complexe. O altă problemă a XMLHttpRequest este faptul că serverul este nevoit să folosească conexiuni TCP diferite pentru fiecare client, una pentru fiecare mesaj primit și alta pentru trimiterea de informații.

Spre deosebire de XMLHttpRequest, protocolul WebSoket oferă posibilitatea creării unui flux de comunicare bidirecțional în timp real dintre un browser și un server. Mai mult, protocolul WebSocket oferă o singură conexiune TCP pentru traficul în ambele direcții.

Un alt avantaj al protocolului WebSocket este faptul că, prin folosirea acestuia, comunicarea se face folosind un bandwidth mai mic, deoarece, spre deosebire de utilizarea XMLHttpRequest, nu sunt folosite headere la mesaje odată cu stabilirea conexiunii.

Pentru a putea deschide o conexiune WebSocket, clientul trebuie mai întâi să trimită serverului o cerere de schimbare a protocolului. O astfel de cerere arată astfel:

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket Connection:
Upgrade Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Origin: http://example.com Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
```

Dacă serverul acceptă un astfel de schimb protocol, atunci acesta returnează un răspuns care conține pe prima linie „101 Switching Protocols”. Un astfel de răspuns arată astfel:

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket Connection:
Upgrade Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: chat
```

Apoi, dacă cererea clientului a fost acceptată, serverul și clientul pot să înceapă să trimită mesaje în mod bidirecțional.

În realizarea aplicației practice am ales să folosesc protocolul WebSockets ca mod de comunicare dintre client și server datorită avantajelor pe care le oferă acest tip de comunicare și datorită simplității acesteia. Acest lucru l-am implementat client-side cu ajutorul framework-ului Spring WebSocket.

În plus, în aplicația practică am folosit STOMP (Stream-Oriented Messaging Protocol) ca protocol de mesagerie. Acest protocol definește un set de reguli semantice pentru mesaje, prin intermediul cărora clientul și serverul pot să comunice.

Protocolul STOMP este folosit împreună cu protocolul de transport WebSocket. Astfel, frame-urile trimise prin WebSocket folosesc sintaxa STOMP dată de comenzile:

- CONNECT
- SUBSCRIBE
- UNSUBSCRIBE
- SEND
- MESSAGE
- COMMIT
- ROLLBACK

Unul din marile dezavantaje al folosirii protocolului WebSocket este faptul că încă există un număr mare de browsere care nu suportă acest tip de protocol, iar unele rețele proxy nu permit folosirea acestuia. Acest lucru poate reprezenta o problemă deoarece, astfel, un număr mare de clienți nu ar avea posibilitatea de a folosi o aplicație care folosește acest protocol.

Pentru a întâmpina această problemă, în aplicația practică am folosit SockJS ca tehnologie de fallback. SockJS conține un set de protocoale de transport cu ajutorul cărora, în cazul în care un browser nu suportă protocolul WebSocket, acesta va găsi metode alternative de transport, păstrând, astfel, o comunicare activă între client și server (Wilton, 2010).

Diagrama ce reprezintă comunicarea WebSocket



Figura 3 – comunicarea WebSocket

În afară de setul de protocoale de transport, SockJS mai conține și o librărie JavaScript folosită de browser pentru a activa opțiunea de fallback, dar și librării server-side pentru mai multe framework-uri cum ar fi Spring Framework.

Modul de lucru al tehnologiei SockJS

Primul pas al acestei tehnologii este făcut de librăria client-side a acesteia. Aceasta face mai întâi o cerere de informații către server. După ce primește răspuns, trebuie să decidă ce fel de protocol de transport să folosească. Dacă este posibil, atunci acest protocol va fi WebSocket. Dacă nu, atunci SockJS încearcă să folosească pe rând mai multe protocoale de fallback, cum ar fi:

- Xhr-streaming
- Xdr-streaming
- Xhr-pooling
- Xdr-pooling
- Jsonp-pooling

6. Aplicația practică

Site-ul este alcătuit din mai multe pagini, fiecare reprezentând o secțiune a aplicației. Toate paginile conțin un meniu responsive, amplasat deasupra conținutului propriu-zis. Acest meniu permite navigarea rapidă a paginilor principale ale site-ului.

Prima pagină, intitulată „Home”, conține în momentul de față doar un buton care te redirecționează către cea mai interactivă pagină din aplicație.

