Facultatea de Electronică, Telecomunicații și Tehnologia Informației
Universitatea Politehnica București

Aplicații web realizate cu tehnologia PHP și baze de date MongoDB

Lăcătușu Raluca – Cristina
Master IISC
An II

București 2015

Cuprins

Introducere

În ultimii azi, aplicațiile web au avut o creștere exponențială, datorându-se mai ales din cauza faptului ca Internetul a devenit un element important în viața majorității oamenilor.

Pentru a beneficia de aplicațiile web avem nevoie de un browser de Internet și de o conexiune la internet. O mare calitate a aplicațiilor web o reprezintă faptul ca actualizarea și menținerea lor nu depinde de utilizator. O altă calitate o reprezintă faptul că nu depind de sistemul de operare. [1]

În trecut, o aplicație web avea nevoie de instalarea parții de client local. Cu alte cuvinte, o aplicație avea propriul ei client care servea ca interfață de utilizator și trebuia să fie instalat separat pe calculatorul personal al fiecărui utilizator.

Acum, aplicațiile web folosesc documente web scrise într-un format standard și sunt susținute de o varietate de browsere web. Aplicațiile web poate fi considerate ca o variantă specifică a software client-server în care software-ul client este descărcat în mașina client atunci când vizitează pagina web relevantă, folosind proceduri standard, cum ar fi HTTP.

Actualizări de software web de Client se pot întâmpla de fiecare dată când pagina web este vizitata. În timpul sesiunii, browser-ul web interpretează și afișează paginile, și acționează ca si client universal pentru orice aplicatie web. [1]

În 1995 Netscape a introdus un limbaj client-side scripting numit JavaScript care permite programatorilor adăugarea unor elemente dinamice în interfața utilizatorului. Deci, în loc de a trimite date la server pentru a genera o întrega pagină web, putem avea un script care poate efectua diferite sarcini, cum ar fi de validarea formularelor, afisarea sau asunderea unor părți ale paginii.[1]

În această lucrare voi incerca să prezint câteva concepte noțiuni teoretice despre PHP, MongoDB, HTML, CSS și JavaScript, dar și modul în care se poate crea o aplicație web folosind aceste tehnologii.

1. PHP

1.1 Generalități

„PHP este un limbaj de programare. Acest nume provine din limba engleză și este un acronim recursiv: **P**hp: **H**ypertext **P**reprocessor. În principiu, acest limbaj de programare este folosit pentru crearea paginilor/aplicațiilor web dinamice, dar, începând cu versiunea 4.3.0, poate fi folosit și în modul „linie de comandă”. Este unul din cele mai importante limbaje de programare web open- source și server-side. PHP însemna inițial *Personal Home Page*.”

PHP rulează pe toate sistemele de operare importante, de la variante UNIX, incluzând Linux, FreeBSD, Ubuntu, Debian și Solaris, până la Windows și MacOSX. Poate fi folosit pe servere Apache, Microsoft IIS și Netscape/iPlanet.

Limbajul este unul flexibil. Poate genera o mulțime de documente: PDF, GIF, JPEG și PNG, dar și filme Flash. Una din caracteristicile importante ale PHP-ului este posibilitatea de a comunica cu bazele de date. PHP poate fi integrat cu foarte multe tipuri de baze de date: MySQL, PostgreSQL, Oracle, Sybase, MS-SQL, DB2. Chiar si sistemele de baze de date mai recente sunt suportate de PHP, cum ar fi MongoDB.

PHP deține o librărie cu cod PHP pentru a ajuta programatorul să efectueze sarcini obișnuite, cum ar fi conectarea la o bază de date, corectarea erorilor, etc.

„PHP este un limbaj server-side. Un limbaj server-side este similat cu JavaScript și permite integrarea unor mici programe (script-uri) în codul HTML al unei pagini/aplicații web. Cînd se execută, aceste programe oferă un control mai mare față de HTML, asupra a ceea ce va apărea în fereastra browser-ului. Diferența esențială dintre JavaScript și PHP este stadiul de încărcare a paginii web în care aceste programe sunt executate.[2]

Codul scris în limbaje client-side, cum ar fi JavaScript, este citit și executat de către browser după ce pagina web este descărcată de pe server. În schimb, codul scris în limbajele server-side (PHP), este executat de către serverul web, înainte de a trimite pagina web browser-ului. În timp ce limbajele client-side oferă control asupra modului în care o pagină web se comportă atunci când este afișată în browser, limbajele server-side permit generarea paginilor personalizate înainte ca acestea să fie trimise către browser. După ce serverul web a executat codul PHP incorporat în pagina HTM, rezultatul ia locul codului PHP în pagină.” [3]

Rasmus Lerdorf a creat PHP în 1994, dar varianta actuală este foarte diferită de cea inițială. Lerdorf a scris o serie de scripturi Perl Common Gateway Interface (CGI), pe care le utiliza la administrarea propriului web-site. Aceste scripturi erau utilizate pentru a afișa diferite informații, dar și pentru a înregistra traficul pe pagina sa personală. El a rescris aceste script-uri în C pe motice de performanță, modificându-le astfel încât să poată lucra cu formulare web, dar și pentru a se putea conecta la baze de date. A redenumit acest proiect „Personal Home Page/Forms Interpreter” sau PHP/FI. PHP/FI putea fi utilizat pentru a construi aplicații web simple, dinamice. Ulterior, când a lansat prima versiune, a adăugat și posibilitatea de a integra sintaxa HTML. Sintaxa PHP era similară cu sintaxa Perl.

Zeev Suraski și Gutmans Andi au rescris limbajul în 1997 și astfel, au lansat PHP 3, schmbând numele limbajului în acronimul recursiv PHP: Hypertext Preprocessor. După o perioadă de testare, au lansat versiunea oficială în Iunie 1998. Suraski și Gutmans au început să modifice nucleul PHP, producând motorul Zend în 1999. Au fondat Zend Technologies, avâand sediul ăn Ramat Gen, în Israel.

În august 2008, PHP 4 a fost lansat de către Zend. În iulie 2004, PHP 5 a fost lansat. Această versiune a adus noi îmbunătățiri, suportând programarea obiect orientată.

Sintaxa PHP este foarte asemănătoare cu sintaxa JavaScript, C, C++, C#, Objective-C, Java, Perl. Numele claselor și funcțiilor definite de utilizator, precum și unele cuvinte cheie (echo, while, class) sunt nu țin cont de majusculă (case-insensitive). Numele variabilelor sunt sensibile la majusculă (case-sensitive).

Un script PHP constă dintr-o serie de comenzi, instrucțiuni sau declarații. Fiecare linie de comandă este o instrucțiune adresată server-ului web, pe care acesta trebuie să o execute pentru a putea trece la următoarea instrucțiune. Fiecare comandă, instrucțiune sau declarație este terminată cu simbolul „;”. O instrucțiune complexă folosește acoladele pentru a marca un bloc de cod.

Simbolul „$” este folosit în multe limbaje de programare, sub diverse forme. În PHP, acest simbol trebuie scris înaintea tuturor variabilelor. Acest lucru este necesar pentru a face mai rapida parsarea codului. [3]

Spre deosebire de alte limbaje de programare, cum ar fi Python, care sunt stricte în privința identării, PHP nu ține cont de acest lucru. Totuși, identarea codului este recomandată pentru a-l înțelege mai bine și pentru a găsi mai ușor eventualele greșeli de sintaxă.

