UNIVERSITATEA BABEŞ-BOLYAI

Facultatea de Ştiinţe Economice şi Gestiunea Afacerilor

Informatică-Economică

Lucrare de licenţă

Absolvent,

Oana Luisa **ROMAN**

Coordonator ştiinţific,

Lector univ. dr. Daniel **MICAN**

2019

UNIVERSITATEA BABEŞ-BOLYAI

Facultatea de Ştiinţe Economice şi Gestiunea Afacerilor

Informatică-Economică

Lucrare de licenţă

Aplicație web care stochează fișierele media de tip imagine din cadrul diferitelor evenimente - collectIn

Absolvent,

Oana Luisa **ROMAN**

Coordonator ştiinţific,

Lector univ. dr. Daniel **MICAN**

2019

Rezumat

Lucrarea realizează o aplicație web destinată stocării materialelor media de tip imagine. Plecând de la ideea că fotografiile au un loc special în viața fiecărui om, am conceput acest sistem pentru a oferi un spațiu de stocare dedicat pozelor efectuate în cadrul diferitelor evenimente private sau publice la care o persoană participă de-a lungul timpului. Astfel, organizarea lor se face folosind structura unui album virtual, cu detaliile aferente fiecărui eveniment. De asemenea, sistemul informatic conceput oferă și posibilitatea de acordarea accesului și altor utilizatori care au luat parte la eveniment, indiferent de tipul acestuia. Pentru dezvoltarea acesteia s-au folosit diferite tehnologii și limbaje de programare, alese în așa fel încât performanța, eficiența, portabilitatea și accesibilitatea sistemului să fie la capacități maxime.

Cuprins

Lista tabelelor şi figurilor

Tabele:

Figuri:

Introducere

În ziua de astăzi, cu ușurință se poate remarca faptul că oamenii sunt tot mai interesați să participe la diferite evenimente, fie ele culturale, concerte, conferințe ș.a.m.d. O activitate comună întâlnită la orice tip de eveniment este fotografiatul. Majoritatea indivizilor doresc să își păstreze amintirile din cadrul activităților la care iau parte sub formă de fotografii. În general, aceste fotografii sunt distribuite în mediul online sau prin intermediul stick-urilor USB sau al CD-urilor, DVD-urilor, urmând apoi ca fiecare să își aleagă modalitatea cea mai potrivită pentru a le păstra.

Lucrarea de față reprezintă o aplicație web având ca scop principal stocarea materialelor media de tip imagine executate în cadrul anumitor evenimente. Utilizatorii își vor putea păstra, dar în același timp și împărți amintirile cu ceilalți participanți, fără a mai fi nevoie de un mijloc de distribuire a fotografiilor de la o persoană la alta sau de un spațiu personal de stocare a acestora. Având în vedere platformele deja prezente care îți oferă, într-un fel sau altul, aceste facilități, un exemplu cunoscut ar fi Facebook-ul, consider că exclusivitatea acesteia, simplitatea, ușurința organizării și posibilitatea restricționării accesului și a stocării fotografiilor constituie diferențele principale între aplicația mea și celelalte existente.

Motivația mea s-a bazat mai mult pe faptul că eu îmi doresc o platformă în care doar să îmi păstrez pozele organizate într-un mod ușor după evenimentele la care particip. Luând în considerare faptul că succesul aplicației ține și de interesul persoanelor și nevoile acestora de a-și păstra amintirile în format digital, mi-am propus să prezint anumite idei legate de importanța evenimentelor și fotografiilor în viața omului. ”De-a lungul istoriei, în culturile din toată lumea, oamenii s-au adunat să celebreze pentru numeroase diferite motive, iar evenimentele au fost întotdeauna un element central al societății umane.” [1] De asemenea, de sute de ani fotografia are un rol din ce în ce mai însemnat în viața de zi cu zi, ajungând astăzi să fie cea mai folosită tehnică pentru capturarea unui moment și păstrarea amintirilor. În plus, fotografiile stau drept dovadă evenimentelor petrecute de-a lungul timpului, marcând momente importante atât pentru întreaga omenire, cât și pentru fiecare om în parte. Am găsit ”6 motive pentru care fotografia contează:

1. Fotografiile noastre ne spun ceea ce e important pentru noi

Când întrebi oamenii ce bunuri și-ar salva din casa lor care a luat foc, unul dintre cele mei frecvente răspunsuri este albumul cu fotografii sau computerul cu pozele lor digitale. Într-un moment de panică este interesant faptul că probabil vom lua mai degrabă pozele decât

bijuteriile valoroase. Acest impuls pentru a ne salva memoriile înregistrate este o forță puternică care ne spune despre rolul fotografiilor în viața noastră și dorința constantă de a salva cele mai prețioase momente în imagini.

2. Fotografiile sunt parte din moștenirea noastră

Fotografiile contează pentru că ele îngheață momentele din viața noastră care trec neremarcabil și care par să aibă o mică importanță pentru noi în acel moment. Ele pot fi piesele mici care completează imaginea de ansamblu a vieții noastre.

3. Fotografiile ne permit să împărțim și să comunicăm

Imaginile sunt mult mai mult decât o simplă înregistrare. Fotografia vorbește celei mai bune și generoase părți din natura umană – dorința de a împărți cu alții ceea ce noi găsim frumos și interesant. Milioane de oameni își împart fotografiile personale, pasionale și uneori ciudate lumii din jurul lor. Imaginile noastre pot invoca o lume a străinilor în viața noastră. Cât de puternic este acest lucru?

4. Fotografia ne face artiști
5. Fotografia este o limbă complexă
6. Fotografia are puterea de a ne muta" [2]

Așadar, oamenii consideră fotografiile ca fiind un lucru important din viața lor, doresc să îl împartă cu ceilalți și sunt dispuși să facă multe lucruri pentru a le salva. Oare nu le-ar fi mai ușor cu o aplicație care să facă toate acestea?

Pentru dezvoltarea acestei aplicații am folosit diferite tehnologii și limbaje de programare, precum: HTML, CSS și JavaScript, pentru interfața cu utilizatorul, respectiv Node.js, JSON și sistemul de gestiune MongoDB pentru realizarea funcțiilor de bază și gestiunea cu baza de date.

Lucrarea este organizată în modul următor: Capitolul 1 prezintă conceptele teoretice, tehnologiile utilizate și descrie principalele noțiuni, Capitolul 2 specifică cerințele de sistem, sursele acestora și documentarea, Capitolul 3 reprezintă proiectarea și prezentarea generală a sistemului informatic, Capitolul 4 implementează și utilizează aplicația descrisă. Lucrarea se încheie cu concluzii și posibilități de dezvoltare.

1. Concepte teoretice și tehnologii utilizate

Acest capitol va descrie principalele concepte teoretice privind aplicațiile web, noțiuni introductive, tehnologiile utilizate pentru realizarea aplicației și alte sisteme prezente pe piață printr-o comparație de avantaje și dezavantaje.

1.1 Descrierea unei aplicații web

O aplicație web poate fi descrisă ca o aplicație client-server unde browser-ul acționează ca și client, iar server-ul web ca și server. Cu alte cuvinte, o aplicație web este orice program care performează anumite funcții folosind un browser web ca și client. Clientul este cel care introduce datele, în cazul nostru, fotografiile și date pentru autentificare, iar server-ul se ocupă de stocarea acelor date. Printre principalele beneficii ale unei aplicații web este faptul că nu necesită instalare din partea utilizatorului, spațiu de stocare sau anumite sisteme de operare. Pentru a o accesa este nevoie doar de internet și un browser web, care în majoritatea cazurilor se află deja instalat pe orice calculator, telefon mobil sau tabletă. Dezavantajele apar mai mult la partea de server, care necesită o conexiune constantă cu aplicația web pentru ca aceasta să poată să funcționeze în mod corect. În mod obișnuit, aplicațiile web se împart în partea de client, numită și front-end, și partea de server, back-end.

Partea de front-end reprezintă acea parte a aplicației cu care utilizatorul interacționează, aceasta fiind la rândul ei împărțită în 2 părți: design-ul și dezvoltarea interfeței. Prin design ne referim la partea creativă a aplicației, ceea ce se vede, iar dezvoltarea constă în implementarea design-ului și funcționalităților folosind limbaje precum HTML, CSS și JavaScript.

Dezvoltarea back-end-ului se referă la implementarea părții de server, care se axează în primul rând pe logica aplicației sau mai precis, cum funcționează aceasta. Este un proces de creare a nucleului aplicației web, dezvoltării platformei pentru aceasta, completând-o cu toate funcționalitățile necesare. Rolul său este de a administra conținutul și de a gestiona datele primite de la front-end, returnând rezultatul în forma înțeleasă de partea client. Este adesea numit ”interfața pentru administratori”, iar, în cazul de față, este dezvoltată folosind Node.js. În mod normal este compus din 3 părți principale: serverul web, API-ul și baza de date.
”Un server web se poate referi la hardware sau software, sau ambele lucrând împreună.

1. Pe partea de hardware, un server web este un calculator care stochează server-ul web software și componentele care alcătuiesc aplicația. Este conectat la internet și suportă schimbul de date fizice cu alte dispozitive conectate la web.
2. Pe partea de software, un server web include mai multe componente care controlează modul în care utilizatorii accesează fișierele găzduite, la cel puțin un server HTTP. Un server HTTP este un software care înțelege adresele URL și HTTP – protocolul pe care browser-ul îl utilizează pentru a vizualiza paginile web. Acesta poate fi accesat prin numele de domenii ale site-urilor cărora le stochează și le furnizează conținutul pe dispozitivele utilizatorilor finali.

La nivelul cel mai de bază, ori de câte ori un browser are nevoie de un fișier găzduit pe un server web, browser-ul solicită fișierul prin HTTP. Când cererea ajunge la serverul web (hardware), serverul HTTP (software) acceptă solicitarea, găsește documentul solicitat (dacă nu este returnat atunci un răspuns 404) și îl trimite înapoi la browser, tot prin HTTP.” [3] Comunicarea client-server HTTP se poate observa în Figura 1, preluată din [3].

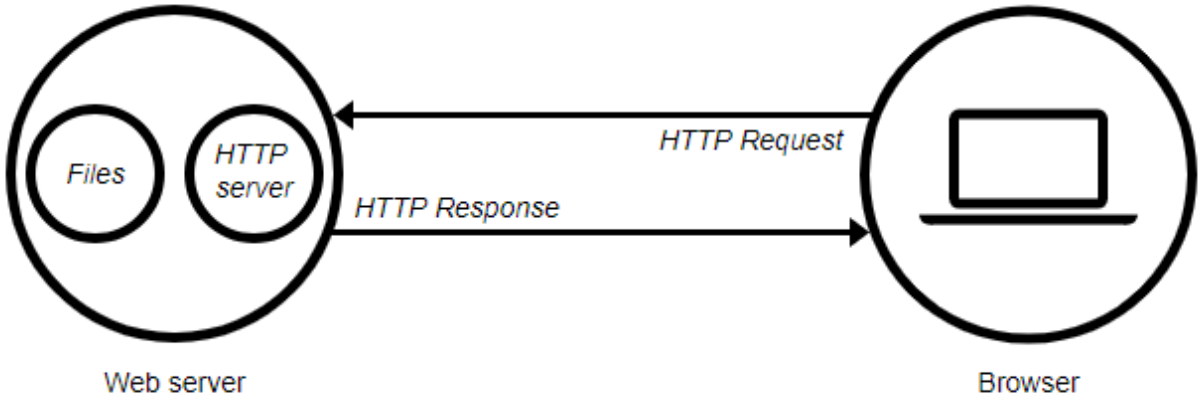

Figura 1. Client-server comunicarea HTTP

Când e vorba să alegi un server web există mai multe opțiuni diferite. Printre cele mai cunoscute sunt: Internet Information Services (IIS) de la Microsoft, nginx de la NGINX, Apache HTTP Server, Google Web Server ș.a.m.d. Următoarea diagramă de la W3techs ne arată procentul de site-uri web care folosesc un anumit tip de server web și este realizată în aprilie 2019.

Figura 2 arată procentele de site-uri web care folosesc servere web diferite, preluată din [4].

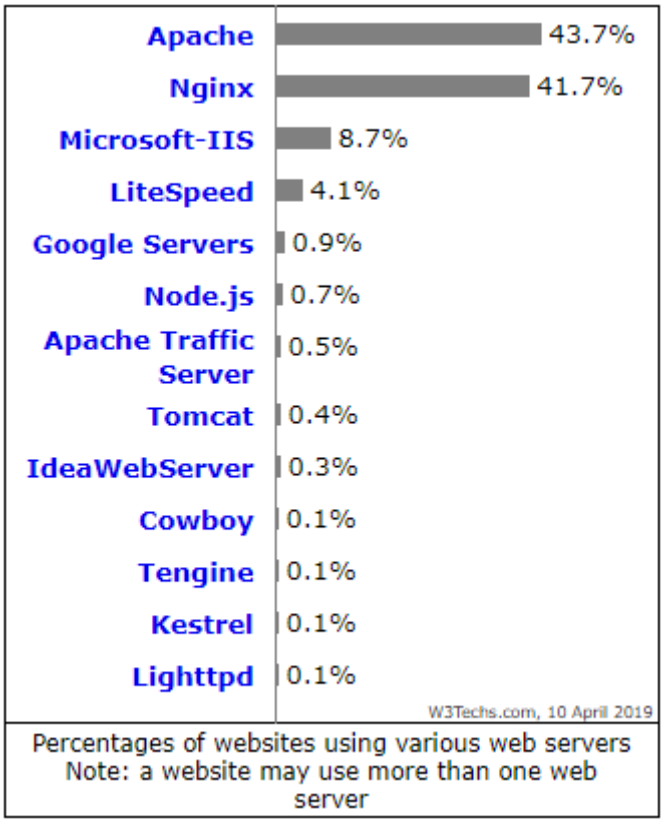

Figura 2. Procentele de site-uri web care folosesc servere web diferite

După cum demonstrează Figura 2, Node.js este folosit de 0,7% din toate site-urile web ale căror server le este cunoscut celor de W3Techs.com. Există diferite aspecte după care alegi ce server web folosești, precum sistemul de operare, nivelul de securitate și puterea de performanță.

API-ul, poate fi văzut ca o poartă către informațiile și conținutul stocat, având acces la baza de date și la fișierele media depozitate pe disk. Aceasta implică toate operațiunile pentru procesarea datelor solicitate și trimise, salvarea datelor în baza de date, realizarea deciziilor privind datele necesare și interogarea datelor necasare din baza de date. De altfel, aceasta permite utilizarea măsurilor de securitate, cum ar fi crearea mecanismelor de autentificare pentru a identifica utilizatorul care solicită sau trimite date de la și către server. Este o parte importantă a dezvoltării back-end-ului, care prescrie modalitățile prin care datele pot fi afișate, create, salvate și manipulate.

Baza de date reprezintă o modalitate de stocare a informațiilor și este manipulată cu ajutorul unui sistem de gestiune a bazelor de date. În societatea modernă, bazele de date au devenit o componentă esențială, fiind accesate în fiecare zi, fie că rezervăm niște bilete de avion, camere de hotel sau realizăm depuneri/extrageri bancare. În ansamblu, o bază de date este o colecție de date corelate din punct de vedere logic, care reflectă un anumit aspect al lumii

reale și este destinată unui anumit grup de utilizatori. Ea trebuie să asigure abstractizarea, integrarea, integritatea, partajarea, securitatea și independența datelor. Există mai multe tipuri, cel mai răspândit este cel relațional, urmând modelul ierarhic, orientat pe obiecte sau XML.