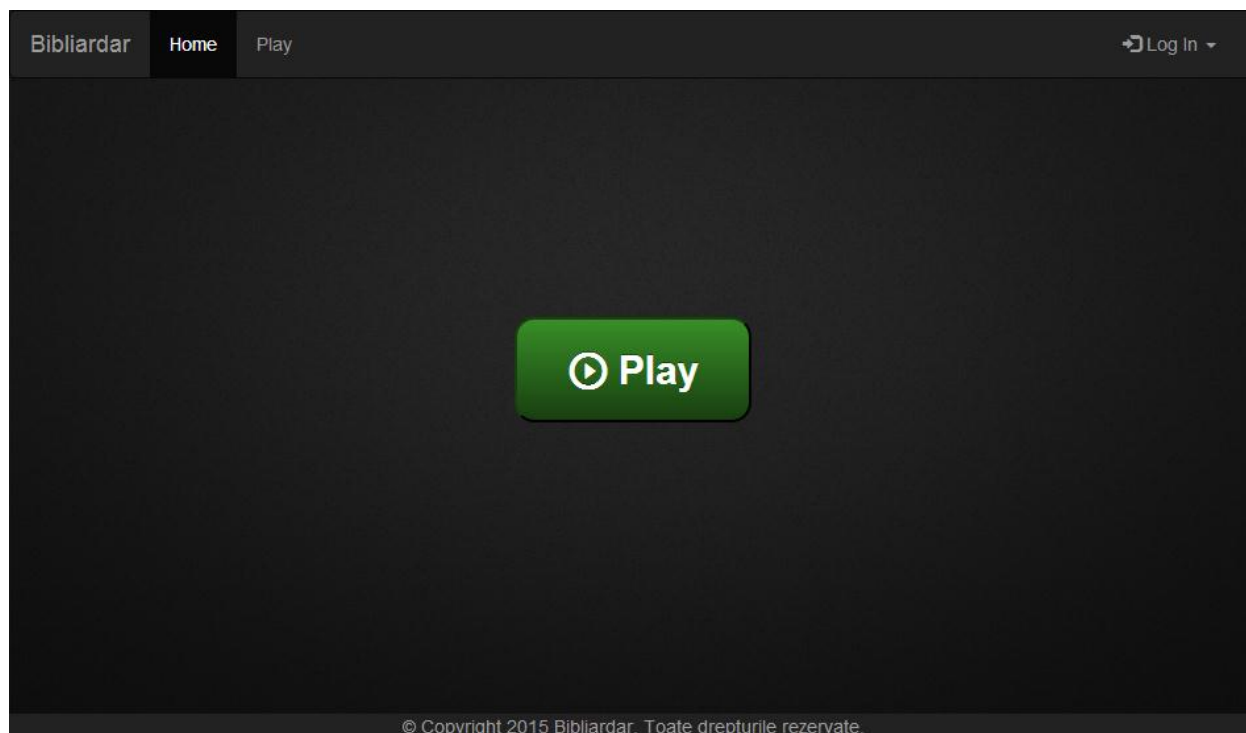


Figura 4 - Pagina „Home” pentru desktop

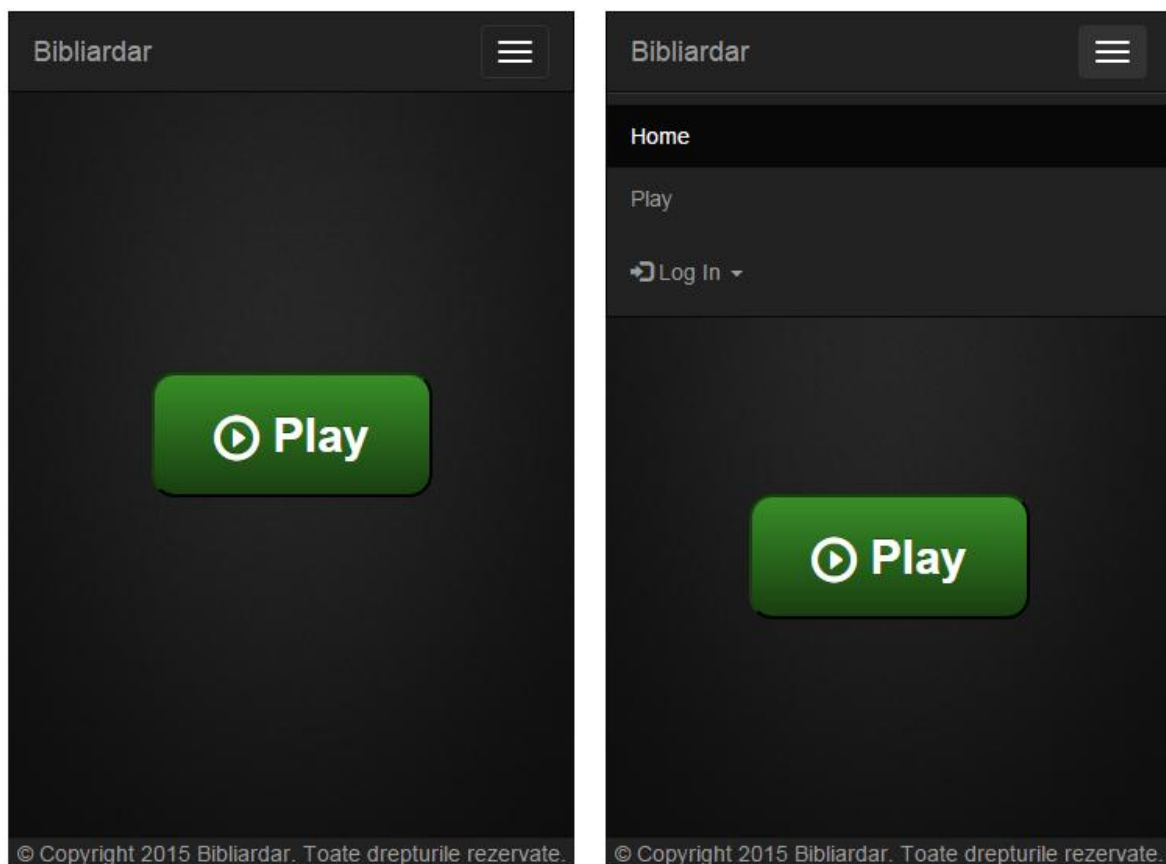


Figura 5 - Pagina „Home” pentru dispozitive mobile

Pagina care poate fi accesată prin apăsarea celui de-al doilea buton din meniu sau din conținutul primei pagini este „Play”, aceasta conține o listă cu toți utilizatorii care-s activi pe pagină, un chat care permite comunicarea între ei și un buton care ne introduce într-un joc de pregătire cu întrebări din Biblie.