Funcțiile sunt utilizate pentru a separa secțiuni de cod care realizează o anumită prelucrare a codului. Funcțiile sunt de obicei realizate în cazul în care aceeași operație se va executa de mai multe ori în același program. Prin folosirea funcțiilor vom economisi timp și linii de cod. Dacă nu vom transforma un anumit bloc de cod într-o funcție, iar acel bloc de cod este folosit de mai multe ori în program, și vom decide să modificăm o instrucțiune, va fi destul de greu pentru programator să modifice blocul de cod oriunde apare. Transformarea blocurilor de cod identice în funcții scurtează codul sursă, îl face mai ușor de citit, dar și mai rapid. [2]

Dacă avem un program cu foarte multe linii de cod, este posibil să atribuim acelaşi nume de variabilă de mai multe ori. Folosind PHP, avem posibilitatea de a decide în ce scop putem declara și accesa o variabilă. De exemplu, putem spune ca variabila $timp poate fi utilizată doar în interiorul unei funcții. În PHP putem declara și variabile locale.

1.2 OOP

Programarea orientată pe obiecte facilitează design-ul programelor. „POO este unul dintre cei mai importanți pași făcuți în evoluția limbajelor de programare spre o abstractizare în implementarea programelor. A apărut din necesitatea exprimării problemei într-un mod natural ființei umane.” [4]

POO înțelege legătura fundamentală între date și codul care funcționează cu aceste date și permite proiectarea și implementarea programelor în jurul acestei legături. „De exemplu, un sistem pentru evidența unor utilizatori ai unei platforme web păstrează date despre foarte multe persoane. Într-un limbaj de programare procedurală, fiecare persoană reprezintă o structură de date și are atașată o listă de funcții care interacționează cu structura respectivă de date (crearea de noi persoane etc.). Într-un limbaj de programare orientat pe obiecte, fiecare utilizator reprezintă un obiect - o structură de date care conține și cod. [4]

Un obiect de tip „persoană” conține numele utilizatorului și parola aferentă, dar și un mecanism de recunoaștere a mesajelor postate de utilizator. Un obiect de tip mesaj știe de către ce thread/discuție aparține. Un obiect de tip discuție este o colecție de obiecte de tip mesaj.

Obiectul, ca uniune de cod și date, este unitatea modulară pentru dezvoltarea aplicațiilor li reutilizarea codului. Axându-ne pe exemplul de mai devreme, avem nevoie de aceleași informații pentru fiecare utilizator. Când proiectam aplicația, definim câmpurile și funcțiile pentru fiecare utilizator. În termeni de programare obiect-orientată, descriem clasa *utilizator*.

Un *obiect* este o instanță a unei clase. Obiectele și clasele seamănă cu valorile și tipurile de date. Există doar un tip de date integer, dar mai multe numere întregi posibile. În mod similar, programul definește o clasă *utilizator*, dar pot fi creați mai mulți utilizatori.

Datele asociate unui obiect se numesc *proprietăți.* Funcțiile asociate unui obiect se numesc *proprietăți.* Când se definește o clasă, sunt definite si proprietățile sale, dar și metodele sale.

Depanarea și întreținerea programului sunt mult mai ușor de realizat dacă se folosește incapsularea. O clasă are anumite metode care sunt folosite de obiecte, iar codul exterior acestei clase nu are acces direct la structura datelor clasei.

În programarea obiect orientată întâlnim noțiunea de *moștenire*. Acesta este un mod de a defini o nouă clasă, spunând că este o clasă existentă, dar cu anumite proprietăți și metode noi sau modificate. Vechea clasă se numește *superclasă*(sau părinte sau clasă de bază), iar noua clasă se numește *subclasă*(sau clasă derivată). Moștenirea este o formă de reutilizare a codului de bază. Astfel, codul de bază este reutilizat în loc de a fi recopiat în noua clasă. Orice modificări facute în clasa de bază sunt automat și în clasa derivată.

Pentru a crea un obiect al unei clase existente putem folosi:

$object = new Class;
Presupunând că s-a definit clasa *Persoana*, vom crea un obiect de tip *Persoana*: *$persoana = new Persoana;*

O dată creat obiectul, putem folosi operatorul „->” pentru a accesa metodele și proprietățile obiectului:

$object->proprietate $object->metoda([*arg*, ...])

Metodele se comportă ca și nițte funcții (specifice obiectului în cauză); ele pot primi argumente și pot returna valori. În definiția unei clase putem specifica ce metode și ce proprietăți sunt publice, dar și care dintre ele sunt accesibile doar în interiorul clasei, folosind modificatori de acces *public* și *privat.* Putem utiliza acești modificatori pentru incapsulare. [4]

Pentru a proiecta și dezvolta aplicații în PHP, într-o manieră orientată pe obiecte, vom folosi cuvântul cheie *class*. Definirea unei clase presupune definirea numelui, proprietățile și metodele acesteia. Numele clasei este case-insensitive și trebuie să respecte regulile PHP-ului pentru identificatori.

O metodă este o funcție definită în interiorul unei clase. Chiar dacă PHP nu impune nicio restricție asupra definirii funcțiilor, cele mai multe metode acționează numai asupra datelor din obiectul în care se află metoda. Numele metodelor care încep cu două simboluri *underscore* (__) sunt specifice PHP-lui li sunt folosite pentru serializarea obiectelor. [4]

În cadrul unei metode, variabila *$this* conține o referință la obiectul în care a fost definită metoda. Metodele utilizează această variabilă pentru a accesa proprietățile obiectului curent și pentru a apela alte metode pentru acel obiect. Declararea proprietăților sunt opționale.

Folosind modificatori de acces, putem schimba vizibilitatea proprietăților. Proprietățile care sunt accesibile din afara domeniului de aplicare a obiectului ar trebui să fie declarate publice. Proprietățile unei instanțe care pot fi accesate doar de metodele din aceeași clasă trebuie să fie declarate folosind modificatorul de acces *private*. În cele din urmă, proprietățile declarate ca și *protected* pot fi accesate doar de metodele clasei obiectului și de metodele claselor care moștenesc clasa respectivă."[5]

1.3 MVC

MVC este un design pattern folosit în domeniul software, în special în web-development. Acestă arhitectură permite izolarea părții logice de interfața proiectului. Modelul reprezintă informația de care are nevoie aplicația, view-ul reprezintă elementele de interfață, iar controller-ul reprezintă sistemul comunicativ care procesează datele informționale, facând legătura între model și view.

Modelul reprezinta partea de hard-programming, partea logica a aplicatiei. El are in responsabilitate actiunile si operatiile asupra datelor, autentificarea utilizatorilor, integrarea diverselor clase ce permit procesarea informatiilor din diverse baze de date.[6]

View-ul se ocupa de afisarea datelor, practic aceasta parte a programului va avea grija de cum vede end-userul informatia procesata de controller. O data ce functiile sunt executate de model, viewului ii sunt oferite rezultatele, iar acesta le va trimite catre browser. In general viewul este o mini-aplicatie ce ajuta la randarea unor informatii, avand la baza diverse template-uri.

Controller-ul reprezinta creierul aplicatiei. Aceasta face legatura intre model si view, intre actiunile userului si partea decizionala a aplicatiei. In functie de nevoile utilizatorului, controllerul apeleaza diverse functii definite special pentru sectiunea de site in care se afla userul. Functia se va folosi de model pentru a prelucra (extrage, actualiza) datele, dupa care informatiile noi vor fi trimise catre view, ce le va afisa apoi prin template-uri.[6]

1.4 Integrarea PHP-ului cu bazele de date

PHP suportă peste 20 de sisteme de baze de date. Bazele de date relaționale, cum ar fi MySQL, PosrgreSQL și Oracle sunt cele mai folosite pentru aplicațiile web dinamice. Există doua moduri de a accesa bazele de date cu ajutorul PHP-ului. Prima este utilizarea unei extensii ale bazei

de date, iar cealaltă este utilizarea unui driver numit PDO. PDO(PHP Data Objects) poate fi folosit ca un nivel de abstractizare pentru conexiunea dintre programele PHP și diverse baze de date. PDO definește o interfață simplă și consistentă între diverse baze de date.