Chiar dacă în prezent aplicația nu va fi nevoită să trateze cantități extrem de mari de date, dar cu o perspectivă în viitor că s-ar putea ajunge la acest lucru, se va folosi o bază de date NoSQL. Aceasta este diferită față de modelul bazelor de date relaționale, gestionând volume mari de date care nu urmează neapărat o schemă fixă și este bine optimizată pentru operații de adăugare sau căutare.

Conform definiției date de Rick Cattell, ”bazele de date NoSQL prezintă șase trăsături de bază:

1. Abilitatea de a scala orizontal pe mai multe servere;
2. Abilitatea de a replica și distribui datele pe mai multe servere;
3. CLI (call level interface) caracterizat prin simplitate (în contrast cu SQL binding);
4. Un model concurențial mai slab decât modelul relațional (ACID);
5. Utilizarea eficientă a indexării distribuite şi a RAM pentru o stocare eficientă;
6. Abilitatea de a adăuga dinamic noi atribute la înregistrările existente.” [5]

Arhitectura „shared nothing” a sistemelor NoSQL reprezintă încă o caracteristică importantă deoarece face ca fiecare nod/server să fie independent, niciunul nepartajând memorie sau spațiu. Astfel, pot fi efectuate un număr mare de operații de scriere sau citire pe secundă.

Există mai multe tipuri de baze de date NoSQL, bazate pe modele de înregistrări cheie-valoare, sub formă de graf sau documente, care stochează toată informația unui obiect într-o singură instanță în baza de date.

Sistemul de gestiune a bazelor de date, prescurtat SGBD, are un rol important în dezvoltarea unei aplicații web, dar și ca parte a back-end-ului. Oferă posibilitatea de salvare, modificare și ștergere a datelor. De asemenea, furnizează modalități de a defini modul în care sunt organizate datele, de a asigura diferite măsuri de securitate, de a menține integritatea datelor, de a adăuga și monitoriza utilizatorii.

Server-ul stochează informațiile obținute sub formă de texte, imagini, profiluri de utilizatori, iar API-ul are acces la baza de date și la fișierele media stocate. Baza de date reprezintă o modalitate de stocare a acestor informații și este manipulată cu ajutorul unui sistem de gestiune a bazelor de date. Datorită faptului că se va lucra cu o bază de date nerelațională, sistemul de gestiune a bazelor de date folosit este MongoDB.

1.2 Tehnologii utilizate

Pe partea de front-end, după cum am văzut în subcapitolul anterior, voi folosi limbaje precum HTML, CSS și JavaScript. Înainte de a le prezenta în detaliu, trebuie să explicăm și noțiunile de World Wide Web, URI.

World Wide Web (Web/www) este o rețea de resurse informatice care se bazează pe 3 mecanisme pentru a pune aceste resurse la dispoziția publicului larg:

1. O secvență alfanumerică pentru localizarea resurselor pe Web (URI)
2. Protocoale pentru accesare resurselor de pe Web (HTTP)
3. Hypertext, pentru o navigare ușoară printre resurse (HTML)

Toate resursele valabile pe Web au o adresă care poate fi codificată de către identificatorul uniform de resurse (URI), deseori identic cu URL-ul resursei – o formă timpurie a identificatorului URI. În mod obișnuit, URI-urile sunt compuse din 3 piese:

a. schema de denumire a mecanismului utilizat pentru a accesa resursa
b. numele mașinăriei care găzduiește resursa
c. numele resursei în sine, dat ca o cale.

În cazul în care o resursă este pe aceeași mașinărie ca și documentul curent atunci URI-ul se numește URI relativ, care nu conține informații despre schema de denumire. Poate conține componente de cale relativă și identificatori de fragmente, aceștia identificându-se după semnul #. În HTML, URI-urile sunt folosite la:

- legarea de un alt document sau resursa
- legarea externă la fișiere de script sau foi de stil
- includerea unei imagini în pagina web
- trimiterea unui formular
- crearea unui document cadru
- citarea unei referințe externe

HTML, HyperText Markup Language, este un limbaj de marcare folosit pentru a publica informații pe o pagină web distribuită global, un fel de limbaj universal pentru toate calculatoarele și limbajul de publicare utilizat de World Wide Web. HTML oferă mijloace pentru a publica documente online cu titluri, texte, tabele, fotografii ș.a.m.d., obține informații online prin intermediul link-urilor, formulare folosite la căutarea informațiilor, rezervărilor, comenzilor online, etc.

De-a lungul anilor, au existat mai multe revizii a standardului HTML, în acest moment, în continuă dezvoltare, fiind HTML5. În principal, obiectivele constau în îmbunătățirea

limbajului, aducerea unor noi caracteristici, precum elemente de video, audio, articole, secțiuni ș.a.m.d, dar bineînțeles și menținerea unui limbaj ușor de citit și înțeles, atât de către oameni, cât și de către browser-ele web, calculatoare etc.

Pentru a crea o pagină HTML trebuie introduse tag-uri sau etichete, iar fișierul să aibă extensia ”.html” sau ”.htm”. Un document HTML trebuie să cuprindă versiunea HTML a documentului, zona de ”cap” a documentului, numită ”head” și corpul fișierului ”body, unde deobicei se află cel mai mult cod care formează pagina. Tag-urile sunt interpretate de browser-ul web, afișând rezultatul pe ecranul dispozitivului.

Ulterior, pentru a oferi un stil extra elementelor unui document HTML, a fost creat CSS-ul, Cascading Style Sheets (Foi de stil în cascadă), care este un standard pentru formatarea acestora. Altfel spus, CSS-ul este un limbaj ce descrie stilul unui document HTML. Se poate realiza prin 3 moduri: în interiorul fișierului HTML, în zona ”head” folosind tag-urile style; în linie, cu ajutorul atributelor unde etichetele permit acest lucru sau într-un fișier extern cu extensia ”.css” atașat documentului. Informațiile legate de stilul elementelor, cum ar fi culoarea unui text, alinierea, dimensiunea, etc. pot fi specificate atât pentru un singur element, cât și pentru grupuri de elemente. Cea mai recentă evoluție este CSS3, un upgrade care vine cu câteva atribute noi și în ajutorul dezvoltărilor noilor concepte de web design. Printre modulele importante adăugate se enumeră: transformările 2D/3D, animațiile, noi proprietăți pentru crearea bordurilor etc.

Există și limbaje care pot fi folosite ca și extensii a CSS-ului adăugând diferite caracteristici precum variabile, amestecuri (mixins), operații aritmetice, funcții, importări și posibilitatea de a scrie codul în formă de ”cuib”, combinat sau în loc de ”cascadă”. Extensia folosită în cazul acestui proiect se numește Less, care vine de la Leaner Style Sheets, o traduce în română ar fi ”Foi de stil mai flexibile” și va fi adăugat în pagina HTML folosind link-urile necesare găsite pe pagina oficială. De asemenea, se vor adăuga și librăriile Bootstrap, care face ca aplicația să se modeleze în funcție de mărimea ecranului dispozitivului folosit de utilizator; DropzoneJS care, din punctul de vedere al afișării, oferă previzualizarea de imagini în browser după ce acestea au fost încărcate, iar din punct de vedere al funcționalității permite încărcarea fișierelor ”drag & drop”, tradus în română ”trage și lasă”; și FancyBox, un instrument de afișare a imaginilor, personalizabil prin setări și CSS.

Introducerea funcționalităților în paginile web are loc folosind limbajul de programare orientat pe obiect JavaScript, prescurtat JS, care de altfel este esențial pentru aplicațiile web moderne. Acesta permite ca conținutul executabil să fie inclus în pagini, transformând o pagină HTML statică în dinamică. Poate include programe care să interacționeze cu utilizatorii, să

controleze browser-ul și să creeze în mod dinamic conținut HTML. Când codul JavaScript se încorporează în browser-ul web rezultatul este numit ”client-side JavaScript”, care ar veni tradus ca ”JavaScript-ul de pe partea client”. Acesta combină capacitatea de scripting a unui interpret JavaScript cu Document Object Model, prescurtat DOM, definit de un browser web, permițând distribuirea conținutul executabil pe Web și aflându-se în centrul unei noi generații de documente HTML dinamice (DHTML).

DOM-ul tratează un document HTML ca o structură de arbore, fiecare nod fiind un obiect care reprezintă o parte a documentului. Cu ajutorul metodelor DOM se poate modifica structura, conținutul sau stilul documentului în mod programat, aceste schimbări fiind apoi vizibile în afișarea documentului.

DHTML reprezintă o tehnologie de construire a paginilor HTML interactive folosind HTML, CSS și JavaScript pentru a putea imprima dinamism. Exemple de aplicare ar fi pentru creare meniurilor, introducerea de grafică animată, verificarea corectitudinii la trecerea într-un alt câmp al unui formular ș.a.m.d. Un avantaj ar fi faptul că permite includerea elementelor formate într-un mod asemănător în toate paginile, evitând greșelile de scriere care pot apărea atunci când acestea trebuie introduse pe fiecare pagină separat.

Printre caracteristicile JavaScript-ului pe partea client se regăsesc: controlul asupra aspectul și conținutul documentului; controlul browser-ului, de exemplu putem să afișăm mesaje scurte utilizatorului folosind ferestre de dialog; interacțiunea cu formularele și utilizatorii; și multe altele. Abilitatea de a interacționa cu formularele HTML și utilizatorii sunt aspecte importante deoarece acestea sunt destul de des întâlnite și în această aplicație.

Pentru a simplifica anumite procese, de exemplu traversarea arborelui DOM în HTML, și a crește interactivitatea, viteza și ușurința de utilizare a aplicației web, se vor folosi Ajax și jQuery, o platformă de dezvoltare JavaScript disponibilă în toate versiunile de browsere importante, care stă și la baza anumitor librării incluse în acest sistem informatic. Ajax este o tehnică de programare care ajută sistemul să fie interactiv, mai rapid și mai ușor de folosit prin schimbul în fundal al unor cantități mici de date cu serverul. Acest lucru face ca pagina pe care utilizatorul o folosește să nu fie nevoie să fie reîncărcată la fiecare acțiune a acestuia.

Randarea datelor din back-end în front-end se face folosind un limbaj de templating numit nunjucks. Nunjucks este un motor de modelare pentru JavaScript care oferă control asincron, performanță ridicată, extensii personalizate și opțiuni detaliate de precompilare.

În ceea ce constă partea de back-end tehnologiile utilizate vor fi Node.js, MongoDB și JSON.

Tehnologia de serializare a datelor numită JSON (JavaScript Object Notation) este, în momentul de față, cel mai comun standard pentru transmiterea informațiilor între aplicații, definind un set de reguli de formatare pentru reprezentarea portabilă a datelor structurate. Aproape orice limbaj sau tehnologie de creare aplicații știe să codeze sau decodeze un set de informații JSON, fiind cel mai potrivit pentru aplicațiile JavaScript.

Documentația oficială [6] definește Node.js ca fiind: ”o platformă construită în Chrome JavaScript runtime, pentru a crea simplu și rapid aplicații în rețea. Node.js folosește un model bazat pe evenimente, fără să blocheze funcțiile de I/O, care permite aplicațiilor care rulează pe diferite dispozitive distribuite să fie eficiente și ușoare, perfecte pentru date intensiv solicitate în timp real.” Node.js este construit peste motorul V8 JavaScript, care, de asemenea, este folosit și în browser-ul Google Chrome și este scris în limbajul de programare C++. Acesta compilează codul JavaScript direct în codul mașinii native fără a fi nevoie de interpretarea codului în timp real. Astfel, Node.js are abilitatea de o execuție foarte rapidă a codului. Întreaga arhitectură a lui Node.js este reprezentată în Figura 3, preluată din [7].

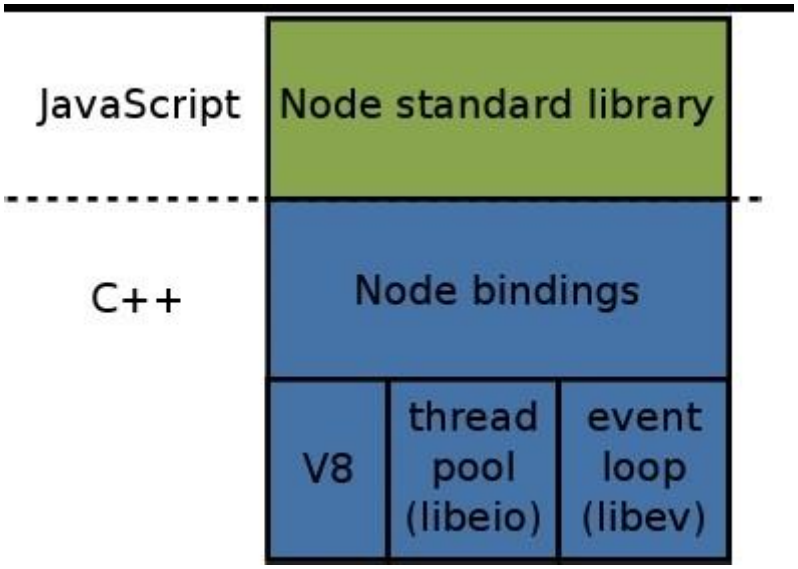

Figura 3. Arhitectura Node.js

După cum se poate observa în Figura 3, Node.js este compus dintr-un număr de componente: librăria Node standard în partea de sus, legături Node, niște legături subțiri C++, la mijloc și motorul V8, libeio și libev în partea de jos. ”Librăria Node standard expune caracteristicile sistemului de operare la aplicație, în timp ce legăturile C++ expun API-urile principale ale elementelor subiacente la JavaScript. Motorul V8 asigură mediul de funcționare pentru aplicație, iar libeio se ocupă de firele de execuție pentru a efectua apeluri asincrone I/O (non-blocante) către libev, bucla evenimentului.”[7] Datorită acestei arhitecturi, Node.js are câteva avantaje principale, printre care: execuția foarte rapidă, explicată mai sus, este asincron

și bazat pe evenimente, ceea ce înseamnă că serverul nu așteaptă pentru un API să returneze date, ci trece la următorul API până ce primește răspunsul de la API-ul precedent, folosește un model cu un singur fir de execuție, permițând extinderea și performanța mult mai bună a aplicațiilor.

Node.js vine și cu un manager de pachete, abreviat NPM, care asigură instalarea pachetelor mai mult ușor prin comanda ”npm install package”. Toate pachetele folosite în cadrul aplicației se găsesc în fișierul node_modules și de asemenea, sunt afișate și în fișierul package.json.

Pentru dezvoltarea aplicației Node.js voi folosi framework-ul Express care conține blocuri principale de construcție și instrumente pentru a rula un server. Se poziționează deasupra acestuia și ajută la: simplificarea operațiilor cu diferite API-uri, definind un standard ușor de implementat și extensibil, adaugă caracteristici noi, facilitează executarea dinamică a paginilor HTML. Un server Express este alcătuit din ruter, rute și middleware, iar funcționalitatea centrală a unui server web depinde de metodele sale de rutare. Express permite crearea rutelor în structuri simple, printr-o combinație din una dintre cele patru metode HTTP: GET, POST, PUT și DELETE și o cale, care reprezintă locația resursei cerute de client, într-o comunicare client-server. Pentru realizarea cererilor HTTP din browser către Node.js, pentru partea de front-end, se va folosi biblioteca JavaScript, Axios.

