

UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” din IAȘI  
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE  
DOMENIUL: Calculatoare și Tehnologia Informației  
SPECIALIZAREA: Calculatoare

# **Soluție de stocare a fișierelor Aplicație web**

LUCRARE DE DIPLOMĂ

Absolvent  
Ionuț Balașanu

Coordonator științific  
Ș.l.dr.ing. Iosif Iulian Petrila

Iași, 2019

DECLARAȚIE DE ASUMARE A AUTENTICITĂȚII  
LUCRĂRII DE DIPLOMĂ

Subsemnatul(a) IONUȚ BALAȘANU,  
legitimat(ă) cu CI seria XT nr. 689078, CNP 1970130073820  
autorul lucrării SOLUȚIE DE STOCARE A FIȘIERELOR – APLICAȚIE WEB

---

elaborată în vederea susținerii examenului de finalizare a studiilor de licență organizat de către Facultatea de Automatică și Calculatoare din cadrul Universității Tehnice „Gheorghe Asachi” din Iași, sesiunea IULIE a anului universitar 2019, luând în considerare conținutul Art. 34 din Codul de etică universitară al Universității Tehnice „Gheorghe Asachi” din Iași (Manualul Procedurilor, UTI.POM.02 – Funcționarea Comisiei de etică universitară), declar pe proprie răspundere, că această lucrare este rezultatul propriei activități intelectuale, nu conține porțiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislației române (legea 8/1996) și a convențiilor internaționale privind drepturile de autor.

Data

04.07.2019

Semnătura

# Cuprins

|  |    |
|--|----|
| Introducere.....   | 1  |
| Capitolul 1. Concepte de bază.....   | 3  |
| 1.1. Domeniul și contextul abordării temei.....                                      | 3  |
| 1.1.1. Cloud Computing.....  | 3  |
| 1.1.2. Cloud Storage.....  | 4  |
| 1.2. Realizări actuale în domeniul Cloud Computing/Cloud Storage.....                | 7  |
| 1.2.1. Cloud Computing.....  | 7  |
| 1.2.2. Cloud Storage.....  | 8  |
| 1.3. Aplicații Cloud Existente.....  | 9  |
| 1.3.1. Google Drive.....   | 9  |
| 1.3.2. Dropbox.....  | 10 |
| 1.4. Noțiuni teoretice privind tehnologiile utilizate.....                           | 11 |
| 1.4.1. Medii de dezvoltare.....  | 11 |
| 1.4.1.1. Microsoft Visual Studio 2019.....   | 11 |
| 1.4.1.2. Microsoft SQL Server Management.....  | 11 |
| 1.4.2. Tehnologii software utilizate.....  | 12 |
| 1.4.2.1. ASP.NET CORE 2.0.....   | 12 |
| 1.4.2.2. JavaScript.....   | 14 |
| 1.4.2.3. Cascading Style Sheets (CSS).....   | 15 |
| 1.4.2.4. HyperText Markup Language (HTML).....                                       | 15 |
| 1.4.2.5. JSON.....   | 16 |
| Capitolul 2. Proiectarea aplicației.....   | 17 |
| 2.1. Baza de date.....   | 19 |
| 2.2. Aplicația de stocare a fișierelor.....  | 20 |
| Capitolul 3. Implementarea aplicației de stocare a fișierelor.....                   | 25 |
| 3.1. Descrierea generală a implementării.....  | 25 |
| 3.2. Probleme întâmpinate și modalități de rezolvare.....                            | 25 |
| 3.3. Funcționarea aplicației.....  | 26 |
| Capitolul 4. Testarea aplicației.....  | 29 |
| 4.1. Testarea realizării directoarelor individuale.....                              | 29 |
| 4.2. Testarea afișării individuale a fișierelor încărcate de fiecare utilizator..... | 30 |
| 4.3. Testarea încărcării mai multor fișiere în același timp.....                     | 31 |
| Concluzii.....   | 33 |
| Bibliografie.....  | 34 |
| Anexe.....   | 36 |
| Anexa 1. Stabilire conexiune cu baza de date.....                                    | 36 |
| Anexa 2. Creare directoare individuale.....  | 37 |
| Anexa 3. Funcționalități.....  | 38 |
| Anexa 4. Interfață.....  | 41 |

# Soluție de stocare a fișierelor – Aplicație web

Ionuț Balașanu

## Rezumat

Aplicația creată în scopul realizării lucrării de licență se încadrează în domeniul stocării fișierelor în cloud. Acest domeniu este unul amplu și are ca scop furnizarea anumitor servicii către utilizatori prin intermediul internetului.

În acest proiect este descris în mod detaliat, de la documentarea teoretică și până la implementarea efectivă, modul în care se poate dezvolta o aplicație web ce oferă servicii de stocare a fișierelor. O astfel de aplicație are rolul de a ajuta utilizatorii atât prin spațiul de stocare pe care îl oferă, cât și prin funcțiile de accesare a fișierelor de pe orice dispozitiv sau partajare către alți utilizatori.

Aplicația descrisă în cadrul acestei lucrări își propune să ofere o parte din funcționalitățile actualelor servicii de cloud. Astfel încât, obiectivul principal al aplicației este să ofere posibilitatea încărcării, stocării și descărcării fișierelor.

Față de serviciile actuale din domeniu, aplicația își propune să ofere un spațiu de stocare mai mare, fără a fi necesară plata unui abonament lunar. De asemenea, față de serviciile actuale, care pun la dispoziție o interfață greu de utilizat, această aplicație oferă o interfață cât se poate de simplă și intuitivă.

Pe parcursul realizării aplicației s-au dezvoltat metode de încărcare, stocare și descărcare a unor fișiere. Modul de dezvoltare al acestor metode este open-source și este disponibil în articolul tehnic „Upload/Download Files în ASP.NET Core 2.0”, scris de către Tahir Naushad. De asemenea, s-a implementat o interfață prin care utilizatorii pot vizualiza fișierele pe care le-au încărcat. Toate aceste implementări au fost posibile doar după o documentare amănunțită atât în ceea ce privește modul în care au fost realizate serviciile actuale, cât și în ceea ce privește tehnologiile folosite.

Având în vedere că aplicația creată oferă toate funcționalitățile enumerate mai sus, se poate spune că obiectivul principal, stabilit odată cu alegerea temei, a fost atins.

## Introducere

Odată cu apariția internetului au fost dezvoltate și numeroase servicii cu scopul de a ne ușura munca. „*Internetul reprezintă o rețea internațională de calculatoare, formată prin interconectarea rețelelor locale și globale, destinată să faciliteze schimbul de date și informații din diferite domenii*”[1].

Primele cercetări în acest sens au început în 1960, când guvernul Statelor Unite ale Americii a alocat fonduri în scopul cercetării și construirii unei rețele de comunicații între calculatoare [1]. Din acel moment, cloud computing-ul a evoluat constant, cel mai recent pas în acest sens fiind apariția Web 2.0.

Totuși, serviciile de cloud oferite maselor de utilizatori au fost dezvoltate mai târziu. Prima aplicație asemănătoare cu percepția actuală asupra cloud-ului a fost Salesforce. Aceștia au fost primii care au implementat cu succes această paradigmă, utilizându-se de internet pentru a furniza soft-uri. Acestea puteau fi accesate și descărcate de către oricine avea o conexiune la internet [2].

Un pion important în dezvoltarea serviciilor de cloud la care avem noi acces acum a fost compania Amazon. În 2002, aceștia au lansat pe piață Amazon Web Services, oferind o serie de servicii cloud, precum stocarea, calculul sau chiar un serviciu inteligent, denumit Amazon Mechanical Turk, ce era folosit de către anumite întreprinderi pentru a angaja oameni care să lucreze de la distanță, în scopul realizării anumitor sarcini pe care calculatoarele încă nu le puteau realiza .

Unele dintre cele mai cunoscute și utilizate servicii dezvoltate pe internet sunt serviciile de socializare, serviciile de poștă electronică(email) și nu în cele din urmă serviciile de stocare în cloud. Acest mod de stocare a fișierelor se realizează cu ajutorul unui serviciu de găzduire pe internet. Acesta ne oferă posibilitatea să accesăm oricând, oriunde și de pe orice dispozitiv fișierele pe care le-am încărcat și stocat acolo.

Încă din momentul apariției primelor servicii de acest gen s-a declanșat o stare de curiozitate în legătură cu modul în care au fost create. Acesta a fost și motivul principal pentru care s-a ales această temă în vederea realizării proiectului de licență.

Un al doilea motiv ar fi faptul că nu de puține ori serviciile actuale de stocare în cloud au dezamăgit. Fie că nu s-a oferit acces la suficient spațiu de stocare, fie ca unele fișiere nu au fost acceptate pe motiv că au o dimensiune prea mare sau că unele formate nu erau suportate, serviciile actuale de stocare au pus și câteva probleme utilizatorilor. Având la dispoziție internetul și marea vastă de informații ce se pot obține de aici, s-a decis crearea unui serviciu de stocare, care să rezolve aceste probleme.

În momentul de față există numeroase companii care oferă spațiu de stocare în cloud. Printre cele mai cunoscute sunt Google Drive și Dropbox. Pe lângă spațiul de stocare pe care îl oferă, aceste aplicații ne pun la dispoziție și funcții precum partajarea fișierelor între utilizatori – fie creând link-uri publice către fișierele respective, fie oferind acces privat unor anumiți utilizatori - sau modificarea fișierelor direct în cloud, fără a le mai descărca pe propriul dispozitiv.

Aplicația creată își propune să realizeze câteva dintre funcționalitățile oferite de către aceste companii. Obiectivul principal al aplicației este reprezentat de încărcarea fișierelor pe server în vederea stocării acestora. Din acest moment fișierele pot fi accesate și utilizate de oriunde și de pe orice dispozitiv care suportă rularea unei aplicații web.

Așadar, în momentul accesării aplicației fiecare utilizator își poate crea un cont personal, utilizând adresa de email. Informațiile personale ale utilizatorilor, precum adresa de email sau

parola aleasă în momentul înregistrării vor fi stocate într-o bază de date.

În vederea realizării acestor obiective s-a folosit Microsoft Visual Studio 2019 pentru dezvoltarea efectivă a aplicației și SQL Server Management Studio pentru server și stocarea informațiilor despre utilizatori în baza de date. În ceea ce privește tehnologiile folosite în dezvoltarea aplicației s-a optat pentru ASP.NET CORE, având la bază limbajul de programare C Sharp. S-a utilizat acest framework folosind modelul arhitectural Model-View-Controller (MVC). Pe lângă limbajul de bază(C#), acest model arhitectural utilizează mai multe tehnologii web, precum:

- CSS – folosit pentru modul în care arată aplicația web,
- JavaScript- folosit pentru implementarea scripturilor ce urmează a fi rulate direct în browser
- HTML-pentru scheletul efectiv al paginii web,
- JQuery-folosit cu același scop ca și JavaScript, însă cu câteva îmbunătățiri în ceea ce privește unele procese pe care le realizează.