Dacă utilizăm o extensie specifică unei baze de date, codul este strâns legat de baza de date utilizată. De exemplu, dacă utilizăm extensia MySQL-ului, numele funcțiilor, parametrii și modul de manipulare al erorilor sunt complet diferite de cele ale altor extensii de baze de date. Dacă dorim migrarea de la MySQL către PostgreSQL, atunci codul va trebui modificat aproape în totalitate. Pe de altă parte, PDO, nu accesează funcții specifice extensiilor sistemelor de baze de date, astfel că migrarea de la un sistem de baze de date la altul se face foarte ușor.

Portabilitatea abstractizării librăriei PDO are un dezavantaj: codul este puțin mai lent decât codul care utilizează extensiile specifice bazelor de date. [5]

PHP comunică cu bazele de date relaționale, cum ar fi MySQL și Oracle folosind Structured Query Language (SQL). Sintaxa SQL este împărțită în două părți. Prima, limbajul de manipulare a datelor sau DML(Data Manipulation Language) este utilizată pentru a prelua și modifica datele într-o bază de date existentă. LMD este compact și este format din doar 4 acțiuni: SELECT, INSERT, UPDATE și DELETE. Setul de comenzi SQL utilizat pentru a crea și modifica structurile bazei de date este cunoscut sub numele de Data Definition Language sau DDL.

Extensia PDO definește o interfață pentru accesarea datelor prin intermediul PHP-ului. Fiecare driver al unui sistem de baze de date care implementeaza interfața PDO poate impune reguli specifice bazelor de date specifice cum ar fi expresiile regulate.

PDO are următoarele caracteristici unice:

- este o extensie nativă C
- are avantajul de a folosi noile caracteristici specifice PHP 5
- folosește buffer-ul de citire a datelor din setul de rezultate
- oferă caracteristici comune bazelor de date
- este încă în măsură să acceseze funcții specifice bazelor de date • poate utiliza tehnici bazate pe tranzacții
- poate interacționa cu LOBS (Large Objects) în baza de date • poate implementa cursoare

„Bazele de date de tip NoSQL sunt în creștere de popularitate, deoarece lucrează altfel decât structura tipică de SQL. Bazele de date de tip NoSQL sunt de asemenea foarte populare pentru aplicațiile pentru terminalele mobile.

```
$mongo = new Mongo();
$db = $mongo->library;
$authors = $db->authors;
$author = array('authorid' => 1, 'name' => "J.R.R. Tolkien"); $authors->insert($author);

$author = array('authorid' => 2, 'name' => "Alex Haley"); $authors->insert($author);
$author = array('authorid' => 3, 'name' => "Tom Clancy"); $authors->save($author);
```

Prima linie de cod din exemplul de mai sus descrie crearea unei noi conexiuni la motorul bazei de date Mongo și a unei interfețe obiect. Următoarea linie arată conectarea la librăria colecției, și dacă nu există colecția, atunci Mongo o va crea. Vom crea o interfață obiect cu conexiunea la baza de date *$db* și vom crea un document în care vom stoca datele despre autor. În următorii pași putem vedea cum sunt adăugate datele în documentul *autori* în două moduri diferite.

Primele două instrucțiuni folosesc metoda *insert(),* iar a treia instrucțiune folosește metoda *save()*. Singura diferență între aceste două metode este că metoda *save()* va actualiza o valoare în cazul în care aceasta este deja în documentul respectiv și are o cheie existentă *_id*."[7]

2. Baze de date MongoDB

2.1 Introducere

MongoDB (numele provine de la „humongous" - enorm) este un sistem de baze de date open- source orientat pe documente, dezvoltat de compania 10gen. Acest sistem de baze de date face parte din familia NoSQL. Spre deosebire de sistemele de baze de date clasice relaționale care stochează informția în tabele, MongoDB stochează datele sub forma documentelor de tip JSON, având o schmemă dinamică. Mongo numește acest format de stocare BSON.

Acest sistem de stocare face ca înregistrarea datelor să fie mai ușoară, dar și mai rapidă. 10gen a început dezvoltarea acestui sistem de baze de date în octombrie 2007. MongoDB este utilizat de MTV Networks, Craiglist, Foursquare si UIDAI Aadhaar. MongoDB este cel mai popular sistem de baze de date din familia NoSQL. [8]

MongoDB este ușor de învățat. Există totuși similarități între conceptele bazelor de date MongoDB și conceptele bazelor de date de tip relațional. Dezvoltatorii care au utilizat RDBMS și migrează către MongoDB pot întâmpina diverse probleme de adaptabilitate.

Implementează ideea unui design flexibil al bazei de date. Nu trebuie definită structura bazei de date înainte de a stoca informațiile, ceea ce este folositor atunci când trebuie să stocăm date nestructurate. Acest sistem este foarte scalabil. Are foarte multe opțiuni pentru a menține performanțe optime, în timp ce dimensiunea și traficul datelor cresc, fără efectua schimbări în structura aplicației.

MongoDB este gratis și poate fi downloadat și utilizat fără a plăti. Are o documentație foarte bine pusă la punct, iar utilizatorii acestui sistem de baze de date contribuie la dezvoltarea lui în permanență.

MongoDB a fost creat de către fondatorii DoubleClick, proiect care a fost cumpărat de către Google pentru suma de 3.1 miliarde de dolari. După vânzarea acestui proiect, echipa a început să lucreze la numeroase proiecte, dar au întâmpinat probleme de scalabilitate. [8]

În 2007 au fondat compania 10gen si au început să lucreze la o platformă pentru cloud, asemănătoare cu Google App Engine. Platforma celor de la 10gen era dezvoltată server-side, folosind JavaScript. Aceasta platformă a fost numită inițial „ed", acest nume fiind compus din inițiala numelor celor 2 asociați - Elliot și Dwight, iar bza de date a fost numită „p". În vara anului 2008 au decis să schimbe numele acestor două proiecte, Sistemul de baze de date a fost redenumit „Mongo", iar platforma pentru cloud a primit numele „Babble".

După un an de zile au renunțat la platforma Babble și au transformat Mongo într-un proiect open- source. După acest pas, sistemul de baze de date a început să devină popular printre utilizatori. În ultimul an, MongoDB s-a dezvoltat foarte mult, deoarece numărul de utilizatori a crescut exponențial. Fiind un sistem de baze de date open-source, utilizatorii au customizat sistemul, creând proiecte proprii: Casbah, Morphia, MongoMapper, Mongoose, CandyGram, MongoKit, Mongoid, Ming, MongoEngine, Pymongo-Bongo, ActiveMongo, morph și MongoRecord. Utilizatorii integrează acest sistem de baze de date în numeroase proiecte deja existente: Drupal, Doctrine, Django, ActiveRecord, Lighttd sau NGINX.

Cine utilizează MongoDB?

- Craiglist - este unul dintre cele mai populare web-site-uri de anunțuti gratuite. Folosește MongoDB pentru a stoca miliarde de înregistrări. Pentru această aplicație web s-a folosit inițial MySQL. Migrarea către MongoDB le-a permis schimbarea design-ului pentru a obține o mai bună scalabilitate.

- Foursquare - este o rețea de socializare bazată pe locație. Stochează informații despre locațiile pe hartă a punctelor de interes (restaurante, cafenele, etc.) și informații despre utilizatorii care vizitează aceste locații. Utilizează MongoDB pentru stocarea datelor.