Funcțiile middleware din Express reprezintă funcțiile care au acces la: obiectul de răspuns (res), obiectul cererii (req) și la următoarea funcție middleware din ciclul de solicitare-răspuns al aplicației (next). Funcțiile Middleware pot executa orice cod, modifică cererea și răspunsul, încheie ciclul de cerere-răspuns prin trimiterea răspunsului și apela următoarea funcție middleware. Dacă ciclul de solicitare-răspuns nu este încheiat de funcția intermediară curentă, trebuie să treacă întotdeauna controlul la următoarea funcție middleware prin apelul "next()". În caz contrar, cererea va fi suspendată. [8]

După cum spuneam, ca și bază de date, voi folosi MongoDB, care este o bază de date NoSQL scrisă în C++. Aceasta stochează datele ca și documente, care deobicei seamănă cu o structură asemănătoare JSON, urmărind modelul cheie-valoare care face ca implementarea să fie realizată cu ușurință, neavând nevoie de o schemă predefinită a datelor. Dacă deobicei suntem obișnuiți cu tabele, în cazul lui MongoDB vom avea colecții care conțin documente (datele), care la rândul lor conțin câmpuri (în loc de rânduri și coloane). Aceste caracteristici aduc diferite facilități în lucrul cu bazele de date MongoDB precum: flexibilitatea migrării și modificării imaginii de ansamblu asupra datelor, nu există nevoia de join deoarece avem încapsulare și legături, posibilitatea stocării fișierelor de mari dimensiuni fără a se crea anumite

complicații, disponibilitatea datelor, suport pentru indexare și mai ales, accelerarea dezvoltării aplicațiilor și reducerea complexității implementărilor.

Pentru această aplicație va fi nevoie și de Mongoose care este o bibliotecă de modelare a datelor obiect (ODM) pentru MongoDB și Node.js. Aceasta gestionează relațiile dintre date, oferă validarea schemei și este folosită pentru traducere între obiectele în cod și reprezentarea acelor obiecte în MongoDB. [9] Figura 4 arată modul în care se realizează asocierea obiectelor între Node.js și MongoDB, administrate de către mongoose și preluată din [9].

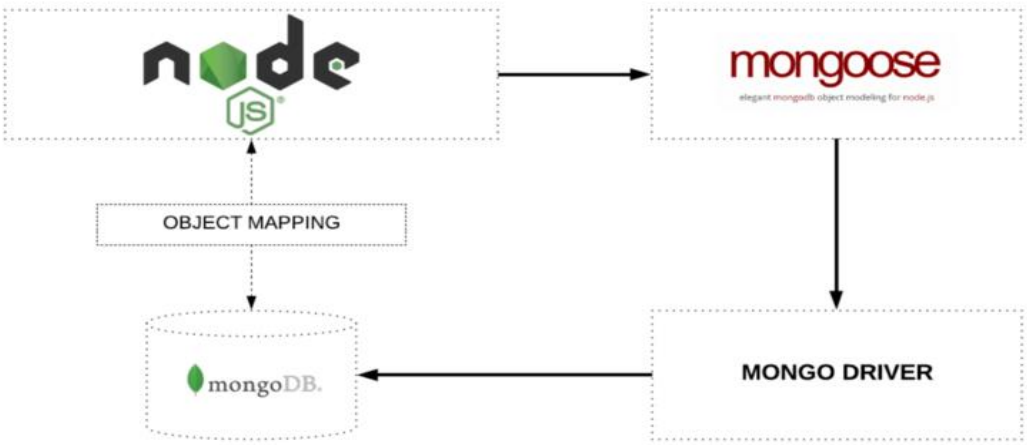

Figura 4. Asocierea obiectelor între Node.js și MongoDB administrate de către mongoose

1.3 Sisteme prezente pe piață

Printre sistemele prezente pe piață din același domeniu sau cu funcționalități asemănătoare și cu un număr impresionant de abonați, în continuă creștere, se află Flickr și Facebook.

Flickr este un serviciu de găzduire a materialelor de tip media imagine sau video, creat în 2004, cu peste 90 milioane de membri înregistrați și peste 25 milioane de fotografii noi încărcate zilnic. De-a lungul timpului această aplicație a avut mai multe versiuni în care dezvoltatorii au încercat noi moduri de interacționare între utilizatori și posibilități de transfer a materialelor, însă, ceea ce pare că a reușit de fapt să atragă publicul, au fost etichetele (tag-urile), marcarea fotografiilor ca favorite și gruparea acestora în funcție de interese și grupuri de oameni. Există două variante de conturi, gratuit și Pro. În prezent, opțiunea gratuită îți oferă posibilitatea de a stoca până la 1000 de fotografii și videoclipuri cu o durată maximă de 3 minute, dar aceste oferte sunt schimbate la o anumită perioadă de timp. De exemplu, până la data de 7 ianuarie 2019, utilizatorii cu un cont gratuit aveau până la 1TB spațiu de stocare, iar acum, dacă aceștia nu trec la varianta Pro, conținutul mai vechi va fi șters automat dacă există

mai mult de 1000 de fișiere. În plus, politica Flickr dă dreptul acestora de a șterge conturi fără a da vreun motiv sau avertisment proprietarului. Varianta Pro oferă, bineînțeles, stocare nelimitată, navigare fără publicitate, videoclipuri cu o durată de până la 10 minute și diferite oferte promoționale. Fișierele media pot fi organizate și afișate în diferite moduri, utilizatorii pot pune etichete cu titluri și descrieri și, de asemenea, pot acorda acces altor persoane pentru vizualizarea acestora. Utilizatorii pot să folosească sistemul de pe orice dispozitiv conectat la internet sau pot descărca aplicațiile mobile oficiale pentru iOS și Android, funcționalitățile de bază a acestuia bazându-se pe caracteristicile standard HTML și HTTP care permit o compatibilitate largă între platforme și browsere.

Flickr se descrie a fi ”aproape sigur cea mai bună aplicație online de gestionare și partajare a fotografiilor în lume, având două obiective principale:

1. Dorim să ajutăm oamenii să își facă fotografiile disponibile pentru persoanele care contează pentru ei.
2. Vrem să implementăm noi modalități de organizare a fotografiilor și videoclipurilor.” [10]

În ansamblu, se pot observa destul de multe asemănări între collectIn și Flickr, dar și deosebiri. În prezent, cea mai mare deosebire ar fi diferențele de conturi pe care Flickr le pune la dispoziție, în timp ce aplicația de față nu cere, momentan, niciun cost pentru stocarea fotografiilor. În plus, collectIn se axează strict doar pe păstrarea imaginilor și posibilitatea de împărtășire a acestora cu oamenii apropiați sau care au fost la evenimente comune și încearcă să ofere un mediu securizat, organizat și prietenos în care utilizatorii să își păstreze cu încredere toate fotografiile, care reprezintă pentru ei momente speciale și amintiri pe care vor să le aibe în viața lor și poate chiar să găsească altele noi în cadrul evenimentelor publice la care au luat parte.

În ceea ce privește rețeaua de socializare Facebook, nu s-ar putea face o comparație în totalitate deoarece această aplicație nu merge deloc în direcția unei rețele de socializare. Însă, Facebook oferă, printre multitudinile de funcționalități, și posibilitatea de creare, stocare, adăugare și partajare a fotografiilor, fie într-un mediu privat, doar cu prieteni apropiați sau public. Mai mult de atât, la fiecare eveniment creat se pot adăuga și fotografii din cadrul acestuia și, un factor destul de important pentru majoritatea utilizatorului este gratuitatea în totalitate.

Cu toate acestea, încrederea în acest proiect se bazează foarte mult pe exclusivitatea domeniului, în ideea că există un procent de persoane care vor efectiv o platformă doar pentru înmagazinarea pozelor și care să ofere într-o manieră intuitivă și simplificată toate modalitățile de care au nevoie ca experiențele lor să fie organizate exact așa cum își doresc.

2. Cerințe de sistem

În cadrul acestui capitol vor fi specificate în detaliu sursele de cerințe identificate și documentarea acestora, părțile implicate și dacă această aplicație are vreun efect asupra lor, metodele de elicitație și reprezentarea cazurilor de utilizare.

2.1 Surse de cerințe

În faza de definire a problemei, ținând cont de faptul că nu există o organizație anume care să beneficieze de această aplicație, fiind valabilă publicului larg, sursele de cerințe provin de la administratorul aplicației, care voi fi eu, impunând anumite constrângeri legate de interfață și calitate, plus câteva cerințe operaționale care să descrie funcționalitățile pe care utilizatorii le așteaptă de la sistem.

Astfel, cerințele privind construirea aplicației și constrângerile sub care aceasta va trebui să opereze, vor fi împărțite în cerințe funcționale, care alcătuiesc sistemul informatic și exemplifică rolurile fundamentale posibile, cât și unicitatea acesteia față de alte aplicații din același domeniu, și cerințe non-funcționale, care reflectă calitatea, comportamentul sistemului la execuție, organizarea software și structura.

În Tabelul 1 sunt trecute principalele cerințe funcționale ale aplicației dezvoltate.

Tabel 1. Cerințe funcționale

Cerințe funcționale	
1	Administrare utilizatori
1.1	- Înregistrare utilizator
1.2	- Autentificare utilizator
2	Personalizare/ Vizualizare profil
2.1	- Vizualizare evenimente personale
2.2	- Vizualizare fotografii din cadrul evenimentelor
2.3	- Adăugare/ștergere eveniment (și detalii eveniment)
2.4	- Adăugare/ștergere fotografii din cadrul unui eveniment
2.5	- Acordare acces și altor persoane la eveniment prin invitație
3	Vizualizare pagină de evenimente
3.1	- Căutare eveniment după filtre
3.2	- Adăugare eveniment pe profil
4	Administrare cont
4.1	- Modificare date cont (adresă e-mail, parolă, numele de utilizator)
4.2	- Ștergere cont

În prima parte de cerințe funcționale se regăsește posibilitatea unui utilizator sau viitor utilizator să își creeze un cont și să se poată autentifica. Pentru înregistrare este nevoie doar de o adresă de e-mail, o parolă, un nume de utilizator ales de acesta și locația (opțională).

După autentificare, utilizatorul intră direct pe pagina de profil a acestuia, unde poată să vizualizeze evenimentele personale (dacă acestea există) sau să creeze alte evenimente. În cadrul unui eveniment, este necesar un nume, o locație și data la care a avut loc. În continuare se vor putea adăuga oricâte fotografii utilizatorul dorește. În plus, după crearea evenimentului, se va putea acorda accesul și altor persoane pe baza unei invitații folosindu-se adresa de e-mail.

Pe pagina de evenimente publice, utilizatorul va putea căuta un eveniment anume folosindu-se de filtre, să vizualizeze fotografiile incluse și să adauge acel eveniment în profilul personal, dacă acesta a participat la el.

Ultima parte de cerințe funcționale cuprinde administrarea contului, prin care utilizatorul să poată să își modifice datele din cont, precum adresa de e-mail, parola sau numele de utilizator, sau să își șteargă definitiv contul. Implicit toate datele, evenimentele și fotografiile utilizatorului vor fi șterse.

Cerințele non-funcționale includ:

1. Securitatea reprezintă gradul de rezistență și protecție necesar împotriva oricărei vulnerabilități sau posibile amenințări. În acest caz, trebuie asigurate toate măsurile de precauție, astfel încât, utilizatorii să își păstreze în siguranță maximă datele personale, dar mai ales fotografiile. De asemenea, aceștia trebuie să fie asigurați că detaliile despre evenimentele publice și fotografiile din cadrul acestora sunt adevărate.
2. Performanța unei aplicații se referă la analiza parametrilor care pot ajuta la îmbunătățirea sistemului, precum: timpul de răspuns, modul de utilizare al resurselor calculatorului, rata de procesare etc. În ceea ce privește performanța, este extrem de important ca aceasta să ruleze la cele mai bune standarde de timp indiferent de numărul de utilizatori care folosesc aplicația în același timp.
3. Eficiența asigură performanța aplicației prin referirea la utilizarea resurselor calculatorului. De exemplu, pentru această aplicație, este necesar o conexiune la internet și navigarea pe un browser web. Pentru un utilizator care dorește doar să vadă anumite evenimente sau fotografii, resursele folosite sunt mai puține, comparativ cu un utilizator care încarcă câteva zeci/sute de fotografii într-un album.
4. Portabilitatea este de asemenea o cerință importantă deoarece aplicația trebuie să funcționeze indiferent de sistemul de operare, browser-ul folosit sau chiar dispozitivul de pe care aceasta a fost accesată.

5. Accesibilitatea constă în modul de utilizare a aplicației, dacă aceasta este ușor de înțeles și de folosit, în special pentru utilizatori. În plus, aceasta trebuie să fie ușor de modificat, ceea ce înseamnă că sistemul trebuie flexibil și de asemenea, să fie operațională în orice moment, ceea ce duce la fiabilitatea proiectului în obținerea unui produs de înaltă calitate.

Începând cu 25 mai 2018, o sursă importantă de cerințe este legislația privind protejarea datelor cu caracter personal, General Data Protection Regulation, abreviată ”GDPR”. Consimțământul pentru prelucrarea datelor are acum un regim mult mai restrictiv, solicitarea acordului trebuie să fie într-o formă ușor accesibilă și inteligibilă, cu un limbaj clar și simplu, bine diferențiată față de celelalte aspecte, dacă acestea există. De asemenea, consimțământul trebuie să se poată retrage într-un mod similar de accesibil și simplu, nefiind admisă condiționarea acestuia, de exemplu, condiționarea prestării unui serviciu pentru acordul de prelucrare a datelor în scopul unui marketing direct.

2.2 Elicitația cerințelor

Părțile implicate sunt publicul larg și ar putea fi anumite organizații care se ocupă de partea de fotografiere la anumite evenimente precum: banchete, nunți, petreceri private, festivaluri etc., și Roman Oana Luisa, partea implicată în analiza, proiectarea și implementarea aplicației web.

Ca prin beneficiu adus, este ușurarea activităților care constau în distribuirea fotografiilor tuturor persoanelor implicate, prin urcarea acestora într-un singur loc, adică pe platforma collectIn și invitarea celor implicați la evenimentul creat printr-o invitație pe e-mail. Astfel timpul de transmitere a informațiilor se reduce semnificativ, putându-se face totul automat. În plus, posibilitatea de restricționare a accesului la eveniment ,strict doar pentru persoanele care au participat, face totul să fie la o siguranță maximă. De asemenea, în cazul publicului larg, aplicația poate servi ca un spațiu de stocare a fotografiilor private, fără a mai fi nevoie de un spațiu personal de stocare, totul fiind foarte ușor de organizat și gestionat. Dezavantajul ar fi faptul că fiecare persoană care vrea să aibă acces la fotografiile de la un eveniment trebuie să își creeze un cont.

Un al doilea beneficiu pentru stakeholder ar fi extinderea globală. Nu contează de unde ești sau unde te afli ca să poți beneficia de funcționalitățile aplicației. Dezavantaj în acest caz ar fi poate serverele și necesitatea de extindere a acestora, fiind nevoie de mai mult spațiu și asigurarea evenimentelor din mai multe țări.

De altfel, printre dezavantaje se enumeră concurența care se află pe piață deja de mai mult timp. Cu toate că, aplicația în sine este foarte exclusivă cu domeniul și nu există alte aplicații care să se bazeze stric doar pe stocarea materialelor media de tip imagine, fie

personale, fie din cadrul unor evenimente, există multe aplicații foarte cunoscute precum: Facebook, Instagram, Flickr care au o parte și din aceste funcționalități. Consider însă, că, cu o promovare bine gândită și poate colaborarea cu anumite organizații prin intermediul cărora aplicația să ajungă cunoscută la mai multe persoane, concurența nu va mai fi o problemă.

Tehnicile de extragere și înțelegere a necesităților stakeholderilor s-au realizat printr-un mix de metode, esențiale pentru succesul proiectului. O metodă de colectare a datelor pentru a intra mai în amănunt, a fost tehnica directă de elicitație a cerințelor, interviul. Pentru a înțelege problemele reale și a posibilelor soluții din perspectiva utilizatorilor, clienților și a stakeholderilor. Interviul constă în culegerea datelor ce implică comunicarea verbală între cercetător și subiect. A fost ales acest tip de metodă deoarece este un mod direct de a intra în contact cu actorii, reușind să surprindă și tipurile de reacție care ajută la o preluare calitativă a informațiilor dorite.