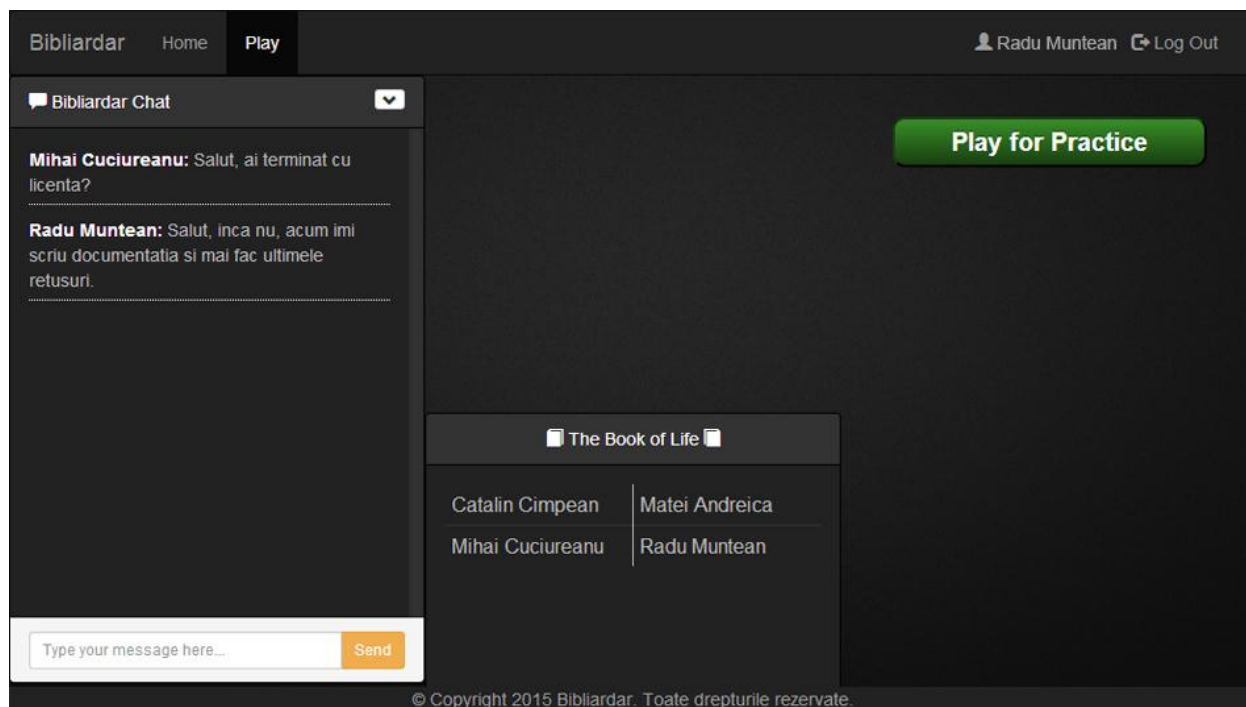


Figura 6 - Pagina „Play” pentru desktop

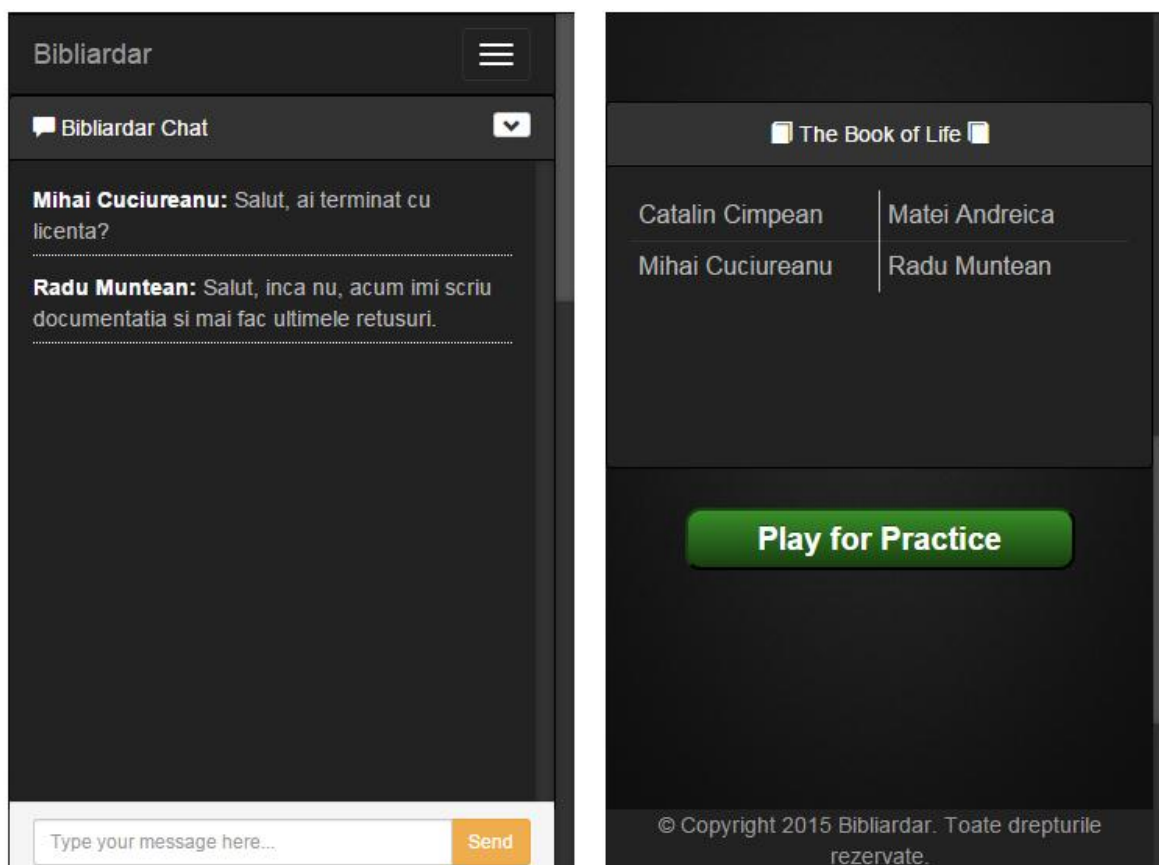


Figura 7 - Pagina „Play” pentru dispozitive mobile

Însa pentru a ajunge la pagina „Play” trebuie ca utilizatorul să fie logat, altfel nu va putea accesa pagina. Dacă nu este logat și totuși dorește să acceseze pagina, acesta va fi direcționat automat către pagina de logare.

Logarea se poate face atât din meniul aplicației cât și din pagina destinată acesteia. Acest lucru poate fi observat în figura 8 pentru vizualizarea în modul desktop, și respectiv din figura 9 pentru vizualizarea din perspectiva dispozitivelor mobile.