- CERN - Renumitul laborator de fizica particulelor localizat în Geneva utilizează MongoDB pentru agregarea datelor necesare pentru experimentul „Large Hadron Collider”. Rezultatele interogărilor, efectuate pe cantități masive de date sunt stocate cu ajutorul MongoDB pentru o eventuală utilizare în viitor.

2.2 Conceptele MongoDB

Un server MongoDB găzduiește mai multe baze de date. Bazele de date pot fi comparate cu niște containere și sunt independente unele de altele. O bază de date MongoDB conține una sau mai multe colecții. O colecție reprezintă un set de documente. Acest concept este analog din punct de vedere logic cu structurarea bazelor de date relaționale. Spre deosebire de tabelele din bazele de date relaționale, în MongoDB nu trebuie să definim structura datelor ce urmează să fie stocate în prealabil. Un document stocat într-o colecție este o unitate de date.

În MongoDB, obiectul principal se numește document. Documentele nu au o schemă predefinită, cum este în cazul tabelelor din bazele de date relaționale. Un document este asemănător cu un tablou multidimensional. Într-o matrice putem avea un set de chei care mapează valorile. Valorile pot fi ele însele un alt tablou. Documentul MongoDB este o matrice JSON.[8]

Un document conține un set de câmpuri sau perechi de tip cheie-valoare. Cheile sunt șiruri de caractere, iar valorile pot fi de mai multe tipuri: șiruri de caractere, numere întregi, numere reale, tipuri de date timestamp, etc. Se poate stoca până și un întreg document ca valoare a unui câmp într-un alt document.

2.3 Anatomia unui document din MongoDB

Exemplul de mai jos stochează date despre un utilizator într-o aplicație web: {

_id : ObjectId("4db31fa0ba3aba54146d851a")

username : "joegunchy"
email : "joe@mysite.org"
age : 26

is_admin : true
created : "Sun Apr 24 2011 01:52:58 GMT+0700 (BDST)"

}

Acest document are șase câmpuri. Organizarea datelor este identică cu sintaxa JSON sau JavaScript Object Notation. Valoarea primului câmp, _id, este generat într-un mod automat. MongoDB generează automat un Objectid pentru fiecare document creat într-o colecție și asignează valoarea acestuia câmpului _id al documentului.

Acest id este unic, ceea ce înseamnă că oricare două documente aflate în aceeași colecție nu vor avea aceleași valori pentru câmpul _id, acest lucru semanând cu conceptul de cheie primară din bazele de date relaționale. Următoarele două câmpuri, username și email conțin date de tipul string(șir de caractere), câmpul age conține o valoare de tip integer, is_admin conține date de tip boolean. Ultimul câmp stochează date de tipul JavaScript Date Time, reprezentat ca un șir de caractere. [8]

2.4 BSON - formatul de date de schimb în MongoDB

Am observat mai sus că structura unui document imită un obiect JSON. Când stocăm un document în baza de date, acesta este serializat într-un format special binar codificat, cunoscut sub numele de BSON, prescurtarea pentru JSON binar. Avantajul formatului BSON este faptul este mult mai eficient decât formatele convenționale, cum ar fi XML și JSON. [9]

De asemenea, BSON suportă toate tipurile de date suportate de JSON(șiruri de caractere, numere întregi, numere reale, Boolean, vectori, obiecte, NULL), plus unele tipuri de date speciale, cum ar fi expresii regulate, object ID, dată, dată binară și cod. Limbajele de programare cum ar fi PHP, Python, Java au bliblioteci care gestionează conversia de structuri de date specifice limbajului. Acest lucru permite limbajului să comunice cu MongoDB și să manipuleze datele cu ușurință.

2.5 Tipuri de date

MongoDB suportă o gamă largă de tipuri de date pentru valorile din documente. Documentele din MongoDB au un format asemănător cu formatul JSON. Acest format este ușor de înțeles și de parsat. Pe de altă parte, capacitățile formatului JSON sunt limitate, deoarece tipurile de date sunt: null, boolean, numerice, șiruri de caractere, vectori și obiecte. [9]

Chiar dacă aceste tipuri de date sunt suficiente pentru majoritatea aplicațiilor, există anumite tipuri de date care lipsesc și care în anumite aplicații nu pot lipsi. De exemplu, JSON nu suportă

data ca și tip de dată. Nu există mai multe tipuri de date numerice, astfel că nu se poate diferenția un număr întreg de un număr real. Totuși, MongoDB suportă alte tipuri de date esențiale de tipul cheie - valoare. [9]

Tipurile de date suportate de acest format:

null - acest tip poate fi folosit pentru a reprezenta o valoare nulă, dar poate fi folosit și pentru o valoare care nu există

{“x” : null}
boolean - poate fi folosit pentru valori de tipul *true* sau *false* (adevărat sau fals)

{“x” : true}

32-bit integer, 64-bit integer - aceste formate nu pot fi reprezentate în consolă. JavaScript suportă doar numere reprezentate pe 64 de octeți în virgulă mobilă. Așadar, aceste tipuri vor fi convertite în 64 de octeți.

64-bit floating point number - numere reprezentate pe 64 de octeți în virgulă mobilă - toate numerele vor fi de acest tip

{“x” : 3.14}, {“x” : 3}
string - orice șir de caractere de tip UTF-8 poate fi reprezentat de *string*

{“x” : “foobar”}

symbol - acest tip nu poate fi reprezentat în consolă. Dacă din baza de date se va primi un simbol, acesta va fi convertit într-un string.

object id - un id obiect este un ID pentru documente reprezentat pe 12 octeți {“x” : ObjectId()}

date - datele sunt stocate sub forma milisecundelor de la începutul epocii. {“x” : new Date()}

expresii regulate - documentele pot conține expresii regulate, utilizând sintaxa JavaScript {“x” : /foobar/i}

cod - documentele pot conține cod JavaScript {“x” : function() {/* ... */} }

date binare - datele binare sunt șiruri de octeți arbitrare. Nu pot fi manipulate utilizând consola.
valoare maximă - BSON conține un tip special de date care reprezintă cea mai mare valoare posibilă. Consola nu poate reprezenta acest tip.

valoare minimă - BSON conține un tip special de date care reprezintă cea mai mică valoare posibilă. Consola nu poate reprezenta acest tip.

undefined - această valoare poate fi folosită în documente. JavaScript are un tip distinct pentru *null* și *undefined.*

{“x” : undefined}
array - tablou - seturile de date sau listele de date pot fi reprezentate utilizând acest tip de date

{“x” : [“a”, “b”, “c”]}
embedded document - documentele pot conține alte documente

{“x” : {“foo” : “bar”}}

Numere

JavaScript are un tip de date numit „number” (număr). Din cauză ca MongoDB are 3 tipuri de numere (întregi pe 4 octeți, întregi pe 8 octeți și numere reale pe 8 octeți), consola trebuie să găsească o modalitate de a ocoli limitările impuse de JavaScript. Prin definiție, orice număr scris în consolă este tratat ca și un număr *double* de către MongoDB. Acest lucru înseamnă ca dacă din baza de date este întors un întreg reprezentat pe 4 octeți și apoi va fi transformat într-un număr reprezentat în virgulă mobilă.

Data

În JavaScript, obiectul Date este utilizat pentru tipul de data MongoDB. Pentru a crea un nou obiect de tip *Date* folosim *new Date()*. Apelând constructorul pe post de funcție este returnat o reprezentare a datei, nu obiectul *Date* în sine.

Arrays - tablouri

Valorile conținute de vectori pot fi de mai multe tipuri - șiruri de caractere, numere, etc. Într-un array poate fi stocat un alt array. Un avantaj pe care îl are un vector în MongoDB este că acest sistem de baze de date știe să înterpreteze și să înțeleagă structura lui.