În primul capitol se găsesc noțiuni generale despre ce înseamnă cloud, ce înseamnă o aplicație de tip cloud, dar și câteva exemple de aplicații disponibile în momentul de față. În cel de-al doilea capitol s-au prezentat câteva noțiuni teoretice despre tehnologiile utilizate, iar mai apoi s-a explicat în detaliu cum s-a proiectat aplicația. În ultimele doua capitole se găsesc câteva detalii despre modul în care funcționează aplicația creată și cum s-a comportat aceasta la diferite teste realizate asupra ei.

## Capitolul 1. Concepte de bază

### 1.1. Domeniul și contextul abordării temei

Termenul de cloud luat la modul literal și tradus în limba română înseamnă „nor”, însă în domeniul informatic, acest cuvânt are o altă reprezentare. Astfel încât, termenul „cloud” este folosit în acest context atunci când detaliile tehnice ale internetului nu contează. Acest termen a fost folosit pentru prima dată în acest sens în anul 2006 [3].

Cloud-ul reprezintă de fapt un ansamblu de servicii disponibil utilizatorului prin intermediul internetului, fără ca acesta să cunoască modul în care au fost configurate sistemele care furnizează aceste servicii și nici locul în care acestea sunt amplasate. Aceste servicii au luat naștere o dată cu evoluția numărului de oameni care beneficiază de o conexiune la internet. Astfel încât, în momentul de față aproape toate resursele pot fi plasate, iar mai apoi partajate, prin internet (vezi Figura 1.1 ) [3].



Figura 1.1. Diagramă Cloud [21]

La rândul său, cloud-ul se împarte în alte două categorii: Cloud Computing și Cloud Storage.

#### 1.1.1. Cloud Computing

În momentul de față există un major interes asupra serviciilor de tip Cloud Computing. Atât companiile, cât și consumatorii de rând apelează tot mai des la un serviciu Cloud, în detrimentul rulării aplicațiilor pe stația personală de lucru. Acest lucru se întâmplă deoarece

furnizorii de servicii cloud oferă, pe lângă un spațiu în care utilizatorul să își poată rula aplicația, și o interfață foarte ușor de înțeles. Prin urmare, utilizatorul nu mai este nevoit să cunoască neapărat tehnologiile folosite, serviciul fiind automatizat cât mai mult, iar funcționalitățile puse la dispoziție printr-o interfață grafică (vezi Figura 1.2).

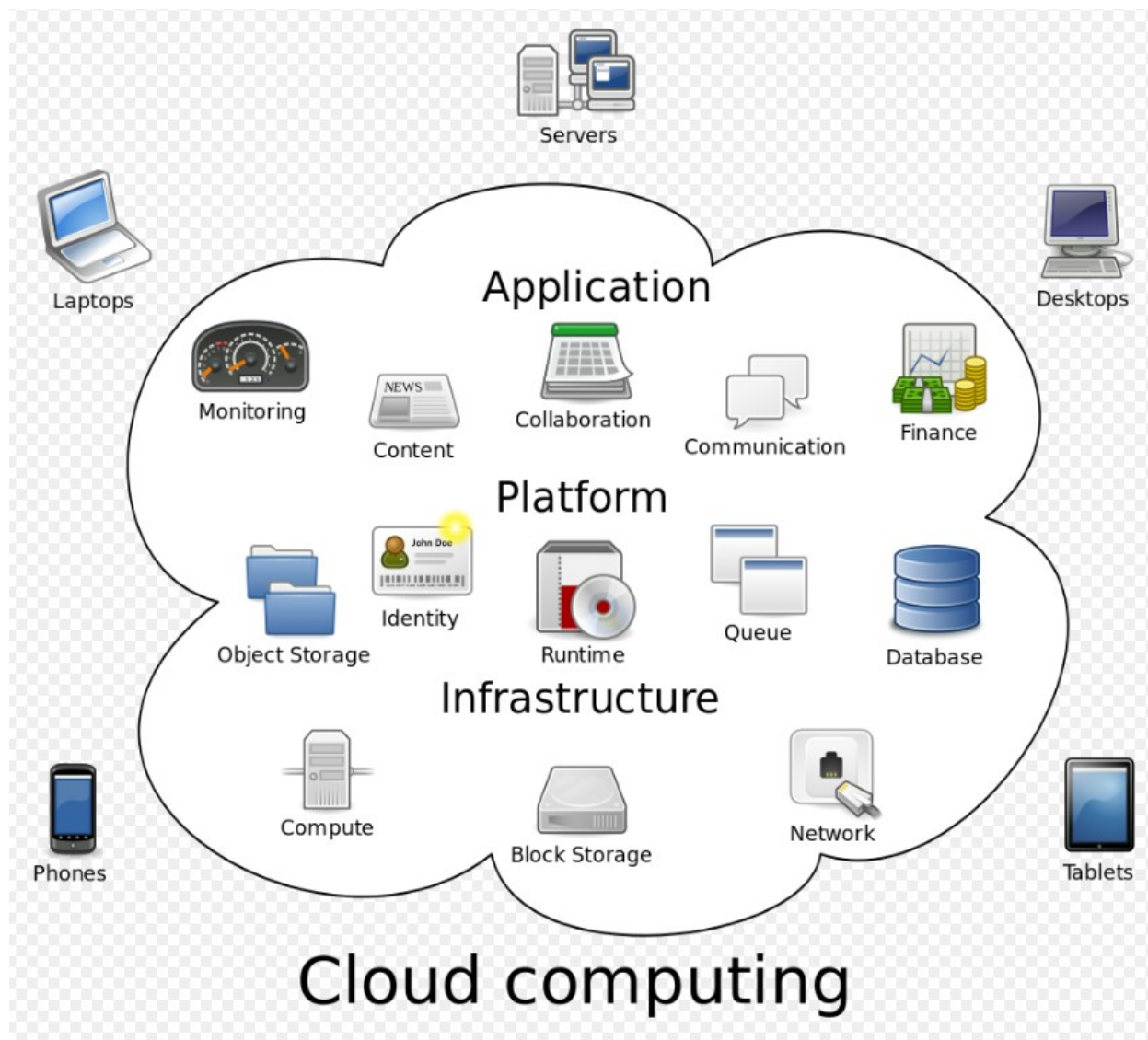


Figura 1.2. Diagrama conceptuală Cloud Computing [22]

Un avantaj important în alegerea unui astfel de serviciu, îl constituie viteza de calcul și capacitatea de stocare sporită, fără necesitatea unei investiții în propria configurație [4].

### 1.1.2. Cloud Storage

Serviciile de stocare a fișierelor accesibile maselor de oameni nu sunt disponibile de foarte mult timp în mediul online, însă acestea au o bază destul de veche. Primul pas spre serviciile oferite în momentul de față a fost făcut încă din 1963, atunci când MIT a primit niște fonduri substanțiale cu scopul folosirii acestora în domeniul cercetării. În cadrul domeniilor de cercetare pe care MIT urma să le exploateze cu ajutorul acelor fonduri a fost stipulată și o clauză care cerea dezvoltarea unei tehnologii care să ofere posibilitatea accesării aceluiași computer de



către doi sau mai mulți oameni simultan. Acesta a fost punctul de plecare în evoluția acestui domeniu, chiar dacă la început drumul nu părea a fi deloc în direcția în care trebuie [3].

Totuși, primii care au oferit un serviciu exclusiv pentru stocarea fișierelor personale au fost cei de la Google. Desigur, există mai multe tipuri de aplicații web bazate pe această tehnologie, nu doar serviciile clasice de stocare a fișierelor. Un serviciu care se folosește de această paradigmă, însă nu oferă posibilitatea stocării fișierelor personale este Netflix. Aceasta a fost și prima platformă de video-streaming în care filmele erau stocate în cloud [3].

Un serviciu de stocare a fișierelor reprezintă de fapt o aplicație web ce are ca și scop exclusiv încărcarea și stocarea fișierelor utilizatorilor. La rândul ei, o aplicație web este construită pe baza unui program de tipul client-server.

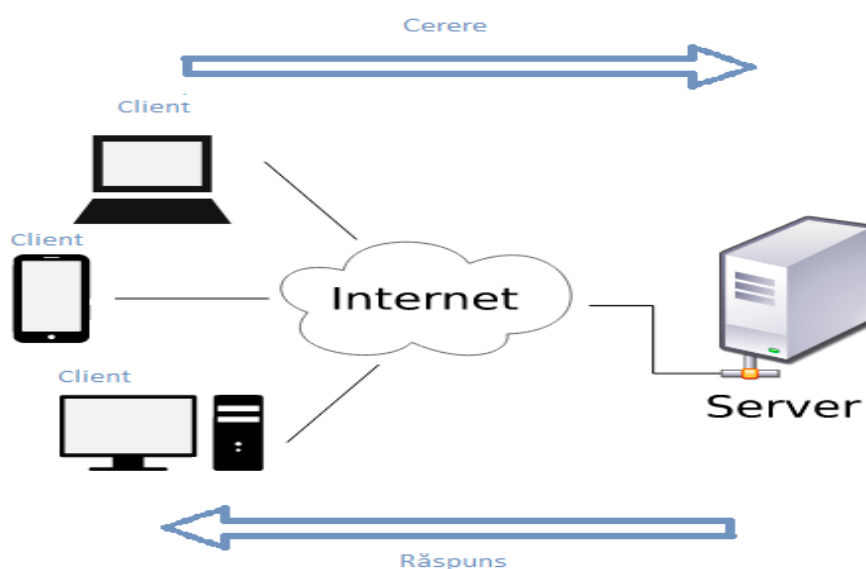


Figura 1.3. Modelul Client-Server

Modelul Client-Server este reprezentat de o relație în care clientul face o cerere de utilizare a unui serviciu sau a unei resurse către server, iar acesta îi oferă un răspuns. Atât cererile făcute de către client, cât și răspunsurile oferite de către server sunt transmise prin internet. La început, această arhitectură era folosită doar în sistemele distribuite, însă în zilele noastre cea mai mare parte a aplicațiilor web destinate maselor de utilizatori disponibile pe internet se bazează pe modelul Client-Server (vezi Figura 1.3).

În acest model arhitectural, clientul care generează cererea poate fi reprezentat de către orice dispozitiv cu acces la internet: telefon, tableta, laptop etc. Un avantaj al acestor aplicații web este reprezentat de numărul infinit de utilizatori care pot genera cereri și pot avea acces la resursele sau serviciile oferite de către server. În același timp, partea de sever este singura din această relație care pune la dispoziție servicii sau resurse, clientul având doar posibilitatea de a formula cereri [3].