Timpul estimat unui interviu este de 1 oră, unde se înregistrează răspunsurile după care se analizează în profunzime. În urma interviului cu administratorul unei organizații care se ocupă de fotografierea anumitor evenimente am înregistrat o serie de răspunsuri nuanțate, datorită timpului de reflexie asupra întrebărilor și datorită mimicii și gesturilor stakeholderului. Întrebările din interviul susținut, vizează trasarea unor concluzii în ceea ce privește cum ar trebui să arate aplicația din punct de vedere grafic, ce funcționalități să cuprindă și ce utilitate să aibă această aplicație.

Interviul a avut o structură împărțită în 2: prima parte ne va răspunde întrebărilor care țin de aspectul sistemului informatic, în timp ce a doua parte va aduna informații despre performanța și utilitățile așteptate. În cadrul primei părți, principalele întrebări au fost:

- Cum considerați că ar trebui să arate meniul, astfel încât utilizatorii să nu întâmpine probleme de orientare sau folosință?
- Care sunt metodele pe care le considerați cele mai eficiente în sortarea evenimentelor?
- Ați avea anumite restricții privind interfața? Dacă da, care sunt acelea?

Urmând întrebările care au alcătuit cea de-a doua secțiune a interviului, și anume:

- Ce considerați că ar trebui să ofere utilizatorilor această aplicație?
- Care sunt așteptările din punct de vedere al facilităților și performanței sistemului informatic?
- Ar avea un impact această aplicație asupra organizației sau vă va ajuta în vreun fel?

Deoarece opinia utilizatorilor cu privire la ce așteaptă ei de la o astfel de aplicație sau dacă ar fi interesați să folosească un asemenea sistem este extrem de importantă, am conceput un chestionar care a fost completat de persoane de diferite vârste, statut social sau profesional.

În redactarea chestionarului, am luat în considerare o listă de obiective clare: cercetarea în primul rând a numărului de oameni care folosesc mediul online pentru stocarea fotografiilor personale, cât de des accesează aceste platforme, ce preferințe au când aleg să își salveze imaginile online.

Chestionarul a cuprins următoarele întrebări și în unele cazuri, variante de răspuns:

1. În ce interval de vârstă vă situați?
   sub 18; 18-25; 25-45; peste 45.
2. Sunteți familiar cu tehnologia folosita in zilele noastre?
   Da/Nu.
3. Aveți încredere să vă păstrați fotografiile într-o aplicație web? De ce?
   Da/Nu.
4. Aveți vreo preferință în ceea ce privește o aplicație care v-ar ajuta la stocarea si gestionarea fotografiilor?
   Să fie ușor de accesat; bine securizată; timp bun de execuție; să nu fie costisitor.
   Altele:
5. Preferați să vă salvați fotografiile în calculatorul personal sau online?
   Calculator personal; online. Altele:
6. Ce v-ar interesa cel mai mult in ceea ce privește salvarea fotografiilor dvs.?
7. Care sunt aplicațiile pe care le folosiți cel mai des, mai ales când vine vorba de încărcarea fotografiilor dvs.?
   Facebook; Instagram; Flickr. Altele:
8. In urma chestionarului acesta, s-ar putea sa va intereseze in viitorul apropiat funcționalitățile oferite de collectIn? De ce?

În urma chestionarului, rezultatele arată în felul următor:

- 37,7% din personale care au răspuns la chestionar se încadrează în intervalul de vârstă 18-25; 26,4% au intre 25-45 ani, 22,8% sub 18 ani, iar 10% peste 45;
- În proporție de peste 80% din urma răspunsurilor persoanele implicate sunt familiarizate cu tehnologia curentă;
- Undeva între 70-80% au încredere să-și păstreze fotografiile într-o aplicație web;
- Preferințele principale sunt securizarea, un timp de execuție bun și navigarea ușoară, 73,1%, 50% și respectiv, 50%;
- 64,3% își păstrează fotografiile în calculatorul personal;
- Securitatea și restricționarea accesului au fost printre cele mai comune interese în salvarea fotografiilor personale;

- In prezent peste 60% folosesc Facebook și Instagram pentru încărcarea fotografiilor;
- Majoritatea s-au arătat interesați de funcționalitățile aplicației web în viitorul apropiat.

Pentru a deduce comportamentul de utilizare în ceea ce privește această aplicație web, voi folosi reprezentarea UML, prin modelul Use Case. Cazurile de utilizare reprezintă un set de acțiuni, servicii și funcții pe care sistemul trebuie să le îndeplinească. În acest context sistemul operat este o aplicație web. "Actorii" sunt persoanele care operează sub roluri definite în cadrul sistemului.

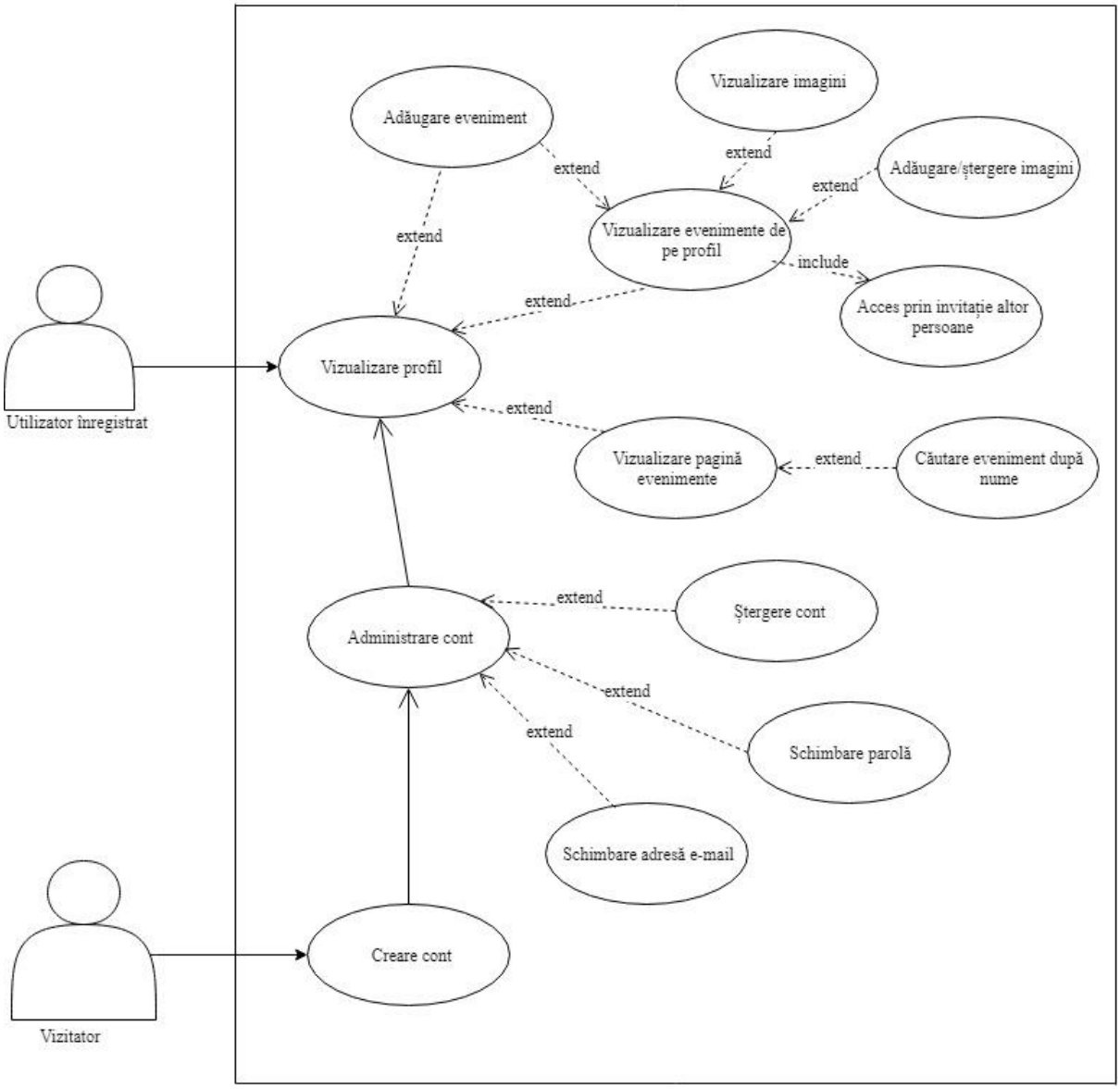

Figura 5. Diagrama cazurilor de utilizare

Figura de mai sus reprezintă ”Diagrama cazurilor de utilizare” având doi actori: vizitatorul, care încă nu are un cont, iar sigurul caz de utilizare pe care îl poate avea este acela de a-și crea unul devenind astfel utilizator și utilizatorul, care poate să își vizualizeze profilul,

să adauge evenimente, fotografii, membrii, să caute un anumit eveniment public sau să facă modificări de cont.

Pentru fiecare caz de utilizare în parte, scenariul de utilizare este prezentat în tabelele următoare.

Tabelul 2 redă cazul de utilizare în care un utilizator se autentifică, existând atât un proces alternativ, cât și unul standard.

Tabel 2. Caz de utilizare – utilizatorul se autentifică

Nume:	Utilizatorul se autentifică
Actori:	Utilizator înregistrat/vizitator
Descriere:	Acest caz de utilizare permite utilizatorului să intre într-un cont deja existent.
Pre-condiții:	Utilizatorul a intrat deja pe pagina de autentificare
Situații de eroare:	Pagina nu se încarcă, pagina eșuează la conectare datorită greșelilor la datele de intrare
Use-case-uri referite:	Administrare cont
Proces alternativ	1. Utilizatorul intră în pagina de autentificare, dar nu are cont 2. Intră pe pagina de înregistrare unde își creează un cont 3. Se autentifică
Proces standard:	1. Utilizatorul intra în pagina de autentificare 2. Introduce adresa de e-mail și parola 3. Conectarea are loc cu succes

Primul caz de utilizare este acela în care un utilizator se autentifică, ceea ce înseamnă că acesta are deja un cont înregistrat și se află pe prima pagină a aplicației web. Dacă adresa de e-mail și parola sunt introduse corect, atunci conectarea are loc cu succes, iar utilizatorului îi va apărea pagina sa de profil. În cazul în care datele nu sunt introduse corect sau nu există niciun utilizator cu aceste date va apărea un mesaj de avertizare.

Pentru utilizatorii care nu au încă un cont, vizitatorii din Diagrama cazurilor de utilizare, se va merge atunci pe procesul alternativ ceea ce înseamnă că de la pagina de start vor fi redirecționați către pagina de înregistrare, unde vor putea să își creeze un cont completând datele necesare, iar mai apoi vor putea să se autentifice.

În Tabelul 3 este descris cazul de utilizare atunci când un utilizator adaugă un eveniment nou, cu un singur proces standard.

Tabel 3. Caz de utilizare – utilizatorul adaugă un eveniment

Nume:	Adăugare eveniment privat sau public
Actori:	Utilizator înregistrat
Descriere:	Acest caz de utilizare se referă la posibilitatea adăugării unui eveniment nou
Pre-condiții:	Utilizatorul a intrat deja pe pagina profilului său
Situații de eroare:	Imposibilitatea de creare a evenimentului
Use-case-uri referite:	Vizualizarea evenimentelor care deja aparțin de utilizator
Proces standard:	1. Utilizatorul creează un album cu toate detaliile aferente evenimentului 2. Încarcă fotografiile care urmează să fie adăugate 3. După ce evenimentul a fost creat, utilizatorul poate permite accesul și altor persoane printr-o invitație prin e-mail

După ce un utilizator este deja autentificat și se află pe pagina sa de profil, acesta poate să creeze evenimente publice sau private, la care să adauge fotografiile dorite, precum și anumite detalii despre eveniment: denumire, locația unde a avut loc și data la care s-a întâmplat. În plus, după creare, utilizatorul poate să adauge noi membri care să aibă acces la eveniment, invitându-i prin adresa de e-mail cu care aceștia s-au înregistrat.

În acest caz de utilizare există un singur proces standard prin care un utilizator poate să creeze un eveniment nou.

Tabelul 4 reprezintă cazul de utilizare atunci când un utilizator vizualizează fotografiile aferente unui eveniment public sau privat.

Tabel 4. Caz de utilizare – utilizatorul vizualizează fotografii

Nume:	Vizualizare și modificare imagini
Actori:	Utilizator înregistrat
Descriere:	Acest caz de utilizare se referă la posibilitatea vizualizării imaginilor utilizatorului.
Pre-condiții:	Utilizatorul a intrat deja pe pagina de profil a acestuia.
Situații de eroare:	Pagina nu se încarcă, albumele nu își fac update decât după reîncărcare.
Use-case-uri referite:	Vizualizare albume și a imaginilor din acestea.
Proces standard:	1. Utilizatorul alege care dintre albume vrea să îl vizualizeze 2. Pe pagina albumului poate să vizualizeze fiecare poză în parte, să adauge sau să șteargă atât fotografii, cât și persoane care să aibă acces la album sau să șteargă tot evenimentul.

Cazul de utilizare descris în tabelul de mai sus reprezintă cazul în care un utilizator autentificat dorește să vizualizeze sau să modifice imaginile adăugate într-un eveniment. Există o singură pre-condiție a acestui caz de utilizare, aceea ca utilizatorul să fie deja autentificat și să se afle pe pagina de profil a acestuia. După ce utilizatorul accesează albumul pe care dorește să îl vizualizeze, pe pagină îi vor apărea toate fotografiile care aparțin de evenimentul respectiv. Acestea pot fi văzute la dimensiuni reduse sau pot fi vizualizate fiecare în parte. De asemenea, utilizatorul va putea adăuga fotografii noi, să șteargă oricare fotografie, să schimbe poza de copertă a evenimentului, să invite membrii noi sau să șteargă complet evenimentul.

Tabelul 5 descrie cazul de utilizare pentru căutarea unui eveniment public de către utilizator.

Tabel 5. Caz de utilizare – utilizatorul caută eveniment

Nume:	Căutare evenimente
Actori:	Utilizator înregistrat
Descriere:	Acest caz de utilizare se referă la posibilitatea căutării evenimentelor, fie în modul în care apar pe pagină, fie după nume
Pre-condiții:	Utilizatorul a intrat deja pe pagina site-ului
Situații de eroare:	Evenimentele nu pot fi găsite din cauza funcționalităților necorespunzătoare sau returnează un alt eveniment decât cel căutat
Use-case-uri referite:	Vizualizarea evenimentelor
Proces alternativ:	1. Utilizatorul intră în secțiunea de evenimente 2. Utilizatorul caută evenimente, după alegerea locației, direct din bara de căutare după nume.
Proces standard:	1. Utilizatorul intră în secțiunea de evenimente 2. Utilizatorul caută evenimente, după alegerea locației, în ordinea apariției lor pe pagină

Cazul în care un utilizator dorește să caute un anumit eveniment, vine de asemenea cu pre-condiția ca acesta să fie autentificat. Procesul alternativ constă în căutarea unu eveniment după numele acestuia sau măcar o secvență din nume. După ce utilizatorul a intrat în secțiunea de evenimente, acesta trebuie să aleagă locația unde a avut loc evenimentul. Dacă nu găsește în lista afișată locația pe care o caută se poate ca evenimentul să nu fi fost creat sau să fi adăugat altă locație. După alegerea locației, utilizatorul poate să introducă numele evenimentului în bara de căutare. Nu este necesar să fie numele complet, se vor returna și evenimente care conțin în nume secvența introdusă de utilizator. Procesul standard cuprinde pașii în ideea în care utilizatorul dorește să vadă toate evenimentele publice dintr-o anumită locație, iar tot trebuie să facă acesta este să aleagă locația și evenimentele vor apărea pe pagină.