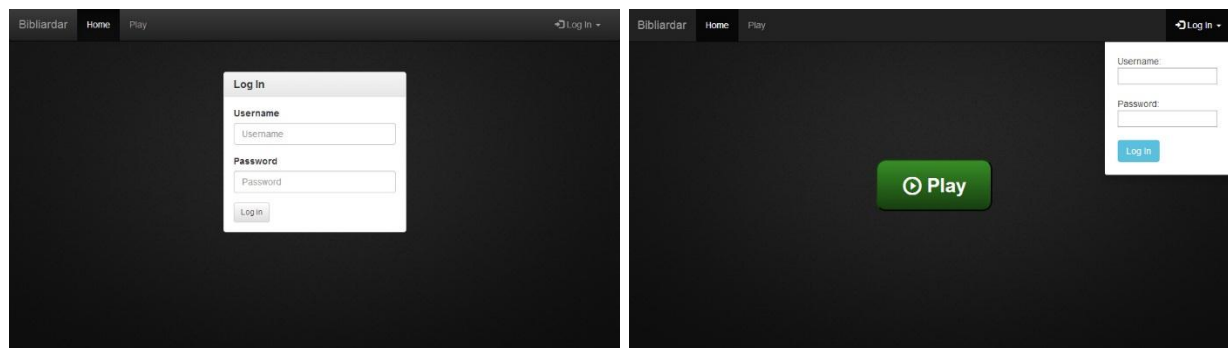


Figura 8 – Logarea atât din meniu cât și din pagina „Login” - pentru desktop

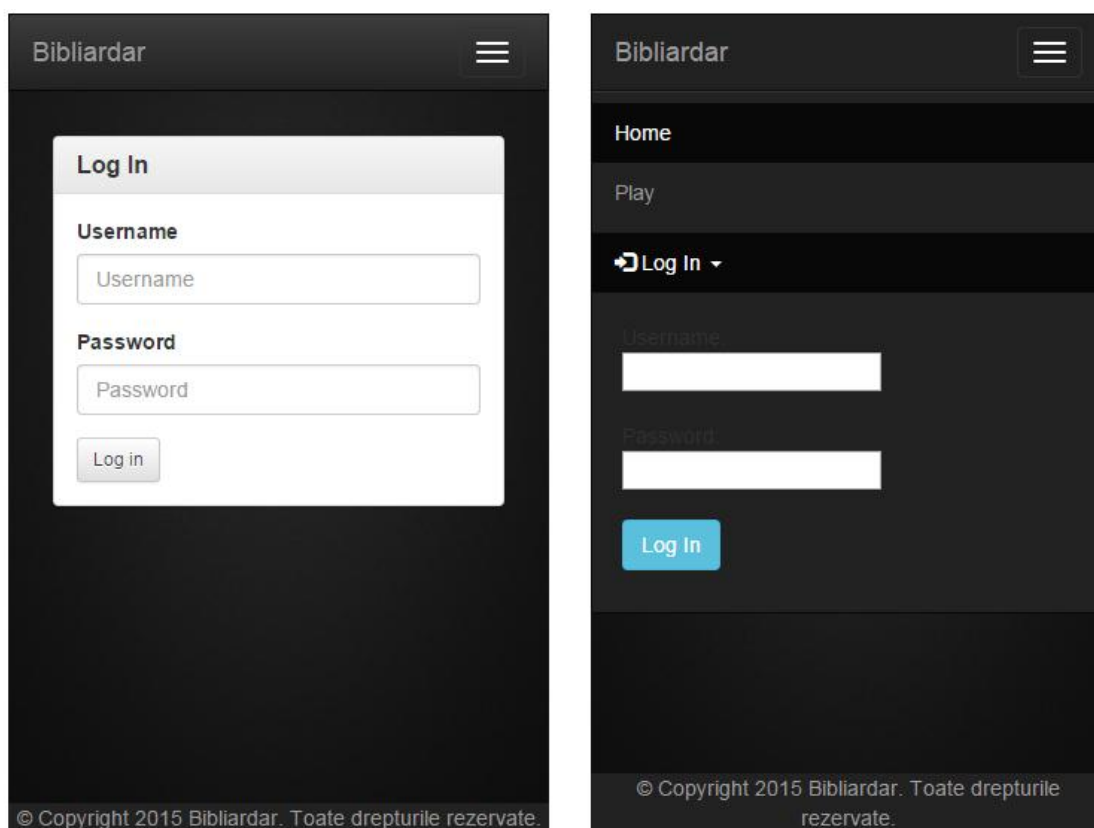


Figura 9 – Logarea atât din meniu cât și din pagina „Login” - pentru dispozitive mobile

Dacă logarea a avut loc cu succes, vei fi redirecționat către pagina „Home”, altfel vei fi informat cu un mesaj sugestiv despre eroarea ce a avut loc.