Aplicațiile de stocare în web a fișierelor fac parte din așa numitul Personal Cloud Storage. Acesta oferă utilizatorilor dreptul de a stoca orice tip de fișiere (de la text până la videoclipuri sau muzică). Utilizatorii au drepturi depline asupra spațiului în care le sunt stocate fișierele, iar aceștia le pot accesa de oriunde prin intermediul internetului. Pentru a privi lucrurile într-un mod mai simplu, ne putem imagina ca acest spațiu reprezintă de fapt un hard-disk fizic de

stocare a datelor pe care îl putem lua cu noi oriunde.

Este foarte greu să facem o diferență între cloud și internetul în sine. Ce putem spune este că serviciile de cloud au nevoie neapărat de internet. La bază, atât internetul, cât și cloud-ul au scopul de a stabili o legătură între diferite dispozitive aflate în locații diferite.

Internetul include o serie de rețele ce oferă posibilitatea comunicării hardware și software în vederea transferului de date. Pentru a realiza această comunicare, internetul se folosește de un protocol, denumit IP/TCP. Acest protocol poartă denumirea completă de Internet Protocol/Transmission Control Protocol și este de fapt o suită de protocoale de comunicare ce ajută la interconectarea dispozitivelor prin internet. TCP/IP are rolul de a specifica cum sunt schimbate datele pe internet. Acesta specifică exact cum sunt împărțite datele în pachete, cum sunt adresate, cum sunt transmise, pe ce rute sunt transmise și cum sunt recepționate la destinație [5].

De cealaltă parte, cloud-ul oferă acces asupra unor resurse ce pot fi folosite ca și servicii în scopul rezolvării a diferite probleme.

Acest tip de stocare a fișierelor personale oferă numeroase avantaje, față de modul clasic de stocare locală a fișierelor:

- **Experiența de utilizare și accesibilitatea** – o aplicație de acest gen este foarte ușor de utilizat. Pentru a încărca fișierele, utilizatorii nu au nevoie de cunoștințe avansate în domeniu, ci pot efectiv să tragă fișierele dorite în aplicație, iar acestea vor fi preluate, încărcate și stocate automat pe server. În același timp, aceste servicii oferă și posibilitatea accesării fișierelor de oriunde, atât timp cât există o conexiune la internet pe dispozitivul de pe care cineva încearcă să își acceseze fișierele personale [5].
- **Backup** – pierderea unor fișiere personale poate fi foarte neplăcută în momentul în care nu există nici un plan de rezervă pentru stocarea acestora. Cloud-ul rezolvă și această problemă, astfel încât în momentul unei pierderi de acest gen, utilizatorul să își poată recupera fișierele stocate. Acest lucru este posibil datorită unui backup al fișierelor stocate. Acestea sunt salvate într-o locație îndepărtată, astfel încât acestea să poată fi disponibile pentru accesare oricând [5].
- **Securitate** – serviciile de acest tip oferă o siguranță destul de mare asupra fișierelor pe care le păstrăm în cloud. Atunci când un utilizator încarcă unul sau mai multe fișiere, acestea sunt stocate și protejate împotriva oricărui tip de defecțiune hardware [5].
- **Costuri** – în cazul în care este utilizat în cadrul unei companii putem reduce considerabil costurile cu un astfel de serviciu [5].
- **Distribuire** – serviciile de stocare în web oferă o modalitate de a distribui fișierele foarte ușoară. Un simplu link de acces este de ajuns pentru ca asupra aceluiasi fișier să aibă acces un număr nelimitat de oameni [5].
- **Automatizarea** – până la apariția acestor servicii, backup-ul datelor reprezenta o problemă destul de mare, fiind nevoie de timp și resurse în plus, mai ales dacă acest lucru trebuia realizat asupra unui număr mare de fișiere. În momentul de față, utilizând tehnologia de stocare în cloud, problema este rezolvată automat.

Utilizatorul trebuie doar să selecteze fișierele cărora dorește să le facă backup și când dorește să se întâmple acest lucru, iar serviciul de cloud v-a realiza acest lucru automat [5].

- Colaborare – aplicațiile de stocare web reprezintă o platformă ideală pentru a lucra în echipă. Acestea oferă posibilitatea ca mai multe persoane să aibă acces și să modifice detalii din același fișier, în același timp. Aceștia pot accesa fișierul de oriunde, iar modificările vor fi vizibile tuturor în același timp [5].
- Flexibilitatea – atunci când dorim să utilizăm un serviciu de acest gen, plătim doar pentru resursele de care avem nevoie [5].
- Sincronizare – în modul clasic, de stocare locală, putem accesa datele doar din locația unde se află dispozitivul pe care le-am stocat. În schimb, cu un serviciu de stocare în cloud, oricare dispozitiv ce suportă accesarea unei pagini web, poate reprezenta un punct de acces către fișierele stocate. De asemenea, această funcționalitate ne scutește și de munca depusă pentru a transfera datele de pe un dispozitiv pe altul. Un alt avantaj al sincronizării fișierelor îl reprezintă faptul că fișierele se actualizează în timp real, astfel încât orice modificare am aduce asupra lor de pe un dispozitiv, aceasta va fi vizibilă pe toate dispozitivele de pe care v-am accesa acele fișiere [5].
- Comoditate – chiar dacă am stoca datele pe un dispozitiv foarte ușor de transportat (cum ar fi un hard-disk sau un stick usb), accesarea acestora tot necesită un minim de muncă fizică. Astfel, utilizând un serviciu de stocare, stresul creat de backup sau disponibilitatea fișierelor dispare complet, aceste procese fiind automatizate [5].

## ***1.2. Realizări actuale în domeniul Cloud Computing/Cloud Storage***

Atât Cloud Computing cât și Cloud Storage reprezintă niște concepte moderne în ceea ce privește realizările din domeniul computerelor și al informaticii. Totuși, evoluția acestora a fost accelerată, iar în momentul de față există numeroase companii care ne pot oferi servicii și funcționalități în acest sens.

### ***1.2.1. Cloud Computing***

Apariția fenomenului de cloud computing a determinat un număr considerabil de companii să ofere o parte din puterea de procesare creată de infrastructura lor, în schimbul unor sume de bani.

În momentul de față, atât numărul cât și tipul serviciilor pe care o companie de acest tip le poate oferi este vast. Atât serverele, platformele hardware, aplicațiile software, spațiul de stocare și toate celelalte componente de infrastructura care urmează a fi la dispoziția utilizatorului aparțin furnizorului de servicii cloud. Aceștia oferă aceste servicii doar la cerere, contra cost unui număr foarte mare de consumatori. Acest lucru face ca serviciile să nu fie foarte

scumpe, iar după un calcul simplu ne putem da seama că este mult mai avantajos să utilizăm serviciile furnizate de o astfel de companie decât să ne construim singuri o platformă cu spațiu de stocare, servere, programe software, echipamente de rețea etc. Acest tip de serviciu este bazat pe o infrastructură de tipul IaaS – Infrastructure as a Service -

Automatizarea cloud-ului este un alt aspect la care s-a lucrat foarte mult în perioada recentă, astfel încât utilizatorii să beneficieze de o experiență cât mai ușoară în folosirea serviciilor. Un procent ridicat al automatizării în ceea ce privește un astfel de serviciu este necesară, atât pentru utilizatori în momentele interacțiunii cu interfața de utilizare a serviciului, cât și pentru o fiabilitate cât mai crescută a sistemelor folosite. O altă consecință a automatizării serviciilor de cloud o reprezintă posibilitatea reducerii angajaților prin folosirea unor servicii care pot face munca respectivă automat.

IBM a început cercetarea în ceea ce privește infrastructurile cloud de tip hybrid. Aceștia au creat un model care oferă utilizatorilor posibilitatea de a accesa unelte de conectivitate, aplicații și nu în cele din urmă platforma de inteligență artificială IBM Watson. Această platforma este de fapt un sistem computerizat care deține capacitatea de a răspunde unor anumite întrebări adresate în limbaj natural [6][7].

Watson are la bază un grup de 90 de servere IBM POWER 750 cu un total de 2880 de procesoare multi-nucleu POWER7 și 16 teraocteți de memorie RAM. Fiecare dintre servere utilizează un procesor de 3,5 GHz POWER7 cu 8 nuclee. Acesta are capacitatea de a înțelege nevoile unui consumator real. Acesta este recunoscut pentru capacitățile sale cognitive, oferind analize interioare care nu ar putea fi văzute și realizate nici de cel mai bine pregătit expert din momentul actual. Watson poate fi ușor configurat pentru a executa cerințe specifice fiecărei organizații în parte [7].

### *1.2.2. Cloud Storage*

La fel ca și în cazul cloud computing, piața de cloud storage a explodat în ultimii ani. Acest lucru se datorează în mare parte și adoptării soluțiilor cloud în domenii precum: sănătate, media, companii de tehnologie etc.

Având în vedere evoluția din ultima perioadă a soluțiilor de tip cloud, funcționalitățile clasice ale unui astfel de sistem nu reprezintă o încercare prea grea pentru dezvoltatori. O realizare de notat, însă, din ultima perioadă ține de sfera securității. Călcâiul lui Ahile în ceea ce privește serviciile cloud o reprezintă modul în care sunt securizate datele. Odata cu evoluția tehnologiilor de spargere a sistemelor informatice (precum programele de tip malware), securitatea datelor este pusă în pericol. Pierderea unor informații stocate în cloud poate conduce la pierdere enormă și în cadrul companiei care deținea acele informații.

Această problemă a fost rezolvată de către de către IAGON printr-o soluție denumită „Secure Lake”. Aceasta se bazează pe o tehnologie de criptare imposibil de penetrat – fișierele sunt tăiate în bucăți cât mai mici, iar fiecare bucată de fișier este criptată separat (vezi Figura 1.4). Această soluție oferă în momentul de față o protecție completă asupra informațiilor stocate de către clienți. În momentul criptării, cel ce deține fișierele primește o cheie de acces, iar singura modalitate prin care informațiile criptate mai pot fi recuperate o reprezintă întocmai această cheie. Această tehnologie ne asigură că în cazul în care serviciul principal de cloud este spart, fișierele noastre nu vor putea fi utilizate[8].