În Tabelul 6 este descris cazul de utilizare atunci când un utilizator dorește să facă modificări de cont.

Tabel 6. Caz de utilizare – modificări de cont

Nume:	Schimbare adresă de e-mail/ date utilizator sau ștergere cont definitiv
Actori:	Utilizator înregistrat
Descriere:	Acest caz de utilizare permite utilizatorului să schimbe adresa de e-mail/ datele de utilizator sau să șteargă contul definitiv.
Pre-condiții:	Utilizatorul este autentificat și pe pagina de profil a acestuia.
Situații de eroare:	Pagina nu se încarcă, pagina eșuează la conectare datorită greșelilor datelor de intrare.
Use-case-uri referite:	Administrare cont
Proces standard:	1. Utilizatorul schimbă adresa de e-mail/datele de utilizator de la setările contului sau șterge contul definitiv

Dacă un utilizator dorește să își schimbe adresa de e-mail cu care s-a înregistrat, acesta poate să facă acest lucru cu condiția să fie autentificat. Din setările contului, utilizator poate să își modifice adresa de e-mail, precum și datele personale introduse: numele de utilizator și locația acestuia. Mai mult, din aceeași pagină, utilizatorul poate să își șteargă contul definitiv, ceea ce va însemna ca va pierde toate fotografiile încărcate și evenimentele create.

Tabelul 7 redă cazul de utilizare pentru schimbarea parolei utilizatorului.

Tabel 7. Caz de utilizare – utilizatorul schimbă parola

Nume:	Schimbare parolă
Actori:	Utilizator înregistrat
Descriere:	Acest caz de utilizare permite utilizatorului să modifice parola existentă.
Pre-condiții:	Utilizatorul este pe pagina de autentificare.
Situații de eroare:	Pagina nu se încarcă, pagina eșuează la conectare datorită greșelilor datelor de intrare.
Use-case-uri referite:	Administrare cont
Proces standard:	1. Utilizatorul se autentifică 2. Intră în setările contului 3. Introduce parolă nouă și face modificările
Proces Alternativ:	1. Utilizatorul selectează că a uitat parola din pagina de autentificare 2. Modifică parola uitată

În cazul în care un utilizator dorește să își schimbe parola actuală, cu condiția ca acesta să fie deja autentificat, ceea ce înseamnă că își cunoaște parola, acesta o poate modifica din setările contului, urmărind procesul standard. Bineînțeles, există posibilitatea ca un utilizator să nu își mai amintească parola actuală, ceea ce ar însemna că nu se mai poate autentifica. Pentru acest caz trebuie urmat procesul alternativ, în care utilizatorul, aflat pe pagina de autentificare, spune că și-a uitat parola accesând legătura aflată în pagină, reușind apoi să își aleagă o parolă nouă.

3. Proiectarea aplicației web

Acest capitol va prezenta în detaliu modul în care aplicația web a fost proiectată, atât din punct de vedere logic, care cuprinde arhitectura folosită, cât și tehnic, unde este reprezentată structura fizică a datelor și diagrama bazei de date.

3.1 Proiectarea logică

Într-un sistem informatic metodologiile apelează la operațiunea de modelare logică a datelor. Astfel, schimburile de date care au loc în cadrul sistemului pot fi reprezentate prin Diagrama fluxurilor de date (DFD). Acestea au ca obiectiv urmărirea modului de transfer al datelor între procesele de prelucrare a lor. Altfel spus, este o reprezentare grafică a „fluxului" de informații care circulă într-un sistem informatic.

Diagrama fluxului de date pentru această aplicație poate fi împărțită pe două nivele: nivelul 0, nivelul contextual al fluxului, reprezintă interacțiunea dintre sistem și agenții externi, utilizatorii; și nivelul 1 care arată cum este sistemul divizat și funcționalitățile proceselor, mai precis, indică sursele datelor de prelucrare, operațiunile prin care trec datele, destinația lor și legătura existentă între prelucrări și activitatea de stocare a datelor. Reprezentarea celor două nivele se face în figurile ce urmează.

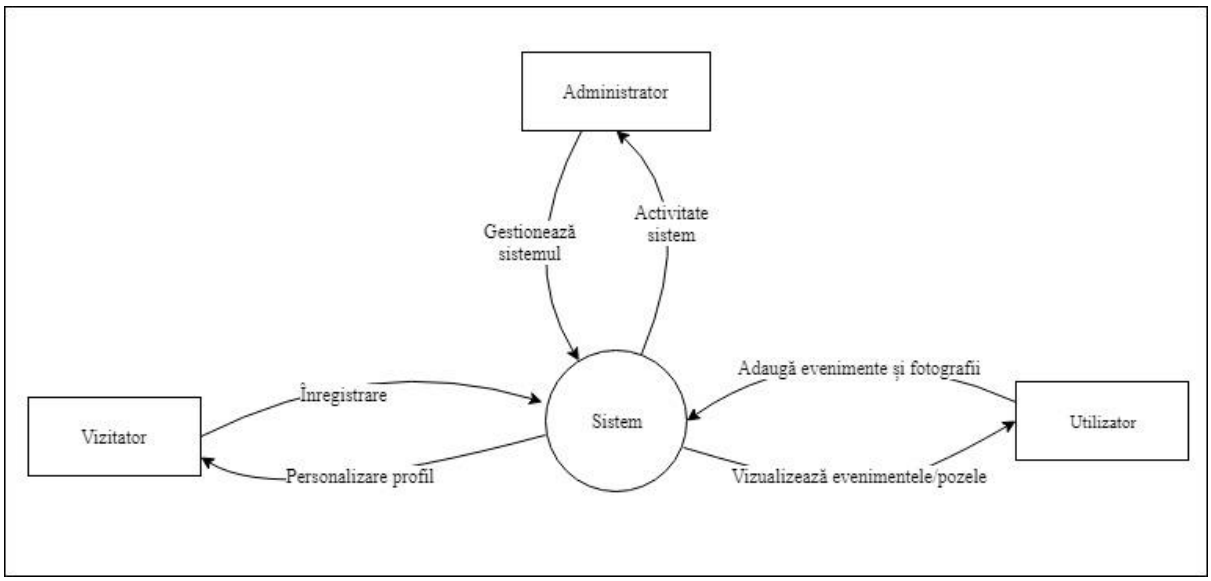

Figura 6. Diagrama flux de date pentru nivel 0

Astfel, figura de mai sus reprezintă diagrama fluxurilor de date pentru nivelul zero al aplicației, în care se poate observa interacțiunea dintre sistemul informatic și fiecare agent în parte: vizitatorul, administratorul și utilizatorul.

Vizitatorul poate doar să se înregistreze în sistem, ceea ce îi va oferi apoi o pagină de profil personalizată după datele introduse de acesta la înregistrare.

Administratorul este cel care gestionează sistemul, adaugă noi funcționalități, corectează anumite greșeli care pot apărea, întreține sistemul la o performanță și capacitate maximă, primind de la acesta activitatea desfășurată, alerte de erori sau avertizări, timpul de răspundere la cereri, etc.

Principala acțiune a utilizatorului cu sistemul este aceea de a adăuga noi evenimente și fotografii, pe care ulterior va dori să le vizualizeze.

Figura 7 prezintă Diagrama de flux de date pentru nivelul 1.

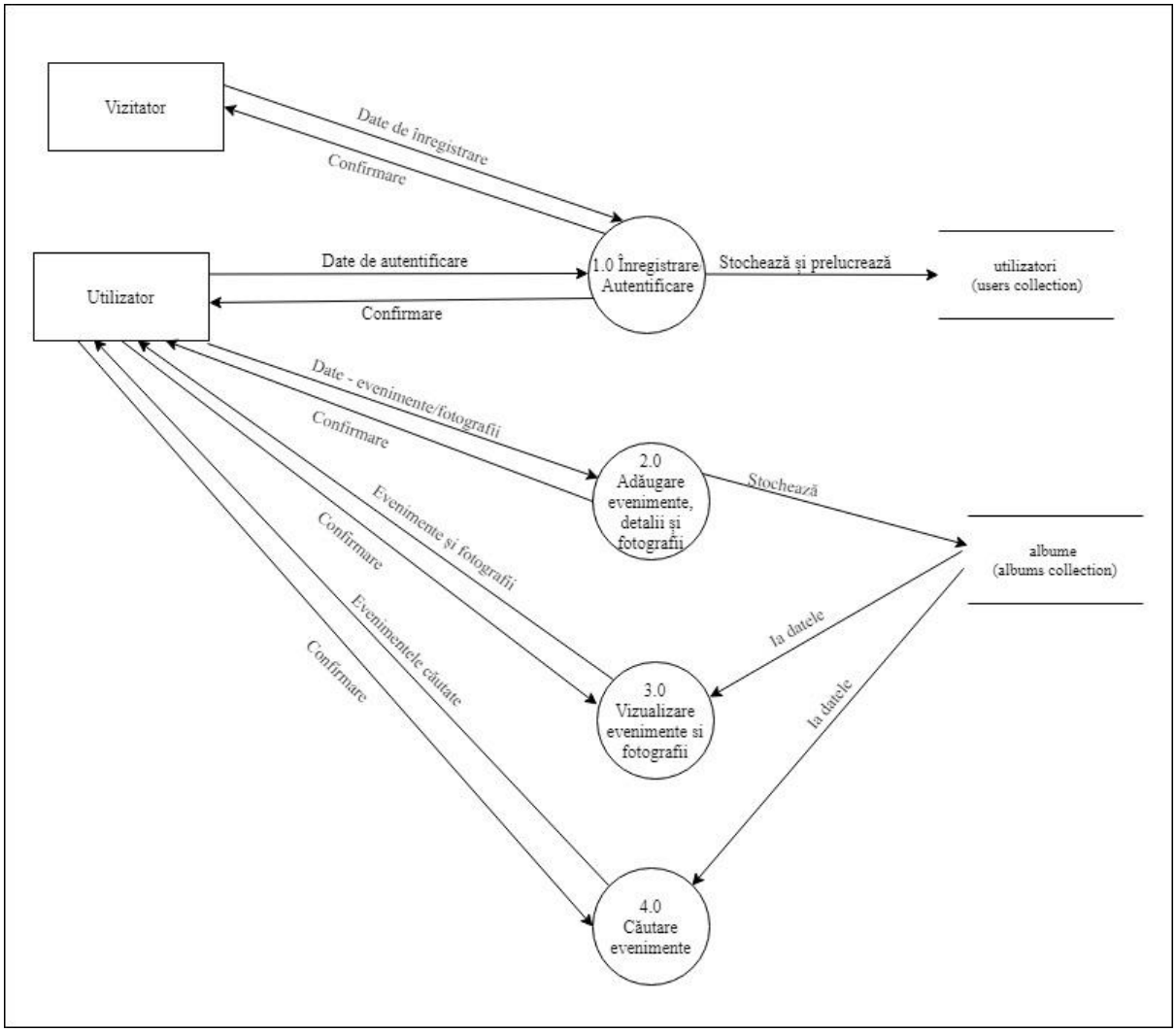

Figura 7. Diagrama flux de date pentru nivel 1

Diagrama flux de date pentru nivel 1, afișată mai sus, arată modul în care datele sunt prelucrate și traseul lor pe întreg procesul. Așadar, în cazul unui vizitator care se înregistrează, datele acestuia trec prin procesul de înregistrare, care le va prelucra și stoca în baza de date corespunzătoare, trimițând înapoi un mesaj de confirmare, astfel încât, vizitatorul să devină utilizator și să se poată autentifica cu datele introduse. Procesul datelor de autentificare este asemănător înregistrării, diferența constând în faptul că datele introduse nu mai stocate în baza de date, fiind deja salvate acolo.

Fluxul de date introdus de un utilizator, care conține detaliile unui anumit eveniment, precum și fotografiile acestuia, trece prin procesul de adăugare care le introduce în baza de date, în colecția albume. Dacă totul a funcționat în mod corect, utilizatorul va primi un mesaj de confirmare care constă în apariția evenimentului pe pagina sa cu toate detaliile aferente.

Al treilea proces constă în vizualizarea evenimentelor și fotografiilor adăugate de utilizator, proces în care fluxul de date este preluat din baza de date și trimis utilizatorului. Iar, cel de-al patrulea proces, se rezumă la căutarea unui eveniment, ceea ce presupune tot preluarea datelor corespunzătoare din baza de date, în funcție de căutările utilizatorului, și transmiterea rezultatului găsit.

Modul în care componentele sunt conectate astfel încât sistemul este capabil să implementeze cerințele impuse de aplicație reprezintă arhitectura generală a sistemului.

Structura acestei aplicații este reprezentată de o arhitectură client-server, cu trei componente principale: clientul, serverul și rețeaua care creează o conexiune între cele două. Clientul reprezintă interfața utilizatorului, componenta prin care acesta are o legătură directă cu aplicația, transmițând apeluri, cereri, către server pentru a primi datele de care are nevoie, în timp ce serverul, așteaptă mereu aceste solicitări de la orice client pentru a le putea procesa și trimite înapoi sub forma unui răspuns. Așadar, printre funcțiile principale ale serverului se află furnizarea de servicii utilizatorilor, administrarea accesului la baza de date, sortarea și persistența datelor.

Pentru o scalabilitate și flexibilitate îmbunătățită, precum și un trafic de rețea în general redus, arhitectura client-server va fi reprezentată pe trei niveluri, astfel codul server-ului și baza de date pot fi separate, dacă va fi nevoie la un moment dat, în plus, vor putea fi înlocuite fără ca alte componente să fie afectate.

Arhitectura client-server pe trei niveluri se poate observa în Figura 8.

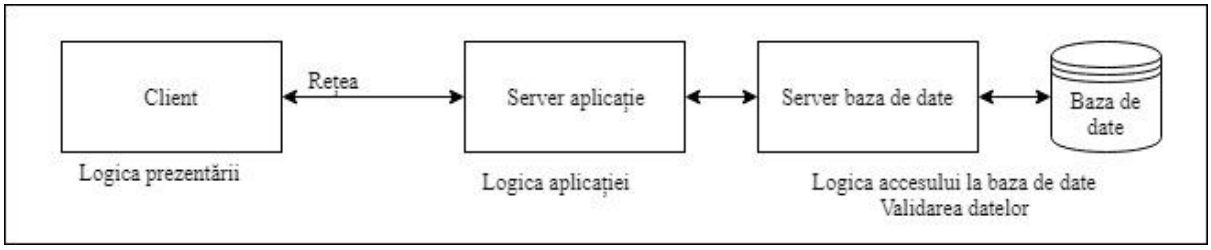

Figura 8. Arhitectura client-server pe 3 niveluri

Astfel, primul nivel, Clientul, este redat de interfața sistemului cu utilizatorul, administrarea design-ului aplicației web, formularele de autentificare, înregistrare etc., cu alte cuvinte, reprezintă logica prezentării. Nivelul doi, sau de mijloc, nu doar că asigură performanța, mentenanța, reutilizarea, flexibilitatea și scalabilitatea aplicației web, prin diferite funcționalități care stau la baza logicii aplicației, dar el este și cel care prelucrează datele pentru ca acestea să poată fi ulterior gestionate cu ajutorului serverului bazei de date. Deci, nivelul al treilea, este dedicat logicii accesului la baza de date și validarea din partea serverului.

Existența unui singur server de aplicații, pe care este stocat întregul procedeu de prelucrarea a datelor, face ca acest sistem informatic să fie centralizat, ceea ce oferă avantajul de a avea control asupra utilizării și dezvoltării aplicației, precum și asupra securității și integrității datelor.