Embedded documents

Documentele încorporate sunt documente MongoDB întregi, care sunt folosite ca și valoare pentru o cheie într-un alt document. Ele pot fi folosite pentru a organiza datele într-un mod mult mai natural.

_id și ObjectIds

Fiecare document stocat în MongoDB trebuie să aibă o cheie „_id”. Valoare acestei chei poate fi de orice tip, dar predefinit este *ObjectId*. Într-o singură colecție, fiecare document trebuie să aibă o valoare unică pentru „_id”, acest lucru asigurând că fiecare document poate fi identificat în mod unic.

ObjectId este tipul implicit pentru câmpul „_id”. Este mult mai ușor să folosim acest tip de cheie, în detrimentul cheilor tradiționale primare, deoarece nu se mai consumă timp pentru sincronizarea tuturor cheilor autoincrementabile între mai multe servere. [9]

2.6 Operații folosite de MongoDB pentru lucrul cu consola

Cele 4 operații clasice cu ajutorul cărora manipulăm și vizualizăm date în consolă sunt: crearea, citirea, actualizarea și ștergerea.

Crearea documentelor

Funcția *insert* adaugă un document la o colecție. De exemplu, dacă am dori să stocăm un articol, mai întâi se creează o variabilă locală, numită *articol.* Această variabilă va fi un obiect de tip JavaScript, care va reprezenta documentul. Câmpurile acestui document vor fi: „titlu”, „conținut” și „dată”.

articol = {“titlu” : “Articol”,
“continut” : “Acesta este articolul din ziar,” “data” : new Date()}

{“titlu” : “Articol”,
“continut” : “Acesta este articolul din ziar.”,
“date” : “Sat Jun 12 2013 11:23:21 GMT - 0500 (EST)”

}
Acest obiect este un document valid MongoDB, deci poate fi stocat în colecția *„ziare”,* folosind

metoda *insert*: *db.ziare.insert(articol)*

Articolul a fost salvat în baza de date. Putem să o vizualizăm folosind metoda *find*:

db.ziare.find()

{
"_id" : ObjectId("4b23c3ca7525f35f94b60a2d"),
"title" : "Articol",
"content" : "Acesta este articolul din ziar.",
"date" : "Sat Jun 12 2013 11:23:21 GMT - 0500 (EST)"

}

Câmpul „_id” a fost adăugat în mod automat, iar ordinea celorlalte perechi cheie-valoare a fost păstrată.

Citirea documentelor

Funcția *find* returnează toate documentele dintr-o colecție. Dacă dorim să vizualizăm un singur document dintr-o colecție putem să folosim funcția findOne:

db.ziare.findOne()

{
"_id" : ObjectId("4b23c3ca7525f35f94b60a2d"),
"title" : "Articol",
"content" : "Acesta este articolul din ziar.",
"date" : "Sat Jun 12 2013 11:23:21 GMT - 0500 (EST)"

}

Actualizarea documentelor

Dacă dorim să modificăm articolul de mai sus, putem folosi funcția *update*. Această funcție primește (cel puțin) doi parametrii: primul este criteriul pentru căutarea documentului pe care dorim

să îl modificăm, iar al doilea este noul document. Dacă dorim să adăugăm comentarii la articolul creat mai sus, creem un vector cu comentarii:

articol.comentarii = [ ]
Apelăm funcția *update*, modificând titlul:

db.ziare.update({titlu : "Articol"}, articol)
În momentul de față, noul document conține și câmpul „comentarii". Dacă vom apela funcția *find*,

vom avea următorul rezultat:

db.ziare.find()

```
{
        "_id" : ObjectId("4b23c3ca7525f35f94b60a2d"),
        "title" : "Articol",
        "content" : "Acesta este articolul din ziar.",
        "date" : "Sat Jun 12 2013 11:23:21 GMT - 0500 (EST)" "comentarii" : [ ]

}
```

Ștergerea documentelor

Funcția *delete* șterge permanent din baza de date. Dacă această funcție este apelată fără parametrii, se vor șterge toate documentele din colecție. Putem să ștergem documentele pe baza unor criterii:

db.ziare.remove({titlu : "Articol"})

3. Tehnologii pentru User Interface

3.1 HTML/XHTML

HyperText Markup Language (HTML) este un limbaj de marcare utilizat pentru crearea paginilor web ce pot fi afișate într-un browser (sau navigator). HTML este un format text proiectat pentru a putea fi citit și editat de utilizatori folosind un editor de text simplu. HTML se poate genera direct, utilizând tehnologii de codare din partea serverului cum ar fi PHP, JSP sau ASP [10]. Atunci când este realizat un web-site, cel mai important lucru este crearea legăturilor.

3.1.1 Introducere

La începutul anilor 90 început dezvoltarea unui nou limbaj de programare/marcare. Limbajul a fost creat pentru a oferi o modalitate pentru utilizatori de a afișa diverse informații în browserele web. Ultima versiune majoră pentru HTML a apărut în februarie 2011. În ianuarie 2000 s-au adăugat unele reguli mai stricte pentru HTML 4.01, astfel apărând XHTML (Extensible Hypertext Markup Language).

HTML și XHTML definesc un set de reguli pentru marcarea documentelor, dar și pentru modul structurării informațiilor.

3.1.2 Structura unei pagini HTML

Când browser-ul primește conținut, acesta încearcă să îl proceseze pentru a verifica dacă este un document HTML.Chiar dacă unele elemente sau tag-uri lipsessc, sau dacă nu este bine structurat, browser-ul poate randa conținutul. Chiar dacă un document este valid (respectă regulile HTML/ XHTML), el trebuie să conțină un număr de elemente structurate cu un conținut adecvat. Un document, pentru a fi valid, trebuie să conțină următoarele informații:

- declarația tipului de document
- tag-ul <*html*>
- în tag-ul <*html*>, tag-ul <*head*>
- în tag-ul <*head*> trebuie să existe elementul <*title*>, adrese URL necesare, script-uri folosite și elementele meta
- după închiderea tag-ului </*head*>, se deschide tag-ul <*body*>, care va conține toate informațiile vizibile utilizatorilor
- în cadrul tag-ului <*body*> putem să definim elemente bloc, putem să structurăm informația după bunul plac. [11]

3.1.3 Tag-uri, elemente și atribute

HTML/XHTML au definite o serie de elemente, fiecare dintre ele încadrându-se într-un anumit domeniu semantic, având un nume derivat sau împrumutat din limba engleză. Elementele definesc structura documentului și pune bazele pentru prezentarea și manipularea acestuia.[11]

Fiecare element de referință dintr-un document este încadrat între două tag-uri - token-uri între paranteze unghiulare (< >), conținând numele elementului. folosit. Tag-urile de deschidere încep întotdeauna cu simbolul „ <”, urmat de numele elementului, de referințele atributelor sau valori asociate elementului.

Atributele sunt folosite pentru a furniza informații suplimentare despre elementul pentru care sunt definite. Toate atributele sunt alcătuite din două părți: un nume și o valoare.

<a href=”http://www.Google.com” target=”_blank”>

- numele este proprietate elementului pe care dorim să o setăm. În exemplul de mai sus, href este numele proprietății pe care am definit-o.
- valoarea este setată pentru atributul definit. În exemplul de mai sus, adresa URL este valoarea pentru atributul href.

Atributele cele mai utilizate în paginile web sunt cele din categoria Core: class, id, title, style. Atributul id estte folosit pentru a asocia un identificator unic unui element într-o pagină web.

Putem asocia acestui id diferite evenimente de tip JavaScript sau proprietăți de tip CSS. Sintaxa pentru a asocia un id este id = ”string”.