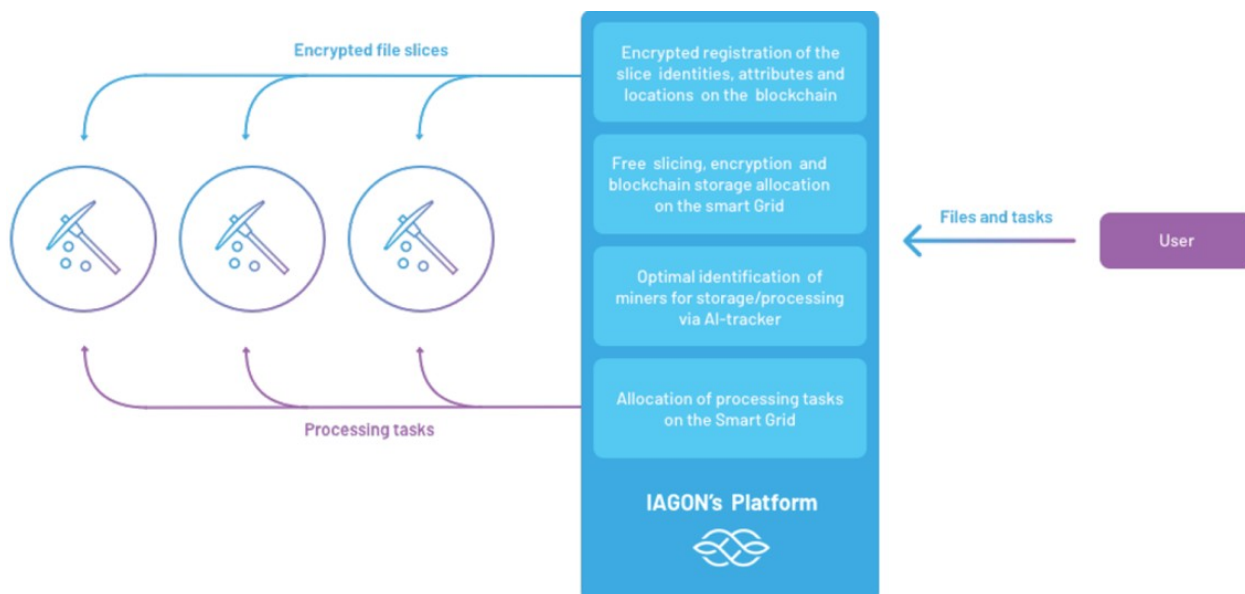


Figura 1.4. Solutia Secure Lake de criptare a datelor[23]

### 1.3. Aplicații Cloud Existente

Dacă facem o scurtă analiză asupra pieței serviciilor de cloud actuale, ne putem da seama că avem o plajă foarte mare de furnizori din care putem alege.

#### 1.3.1. Google Drive

**Google Drive** este unul dintre pionierii serviciilor de acest gen. Această aplicație a fost lansată în 2012 de către compania Google, după cum putem observa și din denumirea acesteia. Pe lângă serviciul de stocare, Google ne oferă și posibilitatea de sincronizare între dispozitive și editarea documentelor direct în cloud. Aceste servicii sunt oferite prin aplicația asociată: Google Docs.

Deși este promovat ca un serviciu gratuit, Google oferă utilizatorilor care nu doresc să plătească doar 15 Gigabytes de stocare. În cazul în care dorim un spațiu mai mare, aceștia nu pot oferi de la 100 Gigabytes până la 30 terabytes, însă contra unui abonament lunar plătit.

O altă funcționalitate foarte utilă oferită de Google Drive este aplicația Backup and Sync, aceasta fiind lansată cu scopul de a înlocui vechile aplicații desktop, Drive și Google Photos. Aceasta oferă acum opțiunea de a face backup la fișiere în Drive cu cea de a adăuga fotografii în Google Photos. Utilizând noua aplicație putem face backup asupra tuturor fișierelor din computer sau doar unor directoare selectate de către noi. În plus, aplicația realizează și o sincronizare ce combină și sincronizează toate directoarele adăugate în aplicație. În acest fel, putem accesa fișierele din contul de Drive chiar dacă la un moment dat nu putem beneficia de o conexiune la internet.

De asemenea, pe lângă serviciile de stocare, Google oferă și accesul asupra altor aplicații cu ajutorul cărora putem edita și crea o gamă mai largă de fișiere: imagini, videoclipuri, documente, proiecte de management, diagrame, tabele etc [9].

Se știe foarte bine că Google este recunoscută pentru faptul că nu își dezvăluie foarte ușor secretele în ceea ce privește tehnologiile utilizate și modul în care acestea au fost folosite. Nu sunt foarte multe informații disponibile despre cum a fost creat serviciul, totuși există câteva

speculații cum că partea de back-end ar fi fost dezvoltată în Python, iar partea de client în Objective-C pentru utilizatorii Mac și wxPython pentru utilizatorii de Windows.

În ceea ce privește sistemele hardware folosite Google nu a fost niciodată genul de companie care să investească în sisteme foarte scumpe. Astfel încât, putem deduce că serverele folosite nu sunt foarte puternice sau foarte avansate. Este foarte probabil ca acestea să fie de fapt niște sisteme normale cărora li se pot adăuga resurse suplimentare cu un minim de investiție.

Aceste sisteme sunt împărțite cel mai probabil între servere pentru aplicație și servere pentru baza de date. Serverele destinate aplicației sunt serverele care vor rula efectiv software-ul din spatele Google Drive, iar cele pentru baza de date sunt folosite pentru stocarea fișierelor [10].

### *1.3.2. Dropbox*

**Dropbox** este un alt serviciu de cloud, înființat în anul 2007 de către Drew Houston și Arash Ferdowsi, ce oferă funcționalități asemănătoare cu cele oferite de către Google. Astfel, Dropbox ne oferă atât un spațiu pe care îl putem utiliza pentru stocarea datelor personale, cât și un serviciu pentru sincronizarea acestora. Aplicația oferă posibilitatea creării unui director special pe fiecare dispozitiv pe care îl asociem cu contul, astfel încât să le poată sincroniza și să creeze impresia că același dosar se află pe fiecare dintre calculatoarele respective.

La fel ca și aplicația celor de la Google, Dropbox se bazează pe un model de afaceri de tip freemium, astfel încât, la înregistrare nu este nevoie de plata vreunei sume de bani, însă utilizatorii care nu doresc să achite un abonament lunar sunt limitați în ceea ce privește spațiul la doar 10 GB. În cazul în care folosim aplicația instalată pe desktop, putem încărca fișiere de orice dimensiune, însă în cazul în care folosim clientul web, încărcarea este limitată la 10 GB per fișier[11].

Desigur, în momentul de față există o gamă mult mai mare de aplicații ce oferă acest tip de servicii, însă Google Drive și Dropbox sunt cel mai bine puse la punct și cele mai utilizate, de altfel. Totuși, ca și aplicații care se apropie foarte mult de funcționalitățile acestor două putem enumera: IDrive, Microsoft OneDrive, Apple iCloud, Sugar Sync. Este o foarte mica diferență, totuși, între ce oferă toate aceste aplicații, prin urmare este foarte greu să facem o alegere și să decidem care aplicație este mai bună. Principalele criterii în acest sens sunt fie spațiul de stocare gratuit care ne este oferit, fie prețul cât mai mic în cazul în care avem nevoie de un abonament plătit.

În prima fază, atât pentru servere cât și pentru implementarea aplicației, cei de la Dropbox au folosit Python. În 2014, însă, aceștia și-au modificat toate serviciile și le-au implementat utilizând limbajul de programare Go. În ceea ce privește partea de website, aceștia au ales ca și tehnologii JavaScript și CoffeeScript.

Partea de hardware a fost cea care le-a pus câteva bețe în roate acestora. Până nu de mult, aceștia foloseau sistemul de stocare de la Amazon, însă, treptat, între 2014 și 2016 aceștia au migrat către propriul sistem hardware.

Pentru partea de sincronizarea și stocare a fișierelor, aceștia au ales să utilizeze protocoale SSL( în momentul de față TLS). Aceste protocoale sunt create special pentru criptarea și securitatea datelor într-o rețea de computere [12][13][14][15].

---

#### ***1.4. Noțiuni teoretice privind tehnologiile utilizate***

În procesul realizării lucrării de licență a fost necesară folosirea mai multor tehnologii software, astfel încât aplicația finală să fie de calitate. Procesul de alegere a tehnologiilor folosite a fost unul destul de greu, având în vedere că aceeași problemă ar fi putut fi rezolvată de către mai multe tehnologii, în mai multe moduri.

##### ***1.4.1. Medii de dezvoltare***

###### ***1.4.1.1. Microsoft Visual Studio 2019***

Visual Studio reprezintă un mediu de dezvoltare integrat (IDE) creat de către compania Microsoft. Acesta poate fi utilizat pentru a dezvolta atât aplicații desktop ( pentru sistemul de operare Windows), site-uri, servicii sau aplicații web, cât și aplicații mobile. Acesta include un editor de cod ce suportă IntelliSense.

Acest mediu de dezvoltare suportă 36 de limbaje de programare diferite, iar editorul de cod și debuggerul pot recunoaște aproape orice limbaj de programare. Acest lucru nu este, însă, adăugat de la bun început în funcționalitățile sale, ci poate fi adăugat de către programator cu ajutorul unui VSPackage. În momentul instalării acestui pachet, funcționalitatea devine disponibilă sub forma unui serviciu. Acest IDE oferă 3 servicii:

- SVsSolution – are rolul de a numerota proiectele;
- SvsUIShell – oferă posibilitatea împărțirii ecranului în mai multe ferestre, în vederea utilizării mai multor unelte în același timp; oferă funcționalități ce țin de interfață;
- SvsShell – se ocupă de înregistrările pachetelor de tip VSPackage

Editorul de cod oferit de către Visual Studio conține funcționalități precum evidențierea cuvintelor cheie din cod, completarea automată a termenilor recunoscuți de către IDE, compactarea unor zone de cod pentru o vizualizare în ansamblu a întregului cod etc.

De asemenea, Visual Studio oferă și un debugger integrat. După cum precizează și cei de la Microsoft în articolul „*Microsoft Visual Studio Debugger*”, acesta lucrează atât ca un debugger la nivel de sursă, cât și ca unul la nivel de mașină. Debuggerul oferă funcționalități precum adăugarea de breakpoint, memory dumps și poate fi lansat și la execuția unui program din afara IDE-ului [16].

###### ***1.4.1.2. Microsoft SQL Server Management***

Microsoft SQL Server Management este un mediu integrat, dezvoltat tot de către Microsoft, ce oferă unelte de administrare a infrastructurii SQL. Apărut în 2005, SQL Server Management are rolul de a ajuta la configurarea și administrarea unui server.

În momentul de față acest mediu dispune de foarte multe funcționalități, însă cea mai importantă este reprezentată de către fereastra de Object Explorer. Aceasta oferă utilizatorului oportunitatea de a vizualiza toate obiectele și caracteristicile specifice serverului respectiv.

Acest mediu suportă o versiune modificată de cei de la Microsoft a limbajului de interogare SQL, unul dintre cele mai folosite limbaje din lume în acest sens. Această versiune poartă numele de Transact-SQL (T-SQL). De fapt, acest T-SQL reprezintă o versiune mai amplă a limbajului de baza SQL, oferind mai multe tipuri de interogări asupra bazelor de date [17].