În ceea ce privește comunicarea dintre client și server, aceasta trebuie să fie stabilă, datele să nu poată fi pierdute și mai ales, ordinea în care acestea ajung la client să fie exact aceeași în care serverul le-a trimis. După cum am explicat în capitolul 1, subcapitol 1.1 Descrierea unei aplicații web, Figura 1. Client-server comunicarea HTTP, componentele client și server vor comunica prin intermediul protocolului HTTP, utilizând tehnologia de comunicare între un client și server, numită, REST, Representational State Transfer.

Această tehnologie este un stil arhitectural care facilitează comunicarea între sistemele informatice de pe web prin furnizarea anumitor standarde. De asemenea, permite implementarea clientului și serverului în mod independent atâta timp cât cele două părți cunosc în ce format va trebui să trimită mesajul între ele. Astfel, utilizatori diferiți vor putea performa aceleași acțiuni și să primească aceleași răspunsuri.

Cererile clienților pentru a primi sau modifica resurse constau, în general, dintr-o metodă HTTP, care definește ce tip de operațiune trebuie performată, un antet și o cale către resursă. Sunt patru metode de bază HTTP folosite: GET, pentru a prelua o anumită resursă sau o colecție de resurse; POST, pentru a crea o resursă nouă; PUT, pentru a actualiza o anumită resursă; DELETE, pentru a elimina o anumită resursă. În acest proiect, fiind folosită doar metoda POST. Prin antet, clientul trimite tipul conținutului care este posibil să fie primit de la server.

Răspunsurile serverului la aceste cereri conțin un cod de stare care alertează clientul despre succesul operațiunii cerute. De exemplu, codul 200 transmite că totul a funcționat cum trebuie, în timp ce codul 404 spune că resursa căutată nu a fost găsită.

În cazul aplicației server, aceasta folosește modelul MVC, Model-View-Controller, care împarte aplicația și dezvoltarea acesteia în 3 părți, având fiecare o responsabilitate proprie, astfel ajutând la mentenanța și reutilizarea eficientă a codului.

Modelul, Model, menține și reprezintă datele aplicației, structura, formatul și constrângerile acestora. Partea de View, vizualizare, utilizează modelul și prezintă datele utilizatorului în forma dorită de acesta după ce, controler-ul, Controller, controlează cererile primite și generează apoi un răspuns adecvat care este alimentat utilizatorului.

Figura 9 reprezintă Arhitectura Model-View-Controller a aplicației server.

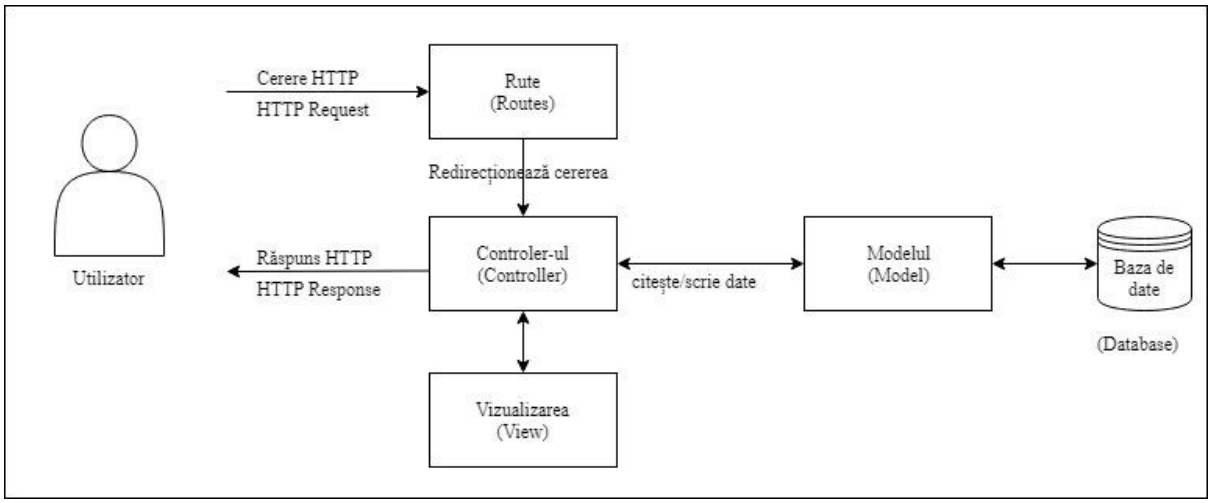

Figura 9. Arhitectura Model-View-Controller

După cum indică figura de mai sus, în arhitectura Model-View-Controller, cererea HTTP a utilizatorului este redirecționată cu ajutorul rutelor către controler-ul corespunzător. Acesta va gestiona cererea conform modelului de date aferent și informațiile din baza de date, urmând apoi să randeze vizualizarea adecvată sub forma unui răspuns HTTP.

3.2 Proiectarea tehnică

Baza de date din acest sistem este o bază de date NoSQL, datele sunt văzute sub formă de documente și stocate în colecții. Pentru a gestiona relațiile dintre date se folosește biblioteca Mongoose, care de asemenea validează și schemele definite.

Așadar, avem o bază de date numită collectin care conține două colecții: users (utilizatori) și albums (albume). În cazul primei colecții, aceasta va stoca datele utilizatorilor precum: e-mailul, parola, numele, locația și un id generat automat de către MongoDB.

A doua colecție memorează datele albumelor create de utilizatori. Pentru fiecare album creat, la fel ca la utilizatori, MongoDB va genera un id unic, va fi salvată data la care albumul a fost creat, dacă acesta este public sau nu, numele evenimentului, data la care a avut loc evenimentul și locația acestuia, de asemenea, pozele adăugate, membrii care au acces la eveniment și id-ul utilizatorului care a creat albumul.

Figura 10 redă într-un mod descriptiv diagrama bazei de date cu cele două colecții.

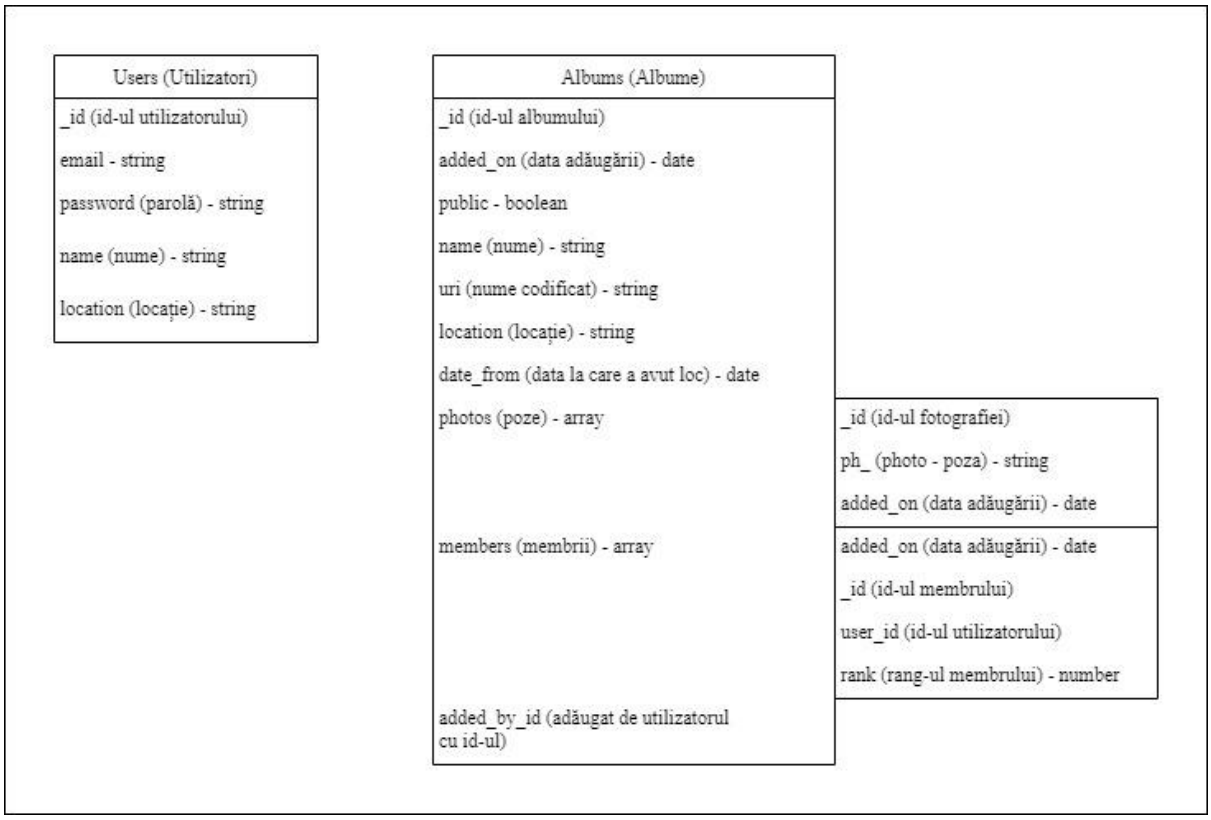

Figura 10. Diagrama bazei de date

Figura de mai sus descrie colecțiile bazei de date și documentele încorporate, precum și tipul de date ale acestora, într-un manieră clară, sub formă tabelară, cu toate că în această bază de date nu există tabele, ci colecții. Astfel, se pot observa cele două colecții: Users (Utilizatori) și Albums (Albume), cu toate documentele aparținătoare și tipul de date.

În cazul fotografiilor, photos, și membrilor, members, avem documente îmbricate, fiecare având propria schemă definită, iar dacă baza de date se vizualizează cu ajutorul interfeței grafice MongoDB Compass Community, acestea apar sub forma unor mulțimi de obiecte. Fiecare obiect, în mulțimea fotografiilor, conține datele unei imagini: id-ul, numele acesteia și data la care a fost adăugată. Vectorul membrilor cuprinde obiecte care stochează date precum: data adăugării membrului, id-ul de membru generat automat de MongoDB, id-ul utilizatorului care a fost de fapt inclus și rangul acestuia.

În plus, dacă un utilizator setează o anumită fotografie ca și copertă a unui album, se va crea un nou obiect, în colecția albums, care va conține informațiile pozei alese: id-ul, fotografia și data adăugării, exact aceeași schemă ca și în cazul obiectelor din vectorul de imagini.

Din câte se poate observa din diagrama anterioară, structura bazei de date este simplificată, fără existența anumitor relații, ci legături doar pe bază de id-uri, ceea ce oferă un timp de răspuns îmbunătățit, rapiditate la accesare și posibilitatea păstrării datelor de mari dimensiuni fără nici o problemă.

Proiectarea interfeței cu utilizatorul trebuie să respecte câteva aspecte generale precum: interacțiunea cu elementele aplicației web să se realizeze într-un mod dinamic, ușor de folosit și intuitiv; să aibă un design orientat spre utilizator, curat, cu un contrast potrivit și forme distincte; organizarea informației astfel încât utilizatorul să nu întâmpine probleme de orientare.

Așadar, voi folosi un design simplu, cu culori deschise, pe nuanțe de albastru deoarece conform unor articole de pe internet, albastrul reprezintă încredere, securitate, stabilitate, pace și calm. Exact sentimentele pe care collectIn dorește să le ofere utilizatorilor săi. Toate aceste aspecte vor fi stilizate cu ajutorul limbajului CSS, folosit cu extensia LESS, și a librăriilor Boostrap, DropzoneJS și FancyBox.

Bootstrap-ul, în primul rând, optimizează codul pentru dispozitivele mobile, iar apoi scalează componentele după dimensiunile necesare. Altfel spus, modelează modul în care componentele vor fi afișate pentru utilizator și dimensiunile acestora, în funcție de mărimea ecranului dispozitivului folosit. Așadar, interfața utilizatorului se va proiecta folosind conceptele lui Bootstrap de grid. Sistemul de grid folosește containere, rânduri și coloane pentru a realiza alinierea și aranjarea conținutului în pagină. Container-ul oferă un mijloc de a centra și regla orizontal conținutul care trebuie să fie plasat neapărat între coloane. Aceste coloane sunt ”învelite” de rânduri, în acest fel, tot conținutul este aliniat vizual în partea stângă. Deoarece Bootstrap împarte un rând în 12 coloane, există mai multe clase de coloane care se pot folosi pentru a indica numărul de coloane dorit să fie utilizat, în funcție de mărimea dispozitivului. De asemenea, Bootstrap ajută și la stilizarea alertelor din pagină, a formularelor, meniului de navigație, butoanelor și a altor elemente de grafică.

Pentru partea de proiectare grafică a spațiului destinat încărcării fotografiilor de către utilizator se va utiliza stilul implicit a librăriei DropzoneJS. În plus, aceasta oferă un aspect personalizat și modului în care imaginile vor fi afișate până la vizualizarea acestora în album și confirmarea individuală pentru fiecare poză că a fost încărcată cu succes.

Utilizatorul va dori să își poată vizualiza fotografiile individual și într-un mod accesibil, din acest motiv am implementat librăria FancyBox, care înfățișează modalitatea de prezentare a acestora. Fiecare poză va fi încadrată într-un cadran simplu și alb, poziționată la mijlocul paginii și afișată la dimensiunea optimă, astfel încât utilizatorul să nu fie distras de alte elemente din pagină, ci concentrarea să fie doar asupra fotografiei pe care vrea să o privească.

Toate celelalte elemente din pagină vor fi stilizate individual, cu ajutorul extensiei LESS care nu doar că permite definirea unor variabile, de care va fi nevoie în mai multe locuri, ci acceptă și amestecarea mai multe proprietăți pentru un singur element. În plus, oferă posibilitatea de a crea grupuri de elemente care aparțin aceleiași componente pentru a fi stilizate într-un mod organizat și explicit. Astfel, se vor folosi: trei culori de bază, definite în variabile pentru a simplifica atribuirea lor elementelor corespunzătoare și trei fonturi diferite, unul general, unul pentru titluri și unul pentru butoane. Fonturile au fost alese și importate de pe Goole Fonts, astfel încât utilizatorii să primească informația de care au nevoie cât mai clar și natural, în funcție de importanța acesteia. Mărimea, grosimea și culoarea textului se vor preciza individual pentru anumite elemente. Bineînțeles, în pagină vor exista și alte culori pe lângă cele de bază.

De asemenea, vor fi setate și alte proprietăți precum: imaginea de fundal a primei pagini care va apărea la accesarea aplicației, margini, alinieri, umpluturi, borduri, umbre, înălțimea liniei, culori de fundal, cât și poziții ale elementelor față de alte componente.

La accesarea aplicației, prima pagină care va apărea unui utilizator este pagina de autentificare care conține un formular cu câmpurile adresa de e-mail și parola. Dacă acesta nu are încă un cont, își va putea crea unul accesând legătura ”Creați-vă unul.” către pagina de înregistrare, unde va trebui să completeze adresa de e-mail, o parolă, un nume de cont și locația acestuia, opțională. După autentificare, utilizator își va putea vedea pagina de profil cu albumele sale, dacă există.

Pentru o navigare ușoară, fiecare pagină, cu excepția celor de autentificare și înregistrare, vor avea în partea de sus un meniu care cuprinde legăturile către panoul principal, care este de fapt pagina cu albumele utilizatorului, pagina de explorează, de creează și un submeniu cu setările contului și ieșirea din cont.

Albumele de pe pagina utilizatorului vor fi afișate sub forma unei căsuțe care conține fotografia de copertă, dacă aceasta a fost aleasă de către utilizator sau o imagine implicită, numele evenimentului, data la care acesta a avut loc și locația. La accesarea unui album, utilizatorul va putea vizualiza fotografiile, atât la scară largă, cât și individual. De asemenea,

va putea adăuga sau șterge imagini, să seteze coperta evenimentului, să șteargă tot albumul, dar și să gestioneze sau să vadă membrii care au acces la el.