Atributul class este folosit pentru a arăta cărei clase de elemente aparține. Dacă avem mai multe elemente de tip <p> care vor primi aceleași proprietăți CSS, atunci putem să atribuim o clasă pentru a fi mai ușor de stilizat. Pentru a atașa o clasă unui element vom folosi următoarea sintaxă: class = ”numeClasa”. Un element HTML poate aparține mai multor clase, numele lor fiind despărțite prin spațiu la definirea lor. [11]

Atributul *title* este folosit pentru a denumi un element HTML. Sintaxa este: title = ”string”. Comportamentul acestui atribut este diferit în funcție de elementul căruia îi este atribuit. De cele mai multe ori titlul apare sub forma unui tooltip sau apare atunci când se încarcă elementul. Nu toate elementele necesită atributul title.

Atributul *style* permite specificarea proprietăților inline de CSS. Sintaxa este <p style=”font-family:Arial, color:black”>Text<p>. Este de evitat acest atribut, deoarece este marcat ca și deprecated în XHTML 1.0, acest lucru însemnând că nu va mai putea fi folosit în versiunile ulterioare ale HTML-ului și XHTML-ului.[11]

Cele mai utilizate categorii de tag-uri în HTML sunt:

Listele - această categorie conține 3 elemente: listele neordonate (în care elementele vor apărea cu bullet-uri), liste ordonate (elementele vor apărea având numere sau litere înainte) și listele cu definiții (permit specificarea termenului, iar apoi definiția acestuia)

Dacă dorim afișarea unei liste neordonate, vom folosit tag-ul <ul>. Fiecare informație din listă va fi încadrată între tag-urile <li>,. Pentru listele ordonate vom folosi tag-ul <ol>, iar pentru listele definite vom folosi <dl>. Listele definite vor avea ca elemente tag-urile <dt> și <dl>. Pentru a defini termenul vom folosi tag-ul <dt>, iar pentru a scrie definiția termenului vom folosi tag-ul <dd>.[11]

Elemente grupate - Pentru a structura mai ușor informația sau pentru a crea secțiuni în document putem sa utilizăm tag-urile și . Aceste elemente nu afectează modul în care o pagină web este afișată, dar sunt folosite pentru a atașa proprietăți cu ajutorul CSS-ului. Tag-ul este utilizat pentru a grupa blocuri de alte elemente, secțiuni, iar tag-ul este utilizat pentru a grupa elemente inline (în linie). se va folosi pentru a grupa elemente în interiorul tag-ului <p>.

Paragrafele - Sunt utilizate pentru a marca un text mai lung. Acest element are câteva proprietăți predefinite de CSS pentru identare.

Link-uri - Un hyperlink este un cuvânt, sau un gruup de cuvinte, sau o imagine, care atunci când este acționat evenimentul de click va trimite utilizatorul către altă pagină sau document sau poate downloada fișiere. Când cursorul este peste un link, atunci el va avea forma unei „mâini”. Atributul href este foarte important pentru acest element, deoarece el va indica destinația utilizatorului la acțiunea de click. Sintaxa este: Click aici . Un alt atribut folosit împreună cu acest element este target. Acest atribut specifică unde să deschidă documentul linkuit.

Headings - Motoarele de căutare folosesc aceste tag-uri pentru a indexa structura și conținutul unei pagini web. H1 este cel mai mai mare heading, iar H6 este cel mai mic.

Imagini - În documente HTML imaginile sunt introduse cu ajutorul tag-ului <img>. Pentru a fi afișată o imagine într-o pagină web trebuie să completăm câmpul src(source - sursă). Acest atribut va lua valoarea URL-ului unde se află imaginea pe care vrem să o afișăm.

Tabele - Un tabel este format din rânduri (<tr>), iar fiecare când este format din celule (<td>). Un tabel se definește cu ajutorul tag-ului <table>.

Elemente de formular - Formularele HTML trimit datele către un server. Un formular de acest tip poate să conțină câmpuri pentru text, checkbox-uri, butoane radio, buton pentru submit.

3.2 CSS

3.2.1 Introducere

CSS - Cascading Style Sheets - este un standard folosit pentru a stiliza și formata elementele dintr-o pagină web. Acest lucru se întâmplă deoarece CSS se află într-o strânsă legătură cu DOM-ul (Document Object Model). Folosind CSS putem modela ușor și rapid orice element. Stilizarea unui element se poate face în interiorul paginii web adăugând atributul *style,* dar se poate face și prin gruparea tuturor stilurilor într-un singur fișier cu extensia *.css.* Gruparea acestor informații reduce cantitatea de cod care poate fi duplicate pe pagini, poate fi mai ușor de menținut. Adăugarea unui fișier cu stiluri într-o pagină web se realizează:

<link rel = ”stylesheet” type = ”text/css” href = ”style.css”>

Pentru a identificaa elementele din pagina web, ne folosim de id-uri, clase și tipurile elementelor. Atunci când utilizăm i*d-uri* folosim sintaxa: *#id { font-style:italic; }*, iar în cazul claselor flosim *.clasa { font-style:italic; }.* Pentru identificarea id-urilor/claselor utilizăm simbolurile „#”, respectiv „ . ”.

Regula CSS este o instrucțiune, sau un grup de instrucțiuni care transmite browser-ului cum să randeze/stilizeze un anumit element din pagină. Fiecare instrucțiune în CSS începe cu *selectorul,* acesta fiind elementul asupra căruia se va aplica regula CSS. Toate proprietățile care trebuie asignate elementului se vor afla între simbolurile {} care vor fi scrise după selector. Instrucțiunea se va termina cu simbolul „ ; ”. Acoladele sunt utilizate pentru a separa multiple instrucțiuni de tip CSS.

3.2.2 Selectori în CSS

Selectorul *tip* - acest selector acționează asupra tuturor elementelor de același tip. De exemplu dacă avem *p {color:black},* atunci toate elementele de tip p (paragraf) vor avea culoarea fontului neagră.

Selectorul de tip descendent - acest selector permite aplicarea stilurilor asupra unor elemente conținute de alte elemente. De exemplu, următoarea linie de cod va afecta toate link-urile conținute de elemente de tip paragraf: *p a {color.red}*.

Selectorul de tip copil - acest tip de selector seamănă cu selectorul de mai sus, de tip descendent, dar este mult mai strict atunci când se aplică regulile. Va selecta doar elementele care sunt copiii direcți ai unor elemente.

Selectorul de tip „înrudit" - acest selector se aplică pentru toate elementele care se află pe același nivel cu cel selectat și care sunt în imediata apropiere. Ex: *p + a {color:red}*. Acest selector va colora toate paragrafele în roșu, doar dacă sunt urmate în cod de elemente de tip link.

Selectorul *id* - id-ul poate fi folosit o singura dată într-un document web. În CSS ne putem referi direct la elementul dorit, dacă acesta are setat un id.

Selectorul class - mai multe elemente dintr-o pagină web pot avea aceeași clasă, iar pentru a seta proprietăți pentru acest grup de elemente folosim selectorul de tip clasă.

Selectorul atribut - multe elemente din HTML suportă atribute, astfel că putem să selectăm elemente pe baza acestor proprietăți. Ex.: *[type = "submit"] {width:100px;}*

Selectorul universal - pentru a stabili un anumit stil pentru toate elementele din pagină, vom folosi acest selector - *. [11]

3.2.3 Proprietăți CSS

Stiluri pentru background

- background-color - definește culoarea fundalului unui element.
- background-image - specifică imaginea care va fi utilizată pe post de fundal.
- background-repeat - în mod normal, imaginea de pe fundal este repetată atât orizontal, cât și vertical. Dacă dorim ca o imagine să fi repetată doar orizontal, sau doar vertical, vom folosi aceasă proprietate.
- background-position - vom folosi această proprietate dacă dorim ca o imagine de fundal să aibă o poziție anume în pagina web.