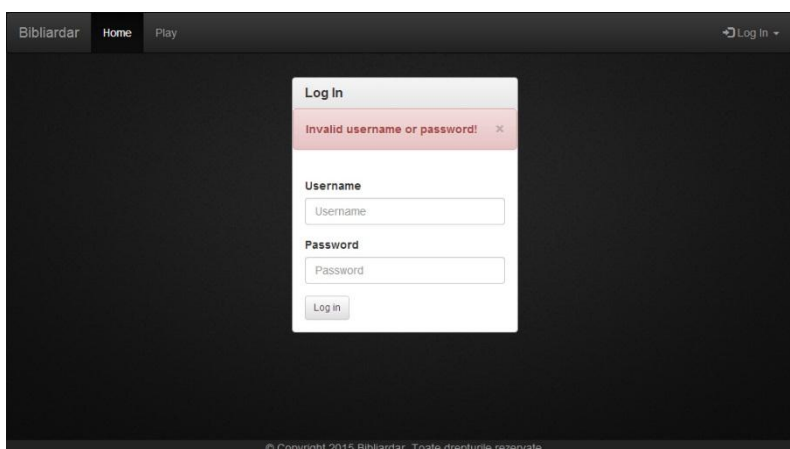


Figura 10 – Logarea a eșuat

Pagina care poate fi accesată prin butonul „Play for Practice”, ne redirecționează către o pagină destinată jocului de îmbogățire a cunoștințelor din domeniul Sfintelor Scripturi.

Această pagină ne întâmpină cu un buton central, care așteaptă să fie accesat pentru a începe jocul și urmărind prin aceasta că utilizatorul este pregătit. După accesarea butonului vom întâmpina o primă întrebare și patru variante de răspuns. În plus pentru ați putea evalua cunoștințele, în partea din dreapta sus vei fi informat pe tot parcursul jocului cu numărul de răspunsuri corecte din numărul de întrebări. După fiecare alegere a variantei, vei fi informat dacă răspunsul tău e corect sau greșit, iar dacă ai răspuns greșit, îți va fi afișat și răspunsul corect. Jocul continuă atât timp cât dorești să-l joci, după fiecare alegere, un buton va apărea care va permite continuarea jocului și va trece la afișarea următoarei variante. În alegerea variantei următoare s-a implementat un algoritm ce te va scăpa de neplăcerea de a întâmpina de mai multe ori aceeași variantă. Duplicatele vor apărea doar atunci când s-au epuizat toate variantele actuale. Jocul este de antrenament, deci se va termina în momentul în care vei apăsa butonul „x” (de ieșire).

În următoarele figuri vom observa cum decurge jocul atât din modul desktop cât și din cel al dispozitivelor mobile.