## 1.4.2. Tehnologii software utilizate

### 1.4.2.1. ASP.NET CORE 2.0

ASP.NET Core reprezintă o arhitectură de tip cross-platform ce se utilizează în scopul dezvoltării de aplicații optimizate pentru internet. O aplicație cross-platform reprezintă un program software ce poate fi executat pe diferite sisteme de operare sau diferite platforme hardware.

Spre deosebire de predecesorul său, ASP.NET 4, acest framework:

- combină logica de dezvoltare a serverului cu cea de dezvoltare a clientului, toate funcțiile fiind scrise în .NET (vezi Figura 1)
- oferă unelte ce ușurează munca în dezvoltarea aplicațiilor web
- oferă posibilitatea de a dezvolta aplicații atât de pe Windows, cât și de pe Linux sau macOS
- oferă un sistem de configurare cloud

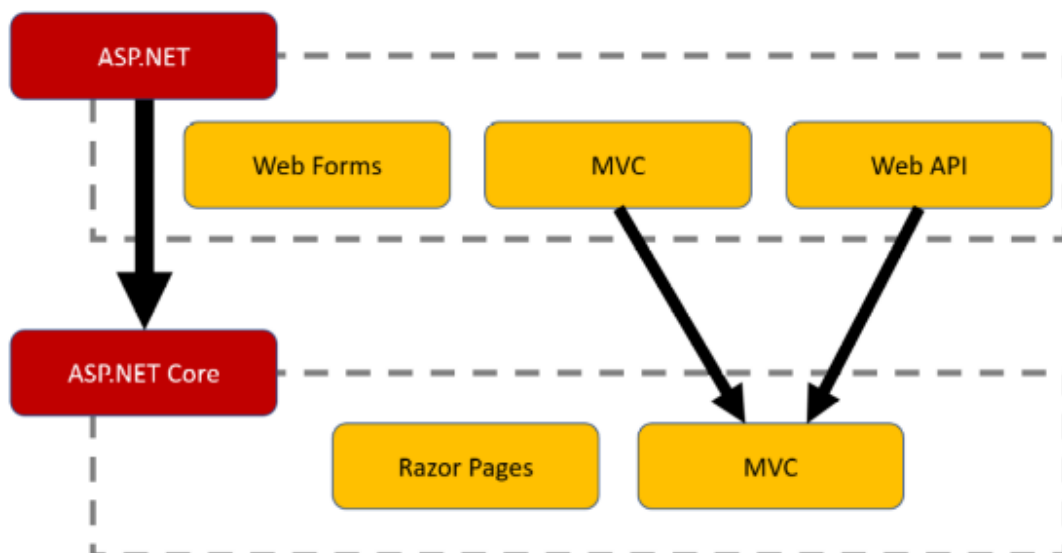


Figura 1: Modificări ASP.NET CORE față de ASP.NET [24]

Utilizând acest framework, avem posibilitatea de a crea interfețe disponibile utilizatorului mai moderne bazându-ne pe tehnologii special create în acest sens precum Blazor, Angular, React sau Bootstrap.

Acest framework este din ce în ce mai utilizat în ultimul timp, iar acest lucru se datorează atât faptului că reprezintă o soluție viabilă în organizarea codului într-un singur proiect, cât și faptului că este open-source, documentația fiind accesibilă tuturor.

Limbajul de programare pe care se bazează acest framework este C#, la fel ca și versiunile apărute anterior. Acesta este un limbaj de programare orientat pe obiect, derivat din C++, dezvoltat de către cei de la Microsoft având ca principal scop înlocuirea limbajului Java.

De asemenea, un avantaj pe care îl oferă acest framework este reprezentat de îmbinarea celor două tehnologii vechi ASP.NET MVC și ASP.NET API formând un singur model de programare.

Modelul arhitectural MVC ( Model-View-Controller) oferit de către dezvoltatorii .NET a reprezentat și punctul de plecare în dezvoltarea aplicației mele. Acesta oferă avantajul de a



împărți aplicația în trei zone de lucru diferite ca și scop, dar interconectate. Astfel încât, o aplicație de tip MVC v-a fi împărțită în zona de model, cea de interfață și nu în cele din urmă, cea de control(vezi Figura 1). Utilizând acest tip arhitectural, putem implementa orice tip de funcționalitate web ce are la bază limbaje precum Python, JavaScript, PHP etc.

Partea de Model este folosită pentru preluarea și prelucrarea datelor care urmează a fi utilizate în program. Printr-un model poate fi descrisă o bază de date, un obiect vizual dintr-un joc sau o anumită informație de tip text. Acesta poate crea relații de tipul unu-la-unu sau unu-la-multi între obiectele pe care le descrie. În mod normal, un model nu este conectat în vreun fel de partea de interfață la care are acces utilizatorul. Acest lucru se realizează cu ajutorul unui controller conectat la model. În cazul în care apare o informație nouă iar modelul se schimbă, acesta trimite un semnal către controller iar acesta din urmă v-a schimba elementele vizibile utilizatorului,

View-ul este reprezentat de către elementele ce sunt vizibile utilizatorului. Aceste elemente au proprietatea de a răspunde la cerințele utilizatorului. Mai exact, un element din interfața destinată utilizatorului realizează o acțiune. În general, aceste elemente sunt de fapt reprezentări grafice ale datelor dintr-un model. La fel ca în cazul zonei de model, partea de view nu are o legătură directă cu cea de model. Astfel încât, atunci când se produce o modificare, aceasta v-a ajunge întâi la controller și abia apoi la model.

După cum ne putem da seama și din explicațiile anterioare, clasele dedicate zonei de controller reprezintă o legătură intermediară între View și Model. În această zonă de cod se creează funcționalitățile efective pe care aplicația în curs de dezvoltare urmează să le aibă. Acesta notifică partea de View în momentul în care s-a realizat o modificare în Model sau invers. În momentul în care modelul se schimbă, acesta anunță partea de interfață să afișeze aceste schimbări și utilizatorului.

În Figura 2 se pot observa relațiile dintre cele trei părți ale arhitecturii Model-View-Controller.

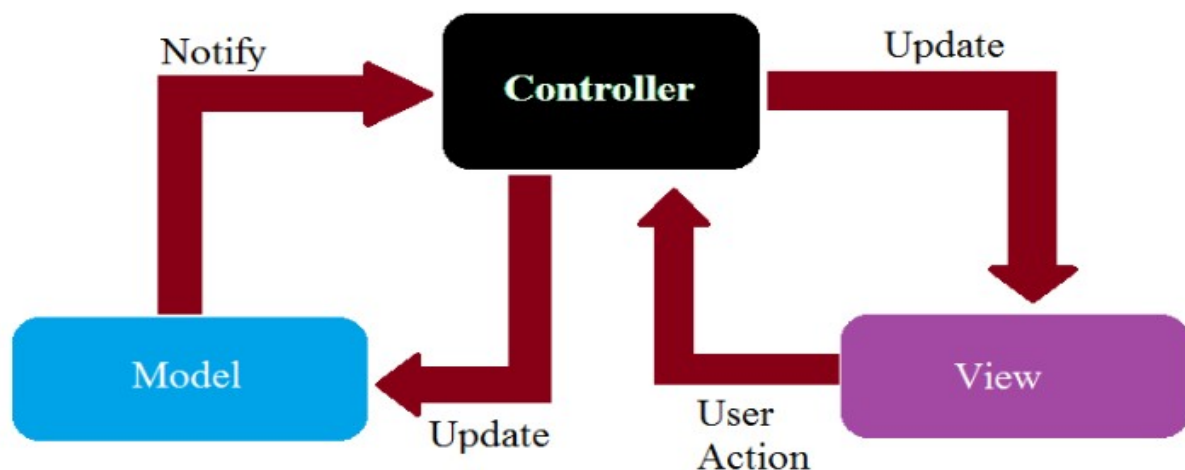


Figura 2. Relațiile Model-View-Controller

### 1.4.2.2. JavaScript

JavaScript este un limbaj de programare de nivel înalt utilizat cu scopul de a face o pagină web interactivă. Acesta a fost dezvoltat și lansat de către Brendan Eich în anul 1995. La momentul actual este cel mai utilizat limbaj de programare în acest sens (vezi Figura 3). Desigur, JavaScript nu este utilizat exclusiv în domeniul dezvoltării paginilor web, ci de asemenea în crearea unor procese în ceea ce privește partea de server sau în domeniul controlerelor de tip embedded [18].

În ceea ce privește utilizarea sa în paginile web, JavaScript este folosit în partea de interfață, vizibilă utilizatorului. În termeni de specialitate, poartă denumirea de limbaj de programare de tip „client-side”. Ce înseamnă de fapt acest lucru? Scripturile create în acest limbaj sunt citite, interpretate și nu în cele din urmă, rulate, direct în browserul utilizatorului. Acest lucru reprezintă un avantaj pentru dezvoltatorii de aplicații web deoarece se pot face modificări constante asupra interfeței grafice fără a fi nevoiți să închidă aplicația actuală și să o ruleze din nou cu toate modificările aduse [18].

Având în vedere că marea majoritate a paginilor/ aplicațiilor web dezvoltate și disponibile pe internet au cel puțin un minim de funcționalități create în JavaScript, s-au dezvoltat în cadrul browserelor motoare de execuție dedicate exclusiv zonei de cod scrise în acest limbaj.

Acest limbaj nu a fost dezvoltat cu scopul de a ajuta la partea de back-end/server a unei aplicații web, ci mai degrabă cu scopul de a face o aplicație mai atractivă din punct de vedere vizual și de a ajuta dezvoltatorii să automatizeze unele funcționalități ale aplicației într-un timp mai scurt și cu un efort mai mic.