Dacă acesta dorește să creeze un album nou, la secțiunea ”Creează” va putea face acest lucru după ce completează denumirea, locația și data evenimentului. În plus, dacă acesta dorește să îl facă public pentru orice utilizator, va putea selecta căsuța de ”Album public”. După creare, se pot adăuga fotografiile prin funcționalitatea de ”drag & drop” care va fi implementată cu ajutorul librăriei DropzoneJS.

Pagina ”Explorează” este locul unde utilizatorul poate să caute un eveniment public în funcție de locul unde acesta a avut loc. După alegerea unei locații există două posibilități de căutare: după nume, dacă știe denumirea albumului sau căutând efectiv în pagină. Ținând cont că este vorba de evenimente publice, este de așteptat ca persoana care l-a creat să ofere o locație reală și un nume sugestiv, astfel încât, utilizatorii care doresc să caute evenimentul să nu întâmpine probleme pentru a-l găsi. Oricum, pentru a nu fi foarte complicat să găsești un album, dacă nu știi numele exact, sistemul de filtrare returnează și albume care conțin în denumire secvența de caractere pe care utilizatorul a introdus-o în bara de căutare.

În secțiunea de ”Setări”, utilizatorul poate să își modifice numele, locația și adresa de e-mail, să schimbe parola actuală sau să șteargă definitiv contul, eliminându-se automat toate evenimentele create de acesta și pozele adăugate.

Partea de jos paginilor vor conține o bară de subsol cu anul în care aplicația a fost dezvoltată și toate drepturile rezervate.

4. Implementarea sistemului

În cadrul acestui capitol va fi prezentat în detaliu modul în care sistemul informatic a fost implementat, programele necesare pentru dezvoltarea aplicației și descrierea algoritmilor utilizați pentru realizarea funcționalităților specifice.

Aplicația a fost dezvoltată pe un calculator care rulează sistemul de operare Windows 10. În afara de acestea, implementarea necesită setarea unui mediu de dezvoltare alcătuit din diferite programe software, precum:

- Node.js, oferă mediul de executare pentru rularea codului JavaScript și poate fi descărcat de pe site-ul oficial (https://nodejs.org/en/download/). În timpul dezvoltării acestui proiect s-a folosit versiunea 8.11.4.
- MongoDB deoarece aplicația necesită o bază de date pentru a stocarea și interogarea datelor. Poate fi descărcat de pe site-ul oficial (https://www.mongodb.com/download-center) și MongoDB Compass (https://www.mongodb.com/download-center/compass?jmp=hero) interfața grafică pentru MongoDB. Versiunile 4.0.8 și, respectiv, 1.17.0 au fost folosite în timp ce s-a desfășurat proiectarea sistemului.
- Visual Studio Code, editor de cod sursă, care dispune de suport încorporat pentru JavaScript, Node.js și alte extensii disponibile.

Componentele structurii aplicației sunt separate corespunzător în funcție de rolul lor. Figura 11 arată modul în care acestea sunt organizate în fișiere.

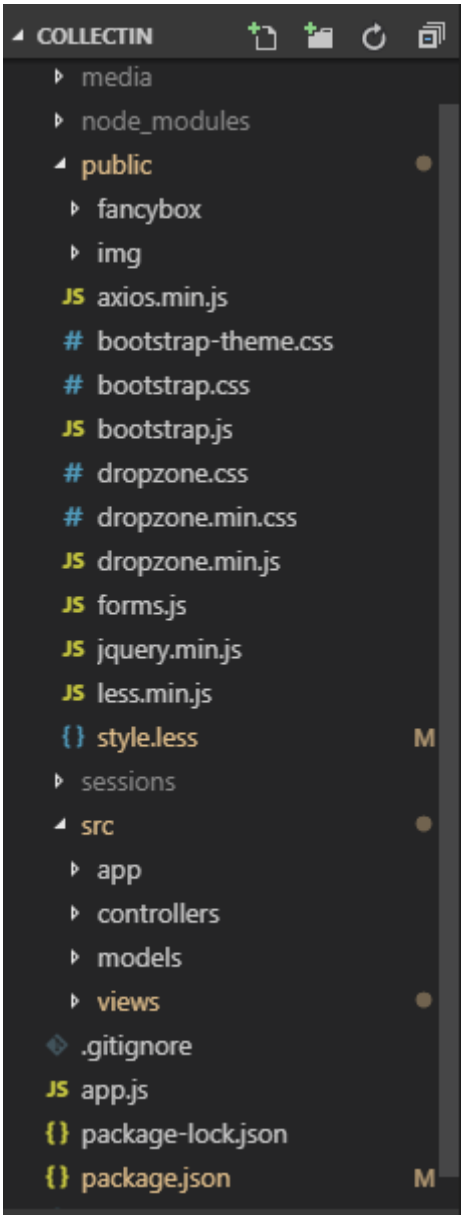

Figura 11. Organizarea în fișiere a sistemului

În package.json pot fi văzute toate informațiile importante despre aplicație, precum numele acesteia, versiunea, licența, autorul și pachetele instalate cu versiunile aferente. Pentru realizarea aplicației collectIn s-au instalat următoarele pachete: ”body-parser”, folosit de framework-ul Express; ”cookie-parser”, ajută la transmiterea cookie-ului clientului din browser la server; moment, pentru formatarea și manipularea datelor; ”mongoose”, folosit de baza de date MongoDB; ”multer”, pentru încărcarea fotografiilor; ”nunjucks”, limbajul de templating; ”session-file-store”, folosit în mod special pentru a stoca sesiunile pe disc, în loc de memorie, în cazul în care aplicația node.js își dă un restart, utilizatorii să nu fie delogați; ”uuid”, pentru generarea codurilor unice, mai precis id-urilor.Toate aceste pachete au fost instalate folosindu-se managerul de pachete, ”npm”.

Fișierul principal al aplicației este ”app.js” deoarece acesta conține toate inițializările, configurările, rutarea și pornirea serverului pe portul ales. Astfel, pe primele linii din cadrul fișierului sunt declarate anumite constante care să ajute la configurarea și inițializarea pachetelor.

Express va fi declarat în constanta ”app” și va fi folosit pentru fiecare inițializare, configurare sau rutare. Funcția principală a unui obiect de tip express este ”use()”, iar cu ajutorul acesteia se va seta ca aplicația să folosească: JSON; modulul express pentru ca interpretarea apelurilor făcute de la client prin Axios să fie de tipul ”urlencoded”; fișierele media și public în mod static; pachetul ”cookie-parser”; și inițializarea sesiunilor utilizatorilor autentificați. De altfel, se vor declara rutele în funcție de cererea primită, să se cunoască fișierul unde va trebui să se prelucreze datele. În plus, se va configura pachetul ”nunjucks” pentru randarea fișierelor html, care va conține și o funcție de filtrare pentru prelucrarea datelor în momentul afișării, în formatul definit cu ajutorul pachetelui ”moment”.

La finalul documentului se găsește declarația de pornire a serverului pe portul 3000. Am ales acest port deoarece nu există aplicații standard care să îl folosească în sistemul de operare Windows.

În dosarul ”src” se află partea de back-end a sistemului și este împărțită în: ”app”, ”controllers”, ”models” și ”views”.

Dosarul ”app” conține funcțiile de bază ale aplicației, în ”api_response.js” fiind codul pentru funcția care pregătește răspunsul pe care serverul îl va trimite către client. ”Auth_js” face funcțiile de bază a verificărilor în cazul în care un user este autentificat sau nu, iar ”utils.js” cuprinde o funcție de ajutor: ”slugify”, care face dintr-un șir de caractere un URI. De exemplu, pentru transformarea în URI a numelor albumelor se folosește funcția ”slugify”.

”controllers” conține logica aplicației, fișierul ”index.js” având terminalele publice pentru înregistrare, autentificare și ieșire, folosite la randarea template-urilor, modelelor, cât și cererilor corespunzătoare.

În ”dashboard.js” se verifică dacă există o sesiune a utilizatorului, altfel acesta va fi deconectat automat, iar în funcție de cererile acestuia, se vor face legăturile pentru randarea paginilor corespunzătoare, gestionate în funcție de modelul definit în ”models.js”, dacă este necesar.

Pentru a da un exemplu, Figura 12 redă codul folosit pentru implementarea uneia din funcționalități, și anume, afișarea unui album.

```
router.get('/album/:album_uri', auth.sessionChecker, (req, res) => {
    models.Album.findOne({uri: req.params.album_uri}, (e, album) => {
        // Get current user's membership in this album
        let member = null;
        for(let i=0; i<album.members.length; i++) {
            if (album.members[i].user_id.toString() === req.session.user_id) member = album.members[i];
        }

        // Get all users which are members in this album
        let membersIds = [];
        album.members.map(m => {
            membersIds.push(m.user_id.toString())
        });
        models.User.find({'_id': {$in: membersIds}}, (e, members) => {
            res.render('album.html', {user: req.session.user, album: album, members: members, member: member});
        });
    });
});
```

Figura 12. Codul care implementează funcționalitatea de afișare a unui album

Dacă utilizatorul dorește să creeze un album nou, accesând secțiunea ”Creează”, îi va fi randată pagina ”create.html” din dosarul ”views”. În schimb, după cum se poate observa în figura de mai sus, dacă dorește să vizualizeze un anumit album, va fi randată pagina ”album.html”, dar înainte se va prelucra pe baza schemei definite în ”models.js”, datele conform modelului pentru albume, astfel încât, să se știe dacă utilizatorul este membru al albumului, ce rang are și ceilalți membrii. În funcție de aceste date, vizualizarea poate să ofere funcționalități diferite. Mai exact, dacă membrul este cel care a creat albumul, atunci acesta va putea adăuga sau ștergere fotografii și membrii, să șteargă complet albumul și să seteze poza de copertă. În cazul în care acesta nu a creat albumul, ci doar a fost invitat, nu va putea adăuga sau șterge noi membrii și nici să șteargă tot albumul.

”api.js” reprezintă API-ul aplicației collectIn și conține toate rutele care întotdeauna sunt folosite de către partea de front-end prin Axios și de către back-end prin metoda HTTP POST.

Aceste rute oferă funcționalitățile de CRUD: creare, inserare, modificare, ștergere. De asemenea, aici sunt definite operațiunile pentru procesarea datelor solicitate și trimise, salvarea și interogarea acestora în baza de date folosind modelele.

Pachetul multer, un pachet Node.js care parsează fișierele și ajută la încărcarea acestora, în cazul de față, a fotografiilor adăugate de utilizator este configurat cu destinația acestora în dosarul ”media” și organizarea lor în alte dosare în funcție de luna și anul adăugării, precum și setarea numelor imaginilor cu ajutorul generatorului de coduri unice, ”uuid”.

Prin urmare, ”api.js” conține toate funcționalitățile de adăugare a unui eveniment, a unei poze, a unui membru; pentru ștergerea unui album, a unei poze, a unui membru, a unui

cont; editarea datelor unui utilizator; alăturarea într-un album public; părăsirea unuia și setarea unei fotografii de copertă. Pe scurt, tot ce poate să facă un utilizator în acest sistem informatic. Figura 13 conține codul care implementează funcționalitatea de adăugare a unei fotografii.

```
router.post('/albums/add-photo/:aid', upload_photo.single('file'), function(req, res) {
    console.log('UPLOAD', req.file.filename);

    let aid = req.params.aid;
    models.Album.findById(aid, function(e, album) {
        album.photos.push({photo: req.file.filename, added_on: moment()});
        album.save().then(x => {
            response.ok(res, req.file.filename);
        });
    });
});
```

Figura 13. Codul care implementează adăugarea unei fotografii

Mai sus este reprezentată ruta creată pentru adăugarea unei fotografii care folosește pachetul ”multer” prin parametrul ”upload_photo.single()”. Calea rutei este ”/albums/add-photo/:aid”, unde ”:aid” reprezintă id-ul albumului în care se face încărcarea, iar în funcție de modelul Album definit în ”models.js” și cu ajutorul funcțiilor ”push()” și ”save()” de la MongoDB se salvează datele aferente imaginii încărcate în baza de date și se trimite un răspuns.

În ”models.js” se face conectarea cu baza de date ”collectin” din MongoDB și sunt definite schemele bazei de date cu ajutorul lui mongoose. Acestea fiind modelele folosite în ”controllers”.

Așadar, există un model pentru colecția users, utilizatori, care declară tipul documentelor e-mail, parolă, nume și locație ca fiind string, adică șir de caractere. Pentru colecția albums, albume, vor fi necesare declararea a încă două scheme, una pentru fotografii, care să conțină numele fotografiei de tip string și data la care a fost adăugată; și una pentru membrii, cu id-ul utilizatorului, rangul de tip număr, number, și data la care a fost adăugat membrul. Deci, modelul unui album constă din numele acestuia, uri-ul și locația, acestea fiind de tip string, data la care a avut loc evenimentul, data la care a fost adăugat, dacă este public sau nu, fotografiile, cu schema implicită, coperta albumului, care va lua schema de la fotografia aleasă, id-ul utilizatorului care a creat albumul și membrii.

”views” conține toate fișierele HTML ale aplicației, unde sunt prezentate componentele care vor fi afișate utilizatorului prin folosirea limbajului de templating ”nunjucks”.

Ca să existe o organizare adecvată, cât și un mod de lucru simplificat, care să poată permite modificări în oricare pagină fără a fi afectată alta, se vor crea mai multe șabloane pentru diferite componente sau părți dintr-o pagină anume.

În primul rând, trebuie să existe un tipar pentru vizitatorii aplicației, acesta fiind descris în ”_layout.public.html”. Pagina ”index.html” extinde șablonul public conținând formularul de autentificare și este prima pagină randată la pornirea aplicației. ”register.html” extinde de asemenea acest model public cuprinzând formularul de înregistrare.

Următorul șablon, ”_layout.html” este folosit doar în paginile în care un utilizator este autentificat și include atât meniul de navigație, cât și bara de subsol. ”dashboard.html” reprezintă pagina de profil sau panoul de control, personalizată pentru fiecare utilizator cu numele și albumele corespunzătoare, dacă există, prin includerea șablonului ”_albums.html”. Acest șablon ajută la afișarea albumelor cu toate detaliile acestora: poza de copertă, nume, dată, locație. De asemenea, un album are o pagină definită ”album.html” în care sunt prezentate, pe lângă detaliile specifice, și fotografiile adăugate, membrii și funcționalitățile posibile. ”album.add.photos.html” reprezintă locul în care utilizatorii își pot încărca fotografiile și conține librăria DropzoneJS care oferă funcționalitatea și modul de afișare a acestuia. Formularul de creare a unui eveniment nou se găsește pe pagina ”create.html” randată la accesarea secțiunii ”Creează” din meniul de navigație. ”explore.html” implementează pagina secțiunii ”Explorează” care afișează toate locațiile evenimentelor publice existente. După ce utilizatorul alege o locație va fi redirecționat către pagina cu albumele găsite, folosind șablonul ”_albums.html” pentru afișarea acestora, ”explore.location.html” care cuprinde și bara de căutare după numele unui eveniment. Profilul unui alt utilizator decât cel autentificat va fi randat cu ajutorul fișierului ”user.html”, care include, de asemenea, șablonul ”_albums.html” pentru prezentarea evenimentelor acestuia. În pagina de setări, ”settings.html”, se regăsesc formularele de modificare a numelui, locației sau adresei de e-mail și parolei, precum și butonul de ștergere definitivă a contului.

În toate aceste fișiere HTML folosim limbajul ”nunjucks” pentru: extinderea și includerea șabloanelor, moștenirea din pagini, folosirea de variabile, afișarea conținutului selectiv folosind sintaxa ”if” și traversarea mulțimilor de obiecte cu ”for”.