Fonturi

- font-family - fontul unui text poate fi stabilit prin aceasta proprietate. Putem alege un anumit font, dar de asemenea, putem alege o familie de fonturi.
- font-style - această proprietate pentru a transforma textul în text *italic*.
- font-size - dimensiunea unui font se stabilește prin această proprietate. Mărimea fontului poate fi relativă sau absolută. Mărimea absolută stabilește o dimensiune fixă, iar mărimea relativă stabilește dimensiunea fontului în funcție de elementele vecine și permite utilizatorului să o

schimbe în browser. Dacă nu este specificată aceastp proprietate, ea va lua o valoare predefinită - 16px.

• font-weight - se folosește pentru a îngroșa textul. Stiluri pentru text

• color - este folosită pentru a seta culoarea unui text. Această proprietate poate primi una din următoarele tipuri de valori: o valoarea hexazecimală (*#fff000*), do valoare de tip RGB (*rgb(255,0,0)*) sau numele unei culori (*red*).

• text-decoration - se folosește pentru a elimina sau adăuga elemente decorative într-un text, cum ar fi linia pentru subliniat care apare la link-uri.

• spacing

• text-align - este folosită pentru a alinia orizontal textul. Poate primi următoareale valori: *center, left, right, justify.*

• text-ransformation - se va folosi atunci când dorim să specificăm dacă un text conține numai majuscule sau miniscule.

• text-indent - este folosită pentru a indenta prima linie dintr-un text. Poziționarea elementelor

• absolute - un element poziționat absolut va fi poziționat relativ față de primul părinte. Părintele nu trebuie să aibă setata proprietatea de poziționare statică.

• relative - poziționarea relativă a unui element.

• fixed - un element care va avea această proprietate va avea o poziționare fixă în pagina web.

Pseudo-clase

• :first-letter - se va aplica un stil pentru prima literă a textului conținut de element.

• :first-child - se va aplica un anumit stil pentru primul copil al elementului.

• :link - se folosește pentru a adăuga un anumit stil pentru link-urile care nu au fost vizitate.

• :visited - se folosește pentru a adăuga un anumit stil pentru link-urile care au fost vizitate.

• :hover - se folosește pentru a adăuga un anumit stil pentru elemente atunci când mouse-ul este deasupra lor.

• :active - se folosește pentru a adăuga un anumit stil pentru link-uri, atunci când sunt accesate.

- :focus

Stiluri pentru liste

- list-style-type - dacă dorim să avem alt simbol decât *bullet-ul* elementelor dintr-o listă neordonată, vom folosi această proprietate.

- list-style-image - putem seta ca și *bullet* o imagine customizată.

Pseudoelemente

- :before - poate fi folosit pentru a insera conținut înainte de elementul selectat.
- :after - poate fi folosit pentru a insera conținut după elementul selectat. Margini și identare

- margin - setează distanța elementului față de elementele vecine. Această proprietate nu are culoare, ea fiind transparentă. Valoarea dată va fi în pixeli.

- padding - indentează textul față de chenarul imaginar al elementului. Această proprietate esrte afectată de culoarea background-ului, dacă aceasta este setată.

Chenar

- border - putem seta dimennsiunea unui chenar, tipul și culoarea. • border-color - se setează doar culoarea unui chenar.

- border-style - se setează stilul chenarului.

- border-width - se setează dimenssiunea chenarului.

Dimensiunile unui element

- width - stabilește lățimea unui element.

- height - stabilește înălțimea unui element.

- min-width - setează o lățime minimă a unui element

- min-height - setează o înălțime minimă a unui element • max-width - setează o lățime maximă a unui element

- max-height - setează o înălțime maximă a unui element

3.3 JavaScript

În prezent, există mai multe limbaje de programare, dar JavaScript este de departe cel mai comun limbaj de programare utilizat pentru crearea paginilor web. JavaScript aduce o funcționalitate dinamică a site-urilor web.

JavaScript a apărut pentru prima dată în browser-ul web Netscape Navigator în 1995. Deseori, JavaScript este confundat cu limbajul Java. Numirea a fost doar un truc de marketing pentru a aduce un beneficiu noului limbaj de programare, beneficiind de popularitatea pe care limbajul Java o avea deja.

JavaScript a crescut și mai mult în popularitate atunci când elementele HTML au căpătat definiție mai formală, fiind structurate în DOM(Document Object Model). Datorită faptului că JavaScript și PHP au o sintaxă ce se bazează pe cea a limbajului C, ele seamănă foarte mult. JavaScript oferă dezvoltatorilor web un limbaj de programare care poate fi utilizat pentru a executa sarcini precum:

- citirea elementelor din documente și scrierea unor noi elemente în pagina web;
- manipularea textului;
- efectuarea unor calcule matematice;
- atașarea unor evenimente elementelor din pagina web;
- determinarea dimensiunilor ecranului utilizatorului, determinarea versiunii sau rezoluția browser-ului;
- efectuarea unor acțiuni bazate pe condiții, cum ar fi alertarea utilizatorului în cazul în care acesta introduce informații greșite într-un formular.

În limbajele de programare orientate pe obiecte, obiectele din viata reală sunt reprezentate/ modelate, folosind un set de obiecte care formează un model obiect. Fiecare metodă descrie o acâiune care poate fi facută asupra unui obiect.

Șirurile de caractere pot exista în JavaScript dacă sunt cuprinse între ghilimele simple sau duble. Vectorii sunt similari cu cei din PHP,, astfel că ei pot conține atât valori, cât si alți vectori.

Pentru a accesa diferite obiecte din DOM, JavaScript separă obiectele, proprietățile și metodele folosind simbolul „ . ”.

La fel ca și în cazul PHP-ului, JavaScript oferă acces la funcții și la obiecte. Utilizarea și sintaxa sunt, de asemenea, destul de asemănătoare cu cele din PHP. Deși avem acces la zecile de funcții predefinite, ne putem crea și funcțiile proprii. [11]

O funcție se definește astfel:

<script type="text/javascript" src="http://someserver.com/script.js"> </script>

Variabilele în JavaScript trebuie să respecte următoarele reguli:

- numele variabilelor pot conține doar literele a-z, A-Z, simbolul „$” și simbolul „_”;
- primul caracter al numelui variabilei poate fi doar literă sau simbolul $;
- numele sunt *case-sensitive;*
- nu există o limită stabilită pentru lungimea unei variabile.

Când o variabilă este declarată într-o funcție, ea poate fi accesată doar în interiorul acestei funcții. După apelarea funcției, variabila se distruge. Aceste variabile se numesc locale. Dacă o variabilă locală este declarată folosind cuvântul cheie *var* în interiorul funcției, ea va ocupa memorie atât timp cât funcția rulează. Dacă o variabilă este declarată în afara unei funcții, ea poate fi accesată de orice meoda/funcție din pagina web. Durata de viață a acestor variabile începe atunci când sunt declarate și se termină atunci când pagina este închisă. Variabilele locale ocupă mai puțină memorie și resurse decât cele globale, deoarece au nevoie de memorie doar în momentul în care funcția se execută.

Șirurile de caractere pot exista în JavaScript dacă sunt cuprinse între ghilimele simple sau duble. Vectorii sunt similari cu cei din PHP,, astfel că ei pot conține atât valori, cât si alți vectori.

Pentru a accesa diferite obiecte din DOM, JavaScript separă obiectele, proprietățile și metodele folosind simbolul „ . ”.