Desigur, fiind utilizat de către marea majoritate a aplicațiilor disponibile în internet, acest limbaj include și anumite vulnerabilități în ceea ce privește atacurile cibernetice. Astfel încât, o vulnerabilitate de acest tip poate fi exploatată de către cei ce lansează astfel de atacuri în scopul de a fura informații despre utilizatori sau de a compromite anumite aplicații în ansamblul lor [18]

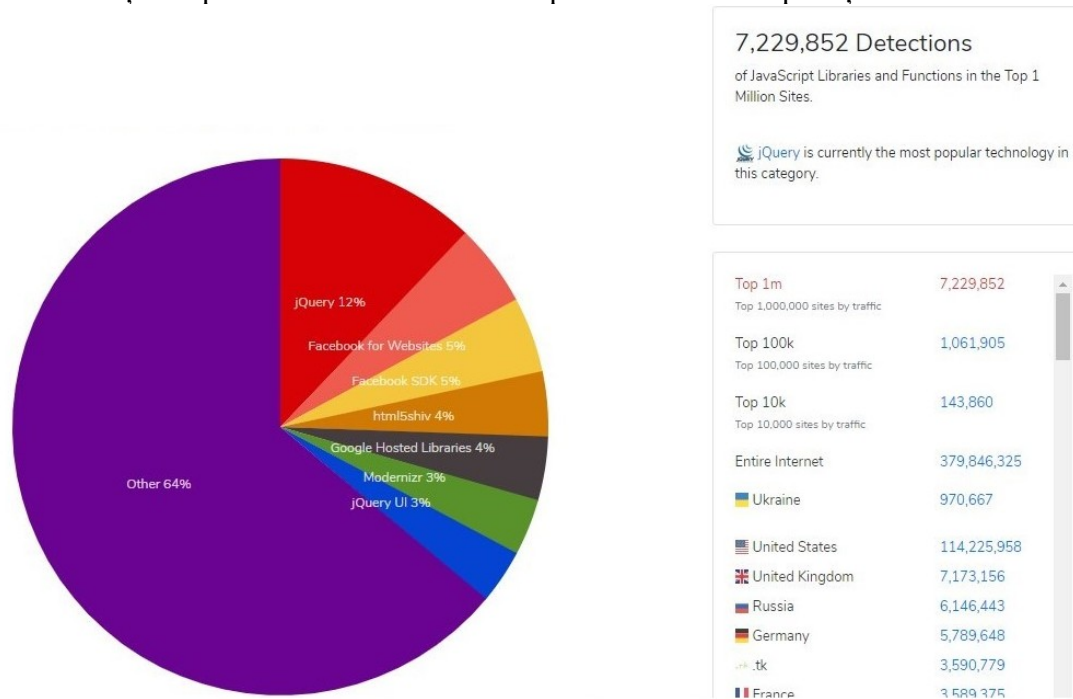


Figura 3. Procentaje utilizare JavaScript în primele 1 milion pagini web [25]

---

### 1.4.2.3. Cascading Style Sheets (CSS)

Cascading Style Sheets este un standard utilizat de către dezvoltatorii de pagini web cu scopul formării elementelor din pagină. Mai exact, utilizând CSS putem oferi atribute legate de design oricărui element creat în HTML. De asemenea, utilizând acest standard, pe lângă modul în care arată un anumit element, putem specifica și modul de prezentare în pagină a acestuia. Acest lucru este foarte util atunci când conținutul unei pagini web trebuie să fie vizualizat pe diferite tipuri de ecrane – telefon, tableta, laptop- [19].

Până nu de mult, codul scris în sensul dezvoltării paginilor web era foarte greu de citit, înțeles sau modificat de către alți programatori din cauza faptului că atât codul HTML, cât și cel pentru design erau scrise în același loc. Acest lucru era realizat de o zonă de cod situată în tagul `<style>`, după cum se poate vedea:

```
<style>
p{
  background-color:green;
  font-size:10px;
}
```

În același timp cu dezvoltarea acestui limbaj, a devenit un standard împărțirea codului în zone diferite, astfel încât zonele de cod HTML să fie destinate doar codului HTML, iar aplicarea designului să fie făcută printr-o legătură către zona de cod CSS. Acest lucru se realizează dezvoltând codul CSS într-un fișier separat, ce poartă extensia `.css`, iar legătura între fișierul HTML și acesta se face astfel [19]:

```
<link rel="stylesheet" type="text/css" href="nume_fisier.css">
```

În ceea ce privește sintaxa, aceasta este una relativ simplă. Codul este împărțit în clase sau id-uri. Clasele se folosesc pentru porțiuni mai mari/ mai generale din pagină, iar id-urile pentru modul în care să arate un singur element din pagină. Același lucru se poate constata și în cazul semanticii. În cele mai multe cazuri, o comandă face fix ce spune numele ei.

Ultima versiune apărută este CSS3. În aceasta au fost adăugate mai multe funcționalități ce pot aduce un plus în dezvoltarea unei aplicații cu o estetică deosebită [19].

### 1.4.2.4. HyperText Markup Language (HTML)

HTML este un limbaj dedicat dezvoltării paginilor web. Utilizând acest limbaj putem reprezenta informații într-o pagină ce poate fi vizibilă de către oricine are acces la internet. Este foarte ușor de înțeles ce își propune acest limbaj dacă ne uităm ce înseamnă exact fiecare cuvânt cheie din denumirea lui.

HyperText reprezintă efectiv metodă ce ne oferă posibilitatea de a naviga pe internet. Acest lucru este posibil datorită hyperlink-urilor. Acestea sunt cele ce ne direcționează către paginile asociate lor. Modul în care tag-urile HTML modifică informațiile din interiorul lor este dat de către Markup. Ultimul cuvânt cheie din denumire este și cel ce a creat câteva controverse. Sunt destule persoane care susțin că HTML nu este un limbaj, însă, din punct de vedere tehnic acesta are toate caracteristicile unui limbaj: sintaxă și semantică [20].

Dezvoltarea paginilor web utilizând acest limbaj este foarte ușoară. Având în vedere că nu are nevoie de compilator, putem crea pagina în orice editor de text, singura cerință fiind aceea de a salva fișierul cu extensia `.html`. Limbajul este foarte ușor de interpretat de către browser. Acesta recunoaște care sunt tagurile specifice limbajului și afișează în pagină informațiile

cuprinse în interiorul acestora [20].

Tag-urile au denumiri specifice și sunt foarte ușor de înțeles, după cum se poate vedea:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Titlul paginii</title>
  </head>
  <body>
    <p>Acest paragraf este in corpul paginii</p>
  </body>
</html>
```

O regulă ce trebuie respectată în utilizarea acestui limbaj este reprezentată de faptul că fiecare tag, o dată ce este deschis, trebuie și închis.

#### 1.4.2.5. JSON

JSON, sau după denumirea sa completă JavaScript Object Notation, a fost creat cu scopul de stoca informațiile într-un mod organizat și ușor de urmărit. Acesta a fost creat în urma apariției și utilizării tot mai frecvente a tehnologiei web AJAX. După apariția acestei tehnologii, a fost necesară găsirea unei modalități ca paginile web să aibă capacitatea de a încărca datele cât mai rapid posibil, evitând în acest fel încărcarea greoaie a acestora.

Sintaxa acestei metode este una destul de ușor de înțeles, după cum se poate observa în exemplul următor:

```
var studenti = {
  "ion" : {
    "nume": "Ion Ionescu",
    "varsta": "22",
    "sex": "masculin"
  },
  "andreea": {
    "nume": "Andreea Andreescu",
    "varsta": "23",
    "sex": "feminin"
  },
  "marian": {
    "nume": "Marian Marinescu",
    "varsta": "21",
    "sex": "masculin".
  }
}
```

## Capitolul 2. Proiectarea aplicației

Având în vedere domeniul ales și faptul că s-a dorit ca aplicația să fie una disponibilă pe internet, s-a ales crearea aplicației în felul următor:

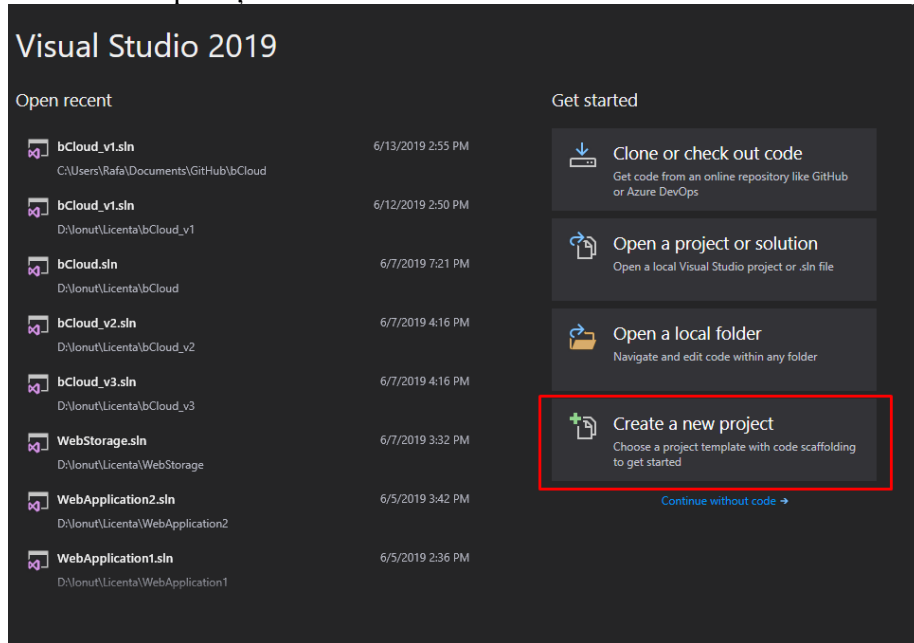


Figura 2.1: Crearea unui proiect nou în Visual Studio

- Din meniul apărut odată cu deschiderea mediului de dezvoltare Visual Studio s-a ales crearea unui proiect nou după cum se vede în Figura 2.1:

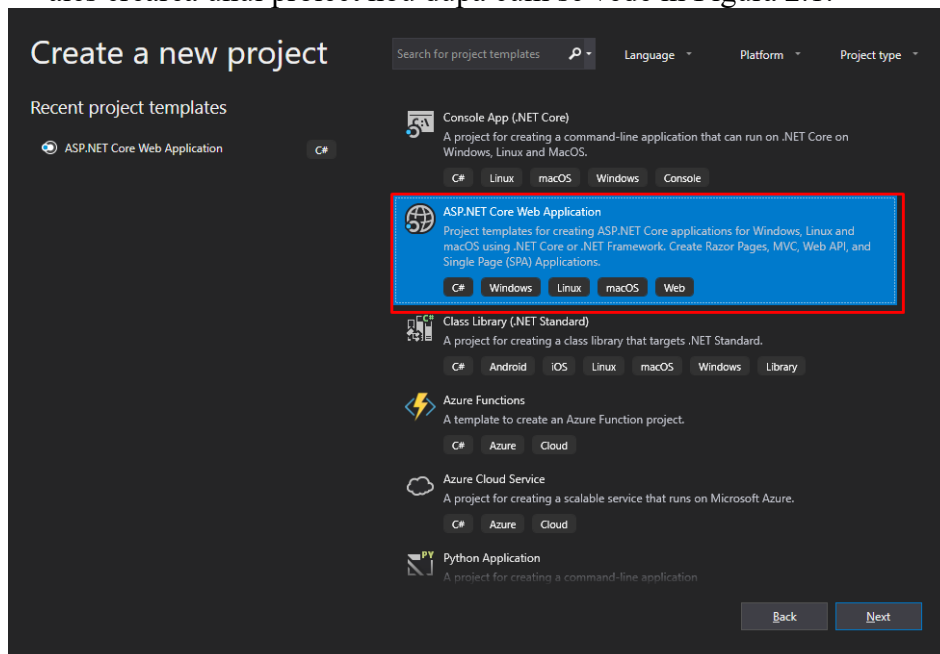


Figura 2.2: Selectarea tipul de proiect

- S-a selectat tipul proiectului ASP.NET Core Web Application (vezi Figura 2.2)

- După selectarea numelui și locului în care să se salveze proiectul, s-a ales frameworkul de lucru, și anume ASP.NET Core 2.0 (vezi chenarul albastru din Figura 2.3)

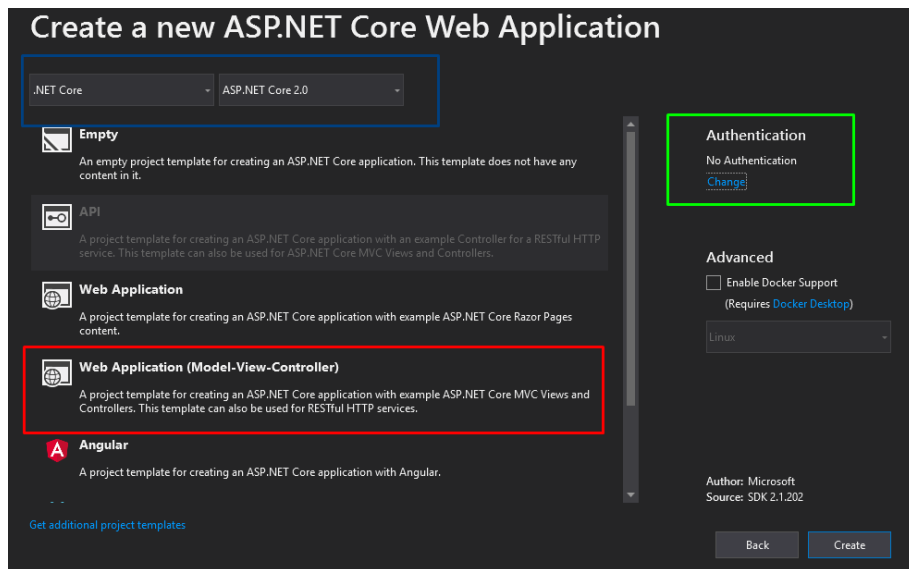


Figura 2.3: Setări aplicație

- S-a selectat modelul arhitectural al aplicației după cum se vede în cherul roșu din Figura 2.3
- S-a selectat tipul de autentificare a utilizatorilor din fereastra apărută după apăsarea butonului Change, din chenarul verde ( vezi Figura 2.3)

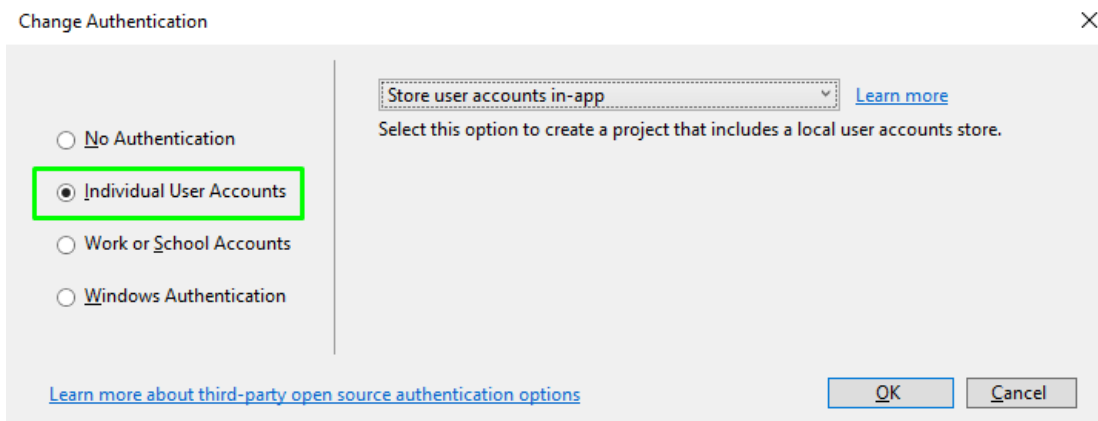


Figura 2.4: Setări autentificare

- Din fereastra apărută, s-a selectat autentificarea de tipul Individual User Accounts, după cum se poate observa în Figura 2.4, chenarul verde

După finalizarea acestor pași s-a început dezvoltarea aplicației în direcția domeniului ales.

## 2.1. Baza de date

În cadrul proiectului se folosește baza de date pentru a stoca informațiile despre utilizatori. Proiectarea bazei de date nu a fost dificilă, având în vedere că framework-ul ales oferă câteva funcționalități de bază în direcția asta. Astfel încât, atunci când s-a creat proiectul în Visual Studio și s-a ales tipul aplicației ce se dorește a fi dezvoltată, s-a ales și opțiunea ca aplicația să conțină de la început modelele specifice metodelor de înregistrare și logare ( vezi Figura 2.4 ).

Metodele care realizează atât crearea tabelor cât și funcțiile de înserare în aceste tabele au fost generate automat la crearea proiectului.

În aceste tabele sunt inserate și salvate informații despre fiecare utilizator care își creează un cont nou ( e-mail, parola ), astfel încât după ce a fost creat, utilizatorul să poată accesa contul utilizând datele folosite la înregistrare.

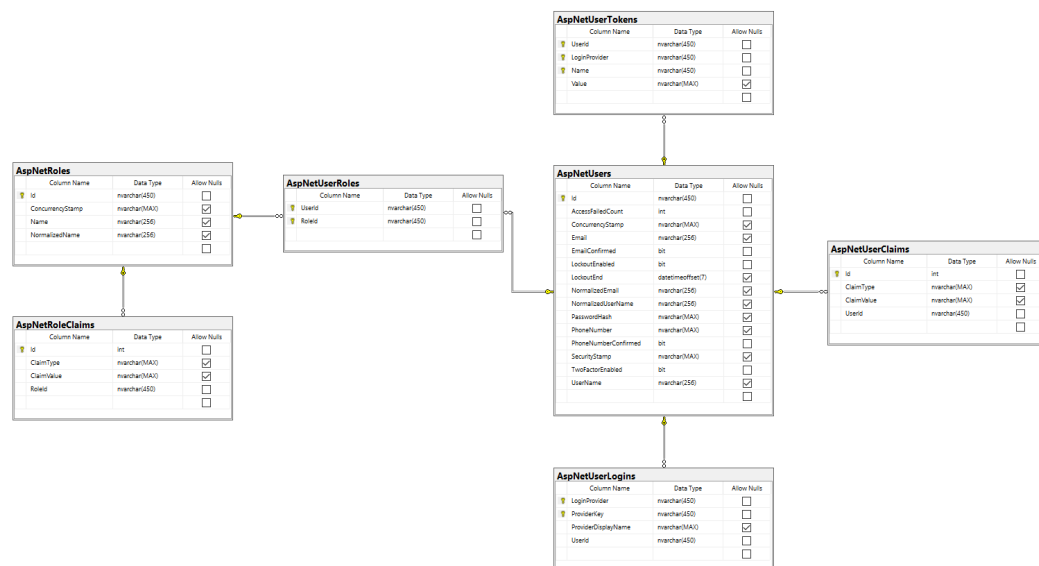


Figura 2.5: Diagrama bloc a bazei de date

În Figura 2.5 se pot vedea tabelele generate automat de către framework, în momentul în care a fost creat proiectul.

S-a decis ca această bază de date să fie amplasată pe un server local, deschis în Microsoft SQL Server, astfel încât a fost necesară o migrare a bazei de date către acest server. Această migrare a fost posibilă cu ajutorul comenzilor:

```
add-migration nume_migrare
update-database
```

De asemenea, pentru a face conexiunea cu serverul, s-a modificat fișierul `appsettings.json`, adăugând numele serverului la care se dorește să fie conectată aplicația. Acest nume a fost adăugat în linia `DefaultConnection`, la `Server`, după cum se poate observa în zona de cod ce urmează.

```
"DefaultConnection": "Server=DESKTOP-MSB9195;Database=aspnet-bCloud_v1-297C53C6-9228-47A1-8501-888DFFD803BF;Trusted_Connection=True;
```

Codul complet din acest fișier este disponibil în Anexa 1. .

După realizarea acestei conexiuni, a fost oferit acces la baza de date direct din programul SQL Server Management.

## 2.2. Aplicația de stocare a fișierelor

Având ca temă implementarea unei aplicații web ce oferă utilizatorilor posibilitatea de a stoca, accesa și descărca fișiere s-a început implementarea, pe rând, a fiecărei dintre aceste funcționalități. Având în vedere faptul că s-a ales implementarea aplicației utilizând modelul arhitectural MVC, s-a împărțit codul ce urma a fi dezvoltat în trei zone diferite, după cum urmează:

- Un controller
- Un model
- Un view

În cele ce urmează, vor fi descrise funcțiile implementate în fiecare dintre aceste fișiere. Metodele de încărcare, descărcare și stocare a fișierelor sunt open-source și sunt disponibile în articolul tehnic „Upload/Download Files în ASP.NET Core 2.0”, scris de către Tahir Naushad [21].

În primă fază, a fost necesară modificarea Controllerului Account, astfel încât, în momentul în care un utilizator nou se înregistrează, să se creeze automat și un director asociat lui. Acest director va fi unic și va purta în denumire adresa de email a utilizatorului.

```
DirectoryInfo di = new DirectoryInfo("wwwroot/upload/" );
di.CreateSubdirectory($"Dir{user}");
```

Zona de cod de mai sus a fost adăugată în partea de controller ce realizează înregistrarea unui utilizator nou. Funcția completă este disponibilă în Anexa 2. .

În prima linie din codul anterior se creează efectiv un director. Acesta va fi creat în după calea scrisă în ghilimele. În cazul acestei aplicații, toate directoarele se vor crea în directorul principal *upload*, creat special pentru a organiza mai bine fișierele încărcate de către fiecare utilizator. În a doua linie s-a specificat numele pe care să îl poarte fișierul nou creat. Astfel încât, toate directoarele create vor avea numele de forma : *Dir + email\_utilizator* ( de exemplu: *Dirionutb@gmail.com*).

După ce s-a dezvoltat zona de cod ce va crea directorul specific fiecărui utilizator, a fost necesară crearea unor modele ce vor reprezenta fișierele încărcate cu ajutorul controllerului.