Figura 14 redă pagina ”dashboard.html”, aflată în fișierul ”views”, reprezentând panoul de control al unui utilizator.

```
{% extends '_layout.html' %}
{% block title %}collectIn - Panou principal{% endblock %}

{% block content %}
<h1>{{ user.name }}</h1>
<div class="subtitle">Panou principal</div>

<div class="row">
    {% if albums|length == 0 %}
        <div class="text-center empty-content">
            Nu aveți încă nici un album.<br>
            <a href="/dashboard/create">Creează unul acum!</a>
        </div>
    {% endif %}

    {% include '__albums.html' %}
</div>
{% endblock %}
```

Figura 14. Pagina ”dashboard.html”

Figura de mai sus prezintă modul de lucru folosind limbajul nujucks, pagina ”dashboard.html” extinde șablonul ”_layout.html”. În plus, se poate observa cum numele utilizatorului este afișat folosind variabile, iar dacă nu există încă albumele create să se afișeze un mesaj, altfel să se afișeze albumele prin includerea șablonului ”_albums.html”.

Pentru realizarea funcționalităților din anumite pagini, între etichetele ”script”, sunt definite evenimentele de control, event handler, care la apăsarea unui buton vor merge la acțiunea corespunzătoare din ”forms.js”. Această acțiune va face un apel la funcția din ”api.js” care prelucrează datele și trimite înapoi un răspuns către partea de front-end.

Fișierele și librăriile folosite de aplicația client se află în dosarul ”public” structurate în următorul mod: un dosar pentru librăria FancyBox; un dosar ”img” care conține imaginea de fundal a paginii de autentificare și coperta implicită a evenimentelor; fișierele librăriilor Bootstrap, DropzoneJS, LESS, axios și jQuery; ”style.less” unde sunt definite stilurile proprii aplicate elementelor; și ”forms.js”.

Fiecare formular din aplicație primește date direct de la utilizator, după completarea acestora, prin apăsarea unui buton. Pentru prelucrarea lor în mod corect am creat în ”forms.js” mai multe funcții și obiecte care să știe, în funcție de acțiunea provocată, ce trebuie să se întâmple. Așadar, există funcția ”Message” pentru afișarea mesajelor de alertă în caz de eroare sau succes și funcția ”CallPost” care prin Axios face un apel la URL-ul primit cu datele aferente executând apoi două apeluri inverse în caz de eroare sau succes.

Obiectul ”forms” conține o funcție ”init” care primește ca și parametru id-ul formularului. În plus, pentru a nu întâmpina probleme precum că un utilizator apasă de mai multe ori pe un buton, creând astfel mai multe execuții care nu sunt necesare, se vor folosi două funcții pentru dezactivarea și activarea unui buton. Tot în cadrul obiectului și funcției se află încă un obiect ”settings” unde sunt definite toate formularele după id-ul lor și apelează funcția ”CallPost” sau acțiunea corespunzătoare pentru fiecare formular. În afara obiectului ”forms” se găsește obiectul ”Actions” care conține toate acțiunile posibile și prelucrează funcționalitățile acestora.

Figura 15 conține codul care implementează acțiunea de ștergere a unei fotografii.

```
deletePhoto: function(albumId, photoId) {
    CallPost('/api/photos/delete-photo', 'album_id='+albumId+'&photo_id='+photoId, function(response) {
        alert('Fotografia a fost ștearsă definitiv');
        window.location.reload();
    }, function(error) {
        console.error(error);
    });
},
```

Figura 15. Codul care implementează acțiunea de ștergere a unei fotografii

După cum se poate observa în figura de mai sus, acțiunea de ștergere a unei fotografii reprezintă o funcție care are ca parametrii id-ul albumului din care face parte poza și id-ul acesteia. Aceasta face un apel prin Axios la ”api.js” cu calea ”/photos/delete-photo” în care trimite id-ul albumului și a imaginii. După ce API-ul trimite răspunsul înapoi, dacă aceste este ok atunci utilizatorul va primi mesajul că ”Fotografia a fost ștearsă definitiv” și se va face o reîmprospătare a paginii, altfel va fi afișată în consolă eroarea întâmpinată.

În urma implementării, aplicația pentru utilizator arată conform figurilor de mai jos. În Figura 16 este prezentată pagina de autentificare a unui utilizator în aplicația collectIn, fiind și prima pagină care apare la orice accesare nouă a aplicației.

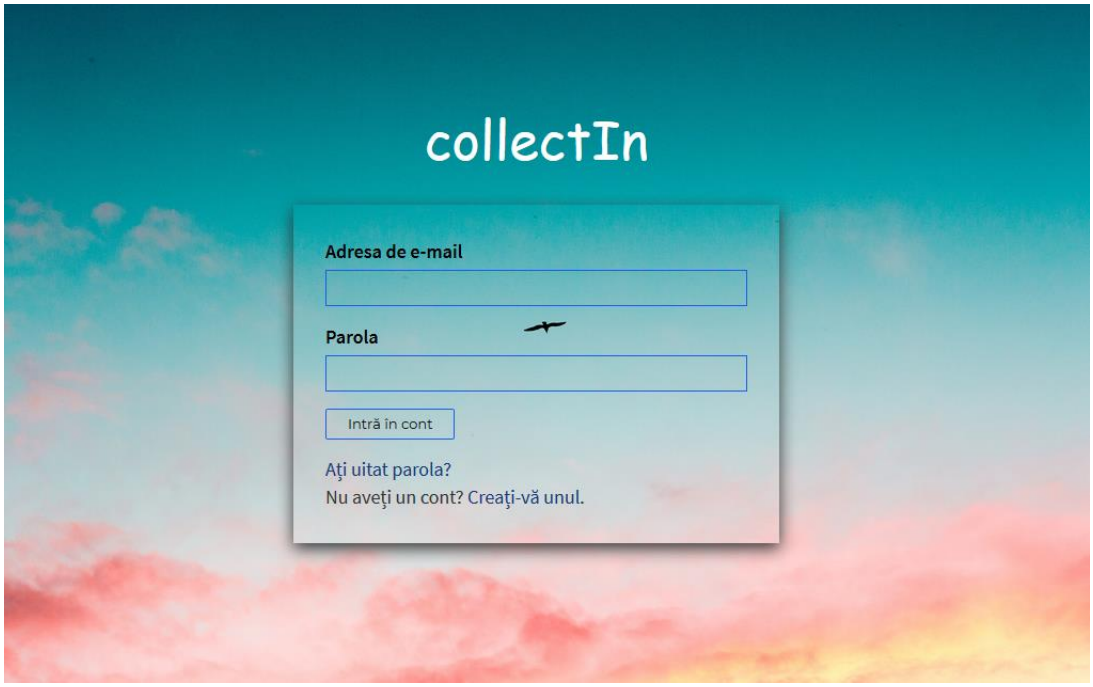

Figura 16. Pagina de autentificare a aplicației

Prima figură reprezintă pagina de autentificare în aplicația collectIn. Se poate observa imaginea de fundal simplă, cu culori deschise care se îmbină creând o ambianță liberă și plăcută la vizualizare, exprimând un sentiment de încredere și siguranță. Formularul alcătuit doar din două câmpuri, adresa de e-mail și parola utilizatorului și legătura către pagina de înregistrare, dacă persoana care a accesat aplicația nu are încă un cont.

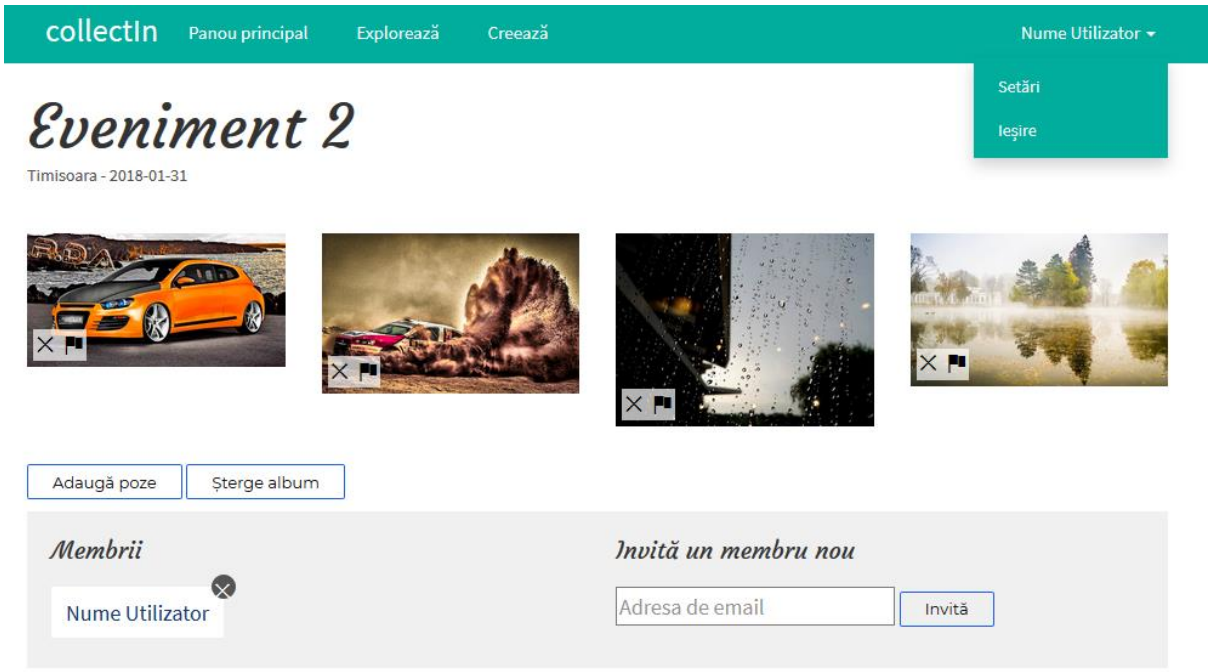

Figura 17. Pagina unui album

A doua figură prezintă modul în care ar arăta pagina unui album creat de utilizator. În partea de sus se constată meniul de navigație care cuprinde, după cum spuneam, secțiunile de ”Panou principal”, ”Explorează” și ”Creează”. Pe săgeata din partea stângă aflată lângă numele utilizatorului se deschide un submeniu cu legăturile către pagina de setări și ieșirea din cont. Se remarcă simplitatea paginii și a culorilor folosite, fundal alb pune în evidență fotografiile evenimentului și a detaliile acestuia: numele, locația unde a avut și data la care s-a organizat. După aceste informații se afișează pozele aferente evenimentului, în cazul acestei figuri sunt adăugate niște poze doar pentru prezentare și funcționalitățile pe care utilizatorul poate să le facă. Iconița de X reprezintă butonul de ștergere a unei fotografii, în timp ce steguleț setează fotografia de copertă. Cele două butoane de mai jos, oferă utilizatorului posibilitatea de a mai adăuga poze sau de a șterge tot albumul. De asemenea, se poate observa partea de membrii, singurul membru în acest moment fiind utilizatorul care a creat albumul, dar și modul în care acesta poate invita un membru nou pe baza adresei de e-mail.

Concluzii

Fotografiile vor fi întotdeauna cel mai folosit mijloc pentru păstrarea amintirilor și experiențelor din viața oamenilor. Nevoia de capturare a momentelor importante din cadrul diferitelor evenimente private sau publice există în fiecare dintre noi deoarece după trecerea timpului doar acele imagini ne mai pot reda, într-o oarecare măsură, ceea ce am trăit la un moment dat. În plus, tot prin intermediul pozelor putem să împărtășim acele clipe cu persoanele apropiate sau chiar să comunicăm un anumit mesaj oricărui om care este interesat să ”asculte”.

Aplicația web dezvoltată în acest proiect este destinată publicului larg care are nevoie de un loc sigur și accesibil pentru stocarea fotografiilor sale. Interfața sistemului este simplă, ușor de folosit, intuitivă și se poate adapta în funcție de mărimea oricărui dispozitiv de pe care este accesată. Obiectivul principal al aplicației a fost crearea unui sistem care să permită depozitarea imaginilor din cadrul evenimentelor private sau publice cu scopul de a le putea și împărți cu alte persoane în cel mai simplu mod posibil, mai exact prin acordarea accesului persoanelor respective la album.

Acestea fiind spuse, consider că prin dezvoltarea acestei aplicații am rezolvat anumite probleme întâmpinate în societate, precum modul de transmitere a fotografiilor de la o persoană la alta, spațiul de stocare necesar pentru păstrarea acestora și siguranța că nu vor fi pierdute în cazul unei defecțiuni al calculatorului personal, telefonului, tabletei.

Bineînțeles, aplicația de față permite posibilitatea unor dezvoltări ulterioare care pot să cuprindă noi funcționalități neimplementate încă, dar și alte dezvoltări mai majore pentru îmbunătățirea maximă a experiențelor utilizatorilor cu sistemul.

O primă dezvoltare ar putea fi adăugarea unui server de e-mail prin care un utilizator să își poată recupera parola, dacă acesta a uitat-o, dar și să trimită o invitație către membrul pe care vrea să îl invite la evenimentul creat. Aceste acțiuni se pot implementa doar cu ajutorul unui server de e-mail deoarece fiecare e-mail trimis trece printr-o serie de servere de-a lungul drumului către destinatarul ales. Fără această serie de servere de poștă electronică nu se vor putea trimite e-mailurile dorite oricărui utilizator, indiferent de domeniul adresei de e-mail al acestuia.

În plus, cu toate că aplicația permite accesarea ei și de pe telefon sau tabletă, s-ar putea dezvolta și o aplicație mobilă care să ofere un mod de lucru mai potrivit în cazul acestor dispozitive, mai ales că sunt obiectele cele mai folosite de către majoritatea persoanelor care nu dețin un aparat fotografic performant atunci când vine vorba de fotografierea unui moment.

Bibliografie

[1] Douglas, Norman, Douglas, Ngaire and Derrett, Ros. 2001. Special Interest Tourism. Milton. John Wiley & Sons Australia, Ltd. 356 - accesat la 4 aprilie 2019

[2] Declan O'Neil, 6 Reason Why Photography Matters, disponibil online la https://digital-photography-school.com/6-reasons-why-photography-matters/ - accesat la 5 aprilie 2019

[3] mdnwebdocs-bot, Masahiro Fujimoto, Chris Mills și alți contribuabili, What is a web server?, disponibil online la https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server - accesat la 10 aprilie 2019

[4] Web Technology Surveys, Usage of web servers, disponibil online la https://w3techs.com/technologies/overview/web_server/all - accesat la 10 aprilie 2019

[5] Rick Cattell, Scalable SQL and NoSQL Data Stores, SIGMON Record, Originally published in 2010, last revised December 2011, Vol 39, Nr 4., disponibil online la http://www.cattell.net/datastores/Datastores.pdf - accesat la 16 mai 2019

[6] Node.js Pagina Oficială, disponibil online la https://nodejs.org/en/ - accesat la 15 mai 2019

[7] Erik Eloff, Daniel Torstensson, An Investigation into the Applicability of Node.js as a Platform for Web Service. Linköping: Institutionen för datavetenskap; 2012. Disponibil online la https://www.scribd.com/document/378217750/an-Investigation-Into-the-Applicability-of-Node-js-as-a-Platform-ForWeb-Services-by-Erik-Eloff-Daniel-Torstensson-2012 - accesat la 16 mai 2019

[8] ExpressJS Pagina oficială, disponibil la http://expressjs.com/ - accesat la 16 mai 2019

[9] Nick Karnik, Introduction to Mongoose for MongoDB, disponibil online la https://www.freecodecamp.org/news/introduction-to-mongoose-for-mongodb-d2a7aa593c57/ - accesat la 16 mai 2019

[10] Flickr Pagina oficială, diponibil la https://www.flickr.com/about - accesat la 03 iunie 2019