La fel ca și în cazul PHP-ului, JavaScript oferă acces la funcții și la obiecte. Utilizarea și sintaxa sunt, de asemenea, destul de asemănătoare cu cele din PHP. Deși avem acces la zecile de funcții predefinite, ne putem crea și funcțiile proprii. [11]

O funcție se definește astfel:

function nume_functie([parametru [,...]]) { instrucțiuni}

Definirea unei funcții începe cu cuvântul *function*, urmat de numele acesteia pe care programatorul îl atribuie. Numele trebuie să înceapă cu o literă sau cu simbolul „_”. Parantezele sunt obligatorii chiar dacă nu există parametrii. Dacă avem mai mulți parametrii, aceștia vor fi separați prin virgulă. În JavaScript există o convenție privind numele funcțiilor: dacă numele este format din mai multe cuvinte, fiecare literă de început al cuvântului va fi o majusculă, excepție făcând prima literă a numelui funcției. Funcțiile sunt folosite pentru a efectua anumite operații asupra datelor.

4. Realizarea unei aplicatii WEB

Dacă se dorește creare unui website care va lucra cu baze de date, atunci avem nevoie de un limbaj de programare de tip backend. În această lucrare vom exemplifica acest proces folosind limbajul PHP, iar bazele de date folosite vor fi cele de tip MongoDB.

MongoDB se instalează în Ubuntu astfel: [12]

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv
7F0CEB10

#

echo 'deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart
dist 10gen' | sudo tee /etc/apt/sources.list.d/mongodb.list

sudo apt-get update

sudo apt-get install mongodb-10gen
```

După ce instalarea s-a terminat, serviciul de MongoDb se pornește și se oprește astfel:

```
sudo service mongodb start

sudo service mongodb stop
```

Realizarea conexiunii dintre PHP și MongoDB se realizează astfel: [12]

```
sudo pecl install mongo
```

Se adaugă linia extension=mongo.so in fișierul php.ini:

```
sudo -i

echo 'extension=mongo.so' >> /etc/php5/apache2/php.ini
```

Restartarea web server-ului: [12]

```
php -i |grep "mongo"

php --re mongo
```

Realizarea conexiunii MongoDB la PHP este similară cu orice altă conectare. Host-ul default este localhost, iar portul este 27017.

```
$connection = new Mongo();
```

Crearea si selectarea unei colecții se realizează astfel:

```
$collection = $dbname->collection;
```

sau

```
$collection = $dbname->selectCollection('collection');
```

Crearea unui array, trimiterea lui către funcția de insert se face astfel:

```
$post = array(
        'title'     => ‘Realizarea unei aplicații web',
        'content'   => ‘Această lucrarea prezinta modul in care o aplicatie web
se realizeaza folosind PHP si MongoDB',
        'saved_at'  => new MongoDate()
    );
    $posts->insert($post);
```

Structura proiectului poate fi de genul acesta: [12]

```
    |-------admin
            |------index.php
            |------auth.php
    |-------view
            |------admin
                    |-------create.view.php
                    |-------dashbroad.view.php
            |------index.view.php
            |------layout.php
            |------single.view.php
    |-------config.php
    |-------app.php
    |-------db.php
    |-------index.php
    |-------single.php
    |-------static
                |-----css
                |-----js
    |-------vendor
                |----markdown
```

Fișierul config.php este fișierul care deține regulile de a ne conecta la baza de date. Aici se pot găsi date precum numele bazei de date, username-ul și parola pentru accesul la baza de date:

```
<?php
```

```
        define('URL', 'http://duythien.dev/sitepoint/blog-mongodb');
        define('UserAuth', 'raluca');
        define('PasswordAuth', 'raluca');
        $config = array(
            'username' => 'raluca',
            'password' => 'qazwsx2013@',
            'dbname'   => 'blog',
            'connection_string'=> sprintf('mongodb://%s:%d%s','127.0.0.1','27017','blog')
        );
?>
```

app.php

```
<?php
    include 'config.php';
    include 'layout.php';
    include 'db.php';
    use Blog\DB,
        Blog\Layout;
   $db = new DB\DB($config);
   $layout = new Layout\Layout();
?>
```

admin.php

```
<?php
require_once 'auth.php';
require_once '../app.php';
if ((is_array($_SESSION) && $_SESSION['username'] == UserAuth)) {
    $data = array();
    $status = (empty($_GET['status'])) ? 'dashboard' : $_GET['status'];
    switch ($status) {
        case 'create':
            break;
        case 'edit':
```

```
            break;
        case 'delete':
        default:
            $currentPage = (isset($_GET['page'])) ? (int)$_GET['page'] : 1;
            $data = $db->get($currentPage, 'posts');
            $layout->view('admin/dashboard', array(
                'currentPage' => $data[0],
                'totalPages' => $data[1],
                'cursor' => $data[2]
            ));
            break;
    }
}
?>
```

db.php

```
<?php
namespace Blog\DB;
class DB
{
    private $db;
    public $limit = 5;
    public function __construct($config)
    {
        $this->connect($config);
    }
   private function connect($config)
    {}
    public function get($page, $collection)
    {}
    public function getById($id, $collection)
```

```
    {}
    public function create($collection, $article)
    {}
    public function delete($id, $collection)
    {}
    public function update($id, $collection, $acticle)
    {}
   public function commentId($id, $collection, $comment)
    {}
}
?>
```

Index.php:

```
<?php
require 'app.php';
try {
    $currentPage = (isset($_GET['page'])) ? (int)$_GET['page'] : 1;
    $data = $db->get($currentPage, 'posts');
    $layout->view('index', array(
        'currentPage' => $data[0],
        'totalPages' => $data[1],
        'cursor' => $data[2],
        'name' => 'Duy Thien'
    ));
} catch (Exception $e) {
    echo 'Caught exception: ', $e->getMessage(), "\n";
}
?> [12]
```

În fiecare pagină de tip PHP se pot introduce elemente de tip HTML, CSS și chiar JavaScript. Aceste elemente sunt folosite pentru a crea diverse design-uri dorite, dar și pentru a le stiliza.

Bibliografie

1. http://en.wikipedia.org/wiki/Web_application
2. http://ro.wikipedia.org/wiki/PHP - accesat la data 1.02.2015
3. YANK K., PHP & MySQL: Novice to Ninja, Fifth Edition, SitePoint Pty. Ltd., United States of America, 2012, ISBN: 978-0-9872478-1-0
4. http://ro.wikipedia.org/wiki/POO - accesat la data 1.02.2015
5. TATROE K., MACINTYRE P., LERDORF R., Programming PHP, Third Edition, O'Reilly Media, Inc., United States of America, February 2013, ISBN: 978-1-449-39277-2
6. http://php-html.net/tutorials/model-view-controller-in-php/ - accesat la data 2.02.2015
7. NIXON R., Learning PHP, MySQL, JavaScript, and CSS, Second Edition, O'Reilly Media, Inc., United States of America, August 2012, ISBN: 978-1-449-31926-7
8. FRANCIA S., MongoDB and PHP, First edition, O'Reilly Media, Inc., United States of America, January 2012, ISBN: 978-1-449-31436-1
9. CHODOROW K., DIRELF M., MongoDB: The Definitive Guide, First Edition, O'Reilly Media, Inc., United States of America, September 2010, ISBN: 978-1-449-38156-1
10. http://ro.wikipedia.org/wiki/HyperText_Markup_Language - accesat la data de 3.01.2015
11. DUCKETT J., Beginning HTML, XHTML, CSS, and JavaScript, Wiley Publishing, Inc., Indianapolis, Indiana, 2010, ISBN: 978-0-470-54070-1
12. http://php.net/manual/en/mongo.tutorial.php