Figura 11 – Joc de pregătire

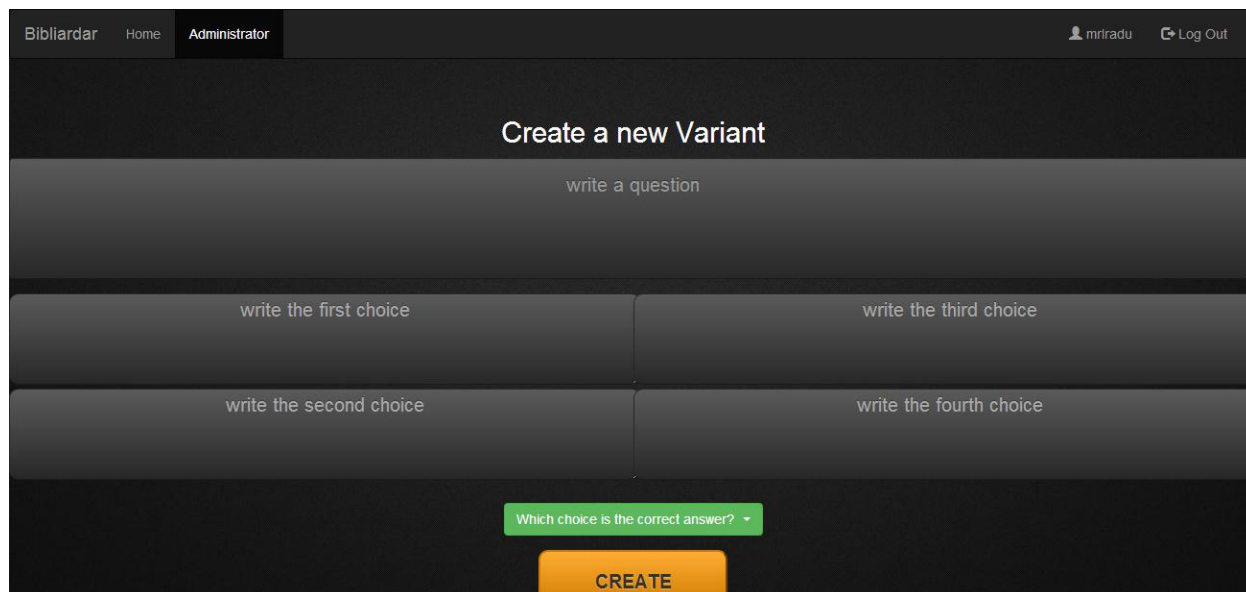


Figura 12 – Joc de pregătire, răspuns corect



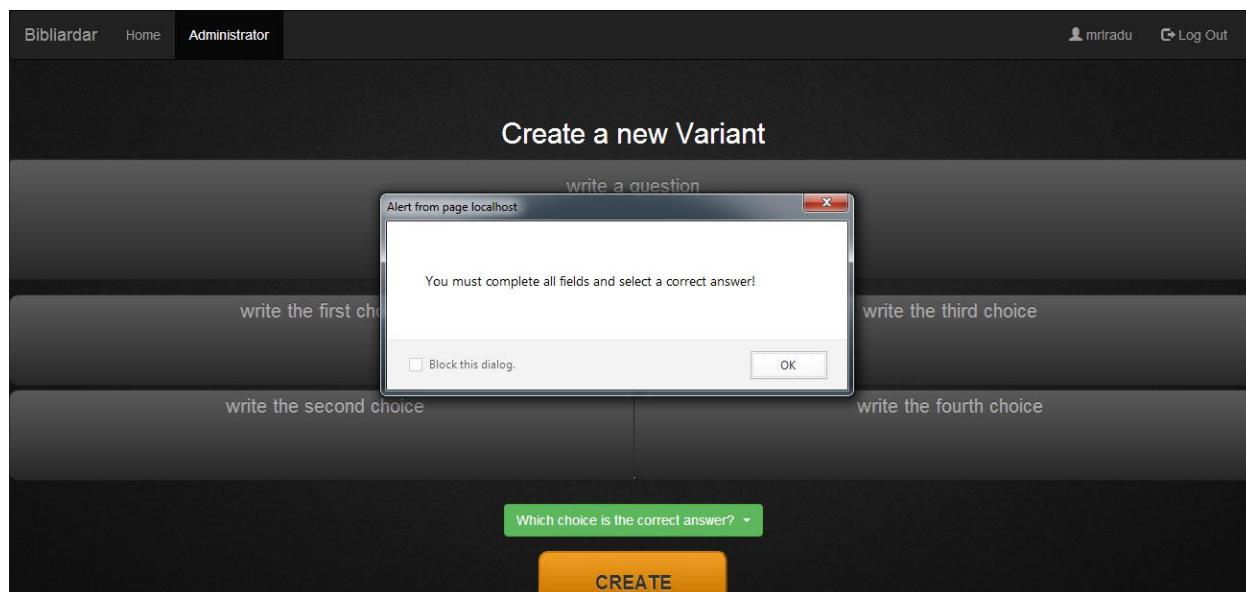
Figura 13 – Joc de pregătire, răspuns greșit

În cazul în care utilizatorul special „admin” este autentificat, aplicația îi va permite să adauge variante noi în baza de date și de fiecare dată va fi anunțat dacă operația a fost executată cu succes sau nu.



The screenshot shows the 'Create a new Variant' form in the Administrator section. The form has a dark background with light gray text. It includes a header bar with 'Bibliardar', 'Home', and 'Administrator' tabs, and a user profile 'mriradu' with a 'Log Out' button. The main form area has a title 'Create a new Variant' and a text input field labeled 'write a question'. Below this are four text input fields labeled 'write the first choice', 'write the second choice', 'write the third choice', and 'write the fourth choice'. At the bottom, there is a green dropdown menu labeled 'Which choice is the correct answer?' and an orange 'CREATE' button.

Figura 14 – adăugare variantă în baza de date



The screenshot shows the same 'Create a new Variant' form as in Figure 14, but with an alert dialog box displayed in the center. The dialog box has a title bar 'Alert from page localhost' and a message 'You must complete all fields and select a correct answer!'. It includes a checkbox labeled 'Block this dialog.' and an 'OK' button. The background form is dimmed.

Figura 15 – adăugarea a eșuat

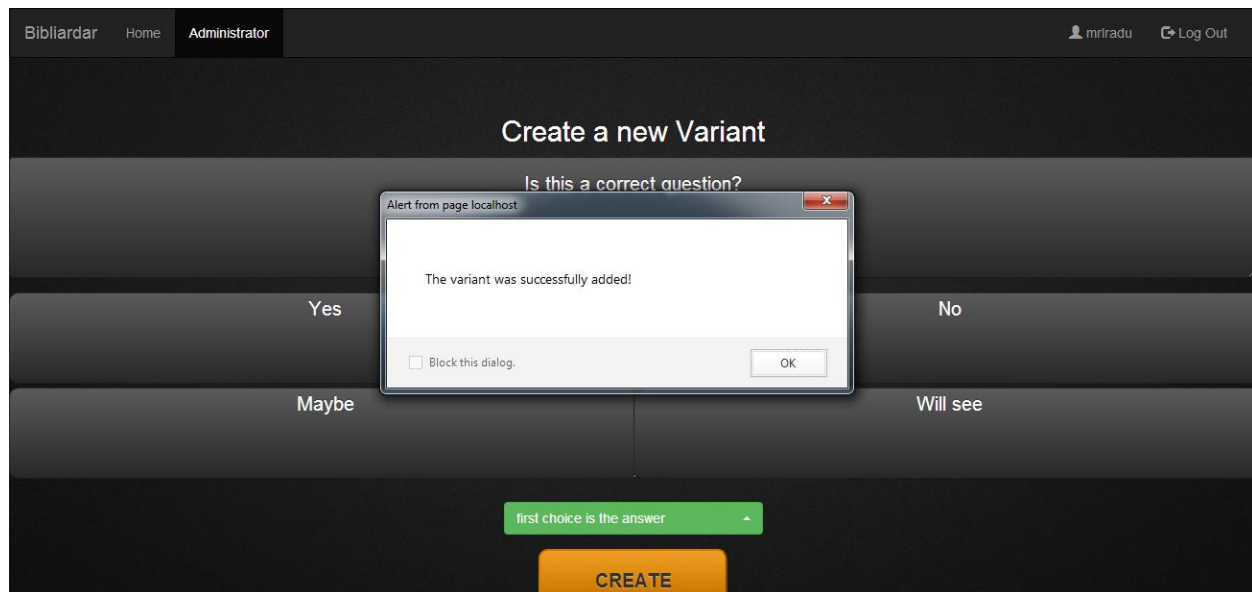


Figura 16 – adăugarea a fost realizată cu success.

7. Concluzii

În concluzie, putem observa cum această modalitate de îmbogățire a cunoștințelor prin intermediul unui joc poate fi implementată cu succes, și care atrage o gamă mai mare de utilizatori înspre a se documenta, fiind o cale mai accesibilă și interactivă.

Caracteristicile originale ale acestei aplicații îi conferă un avantaj în fața altor jocuri similare. Acestea constau în câteva aspecte ce îmbunătățesc funcționarea. Dintre ele amintim: posibilitatea ca interfața să se actualizeze în timp real în funcție de comenzile executate de utilizatori, făcând astfel procesul de documentare mai interactiv și mai facil. Acest lucru se realizează cu ajutorul protocolului de comunicare WebSocket care, spre deosebire de modul de comunicare obișnuit prin care un client trimite cereri serverului la care acesta răspunde, în acest caz serverul poate să comunice cu orice client fără ca în prealabil să fi primit o cerere din partea acelui client.

Cele cinci capitole care descriu conceptele de bază ale unei aplicații web în general, precum și tehnologii server-side și client-side necesare unei astfel de aplicații, oferă o perspectivă detaliată asupra provocărilor și dificultăților întâmpinate în crearea aplicației practice.

Pe viitor, o astfel de aplicație poate fi extinsă pentru a deservi și în alte domenii care pot beneficia de pe urma caracterului interactiv. O eventuală îmbunătățire ar putea consta în implementarea unui mod de joc mult mai competitiv.

Bibliografie

- [1] Brooks, David R.: *An Introduction to HTML and JavaScript for Scientists and Engineers*, Springer, London, 2007.
- [2] Meyer, Eric A.: *CSS: The Definitive Guide*, O'Reilly, Sebastopol, CA, 2007.
- [3] Flanagan, David: *JavaScript: The definitive guide*, O'Reilly, Sebastopol, CA, 2002.
- [4] Saternos, Casimir: *Client-server Web Apps with JavaScript and Java*, O'Reilly, Sebastopol, CA, 2014.
- [5] Goodwill, James: *Developing Java Servlets*, SAMS, Indianapolis, IN, 2001.
- [6] Deinum, Marten, Serneels, Koen: *Pro Spring MVC: With Web Flow*, Apress, New York, NY, 2012.
- [7] Murach, Joel, Steelman, Andrea: *Murach's Java Servlets and JSP: Training & Reference*, Mike Murach & Associates, Fresno, CA, 2008.
- [8] Wilton, Paul, McPeak, Jeremy: *Beginning JavaScript*, Wiley Publishing, Inc., Indianapolis, IN, 2010.
- [Chang] Chang, Patrick, Vittie, Lori Mac: *The Fundamentals of HTTP*, <http://www.f5.com/pdf/white-papers/http-fundamentals-wp.pdf>.
- [Johnson] Johnson, Rod, Hoeller, Juergen, et al: *Spring Framework Reference Documentation*, <http://docs.spring.io/spring/docs/4.0.0.RELEASE/spring-framework-reference/htmlsingle/>.
- [Robie] Robie, Jonathan: *What is the Document Object Model?*, <http://www.w3.org/TR/1998/WD-DOM-19980720/introduction.html>.