```
public class FileInputModel
{
    public IFormFile FileToUpload { get; set; }
}
```

În codul prezentat anterior se accesează proprietățile fișierului ce urmează a fi uploadat cu ajutorul accesoriilor *get* și *set*.

În continuare a fost necesară crearea unui model ce va accesa numele și calea către fișierul încărcat. Tot aici s-a creat și o clasă numită *FilesViewModel.cs*. Aceasta reprezintă modul în care fișierele încărcate vor fi văzute de către utilizator. Aceasta metodă este open-source și nu



a fost modificată de către mine.[21]

```
public class FileDetails
{
    public string Name { get; set; }
    public string Path { get; set; }
}
public class FilesViewModel
{
    public List<FileDetails> Files { get; set; }
    = new List<FileDetails>();
}
```

Următorul pas în realizarea aplicației propuse a fost crearea unei funcții care să realizeze efectiv încărcarea fișierului.

```
public async Task<IActionResult> UploadFiles(List<IFormFile> files)
{
    System.Security.Claims.ClaimsPrincipal currentUser = this.User;

    var user = await _userManager.GetUserAsync(User);
    var email = user.Email;
    var dirName = $"Dir{email}";

    if (files == null || files.Count == 0)
        return Content("files not selected");

    foreach (var file in files)
    {
        var path = Path.Combine(
            Directory.GetCurrentDirectory(), $"wwwroot/upload/{dirName}/",
            file.GetFilename());

        using (var stream = new FileStream(path, FileMode.Create))
        {
            await file.CopyToAsync(stream);
        }
    }

    return RedirectToAction("Files");
}
```

Față de modul în care a fost dezvoltată funcția open-source, a fost necesară obținerea informațiilor despre utilizatorul conectat în momentul de față. Acest lucru a fost necesar pentru a afla numele directorului asociat acestui utilizator. După ce s-a obținut adresa de email a acestuia, s-a stabilit calea către directorul în care urmează a fi încărcate fișierele și anume calea către directorul utilizatorului curent. În cele ce urmează s-a verificat ce fișiere a selectat utilizatorul, s-a încărcat o copie a acestora și mai apoi s-au salvat în directorul corespunzător acestuia.

Următorul pas care s-a făcut a fost crearea unei noi acțiuni, în vederea afișării fișierelor încărcate de către utilizator în View-ul special creat pentru acest lucru. Această funcție este de asemenea open-source și nu a fost modificată[21].

```
public IActionResult Files()
{
    var model = new FilesViewModel();

    foreach (var item in
this.fileProvider.GetDirectoryContents($"Dir{currentUser.Identity}"))
    {
        model.Files.Add(
            new FileDetails { Name = item.Name, Path =
item.PhysicalPath });
    }
    return View(model);
}
```

Pentru a realiza acțiunea s-a declarat o variabilă ce a fost inițializată cu modelul dedicat vizualizării fișierelor, s-a parcurs fiecare fișier din directorul utilizatorului curent, s-a adăugat fiecare fișier în model și într-un final s-a returnat o pagină HTML în care sunt vizibile aceste fișiere.

Față de funcția open-source, în aceasta a fost adăugată și calea către directorul utilizatorului curent conectat.

Ulterior, s-a creat o funcție de download, astfel încât, fiecare utilizator să își poată descărca fișierele încărcate la un moment dat. Ca și în funcția de încărcare, a fost necesară stabilirea unei căi către directorul specific utilizatorului. Aceasta reprezintă și modificarea adusă față de modul în care a fost dezvoltată funcția open-source.[21]

```
var memory = new MemoryStream();
using (var stream = new FileStream(path, FileMode.Open))
{
    await stream.CopyToAsync(memory);
}
memory.Position = 0;
return File(memory, GetContentType(path), Path.GetFileName(path));
```

Odată stabilită calea către directorul ce conține fișierul, s-a creat o copie a fișierului din director ce se va salva pe dispozitivul utilizatorului.

Atât pentru a încărca, cât și pentru a descărca fișierele, este absolut necesar să se știe tipul acestora. În acest sens s-a creat o funcție care va returna tipul fișierului pe care a ales utilizatorul să îl încarce.

```
private string GetContentType(string path)
{
    var types = GetMimeTypes();
    var ext = Path.GetExtension(path).ToLowerInvariant();
    return types[ext];
}
```

De asemenea, s-a creat și un dicționar de extensii. Acest dicționar s-a utilizat pentru a stabili ce fișiere sunt acceptate de către aplicație. În cazul în care, după verificarea unui fișier, funcția de mai sus nu returnează un tip recunoscut de către aplicație (care se află în dicționarul creat) atunci utilizatorul nu va putea încărca acel fișier.

Această funcție nu a fost modificată față de cea open-source[21].

```
private Dictionary<string, string> GetMimeTypes()
{
    return new Dictionary<string, string>
    {
        {".txt", "text/plain"},
        {".pdf", "application/pdf"},
        {".doc", "application/vnd.ms-word"},
        {".docx", "application/vnd.ms-word"},
        {".xls", "application/vnd.ms-excel"},
        {".xlsx", "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"},
        {".png", "image/png"},
        {".jpg", "image/jpeg"},
        {".jpeg", "image/jpeg"},
        {".gif", "image/gif"},
        {".csv", "text/csv"}
    };
}
```

După cum se poate observa în codul de mai sus, aplicația acceptă o serie de fișiere de tip text( .txt, .pdf etc.), cât și fișiere de tip imagine ( .png,.jpg etc). Codul complet ce conține funcțiile de mai sus este disponibil în Anexa 3. , însă este disponibil și pe internet.[21]

După implementarea tuturor funcționalităților aplicației, s-a trecut la partea de interfață, unde s-a creat o pagină HTML nouă. În momentul în care utilizatorul v-a accesa această pagină, v-a putea vizualiza și descărca propriile fișiere.

```
@model bCloud_v1.Models.Home.FilesViewModel;

<center><h1>My files</h1></center>
<ul>
    @foreach (var item in Model.Files)
    {
        <li>
            <a asp-action="Download"
               asp-route-filename="@item.Name">
                @item.Name
            </a>
        </li>
    }
</ul>
```

Pentru a afișa aceste fișiere după modelul creat la început, a fost necesară injectarea modelului în această pagină. În continuare, s-au parcurs toate fișierele utilizatorului și s-au afișat sub forma unei liste, nu înainte de a aplica funcția de download fiecăruia dintre ele.

De asemenea, au fost necesare câteva modificări și asupra paginii principale. Inițial, pagina principală generată automat de către framework arăta la fel, fie că era un utilizator conectat, fie că nu. Astfel încât, s-a creat o verificare pentru a vedea dacă este sau nu un utilizator conectat. Această verificare a fost făcută în felul următor:

```
@if (SignInManager.IsSignedIn(User))
```

În cazul în care, în urma verificării, nu există nici un utilizator conectat, atunci pe pagina principală vor apărea cele doua butoane specifice: butonul de logare și butonul de înregistrare, pentru utilizatorii care accesează aplicația pentru prima oară. În cazul în care există un utilizator conectat, atunci pe pagina principală a acestuia vor apărea butoanele pentru selectarea și încărcarea fișierelor. Codul este disponibil în Anexa 4.

## Capitolul 3. Implementarea aplicației de stocare a fișierelor

### 3.1. Descrierea generală a implementării

În urma implementării tuturor funcțiilor, aplicația permite unui utilizator să își creeze un cont nou, să se conecteze cu datele folosite la înregistrare în cazul în care nu este prima oară când accesează aplicația, să încarce fișiere și să vizualizeze fișierele încărcate de către el(vezi Figura 3.1).

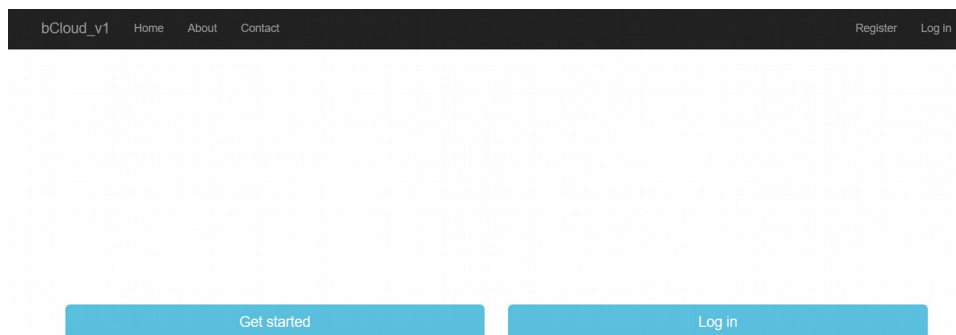


Figura 3.1: Intefată utilizatori neautentificați

### 3.2. Probleme întâmpinate și modalități de rezolvare

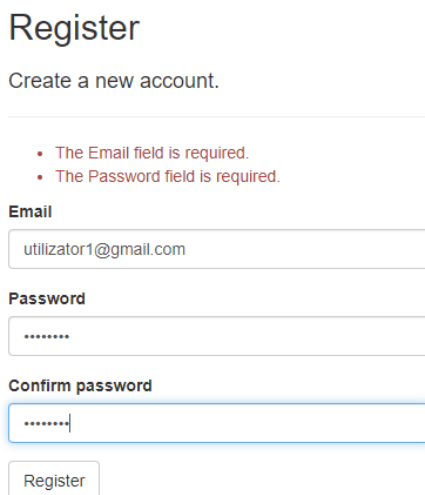
Una dintre cele mai mari probleme a fost reprezentată de modul în care se poate oferi acces unui utilizator doar către fișierele pe care le-a încărcat acesta. S-au încercat mai multe modalități de rezolvare a problemei precum asocierea id-ului utilizatorului fiecărui fișier încărcat de acesta sau returnarea unui view special în funcție de fiecare utilizator.

În cele din urmă, niciuna dintre aceste metode nu s-a dovedit a fi o soluție viabilă a problemei ce trebuia rezolvată. Pentru a rezolva această situație s-a ales modificarea funcției de înregistrare, astfel încât, în momentul în care utilizatorul se înregistrează i se creează automat și un director special ce conține în denumire numele folosit la crearea contului. În continuare, s-a folosit calea către acest director pentru a adăuga fișiere, pentru a le accesa însă și pentru a le descărca.

O altă problemă întâmpinată a fost reprezentată de accesarea informațiilor utilizatorului conectat. După cum s-a precizat și mai sus, atât pentru încărcare, cât și pentru vizualizare și descărcare a fost nevoie de calea către directorul asociat fiecărui utilizator. Această problemă a fost rezolvată utilizând funcția *Claims*, deja implementată în .NET CORE. Această funcție returnează informații precum: nume, email, id despre utilizatorul curent conectat în aplicație.

### 3.3. Funcționarea aplicației

În momentul în care un utilizator rulează aplicația pentru prima dată, acesta va fi nevoie să își creeze un cont nou. Tot ce trebuie să facă este să acceseze butonul *Get Started* disponibil pe prima pagină, sau butonul *Register* disponibil în bara de navigație.

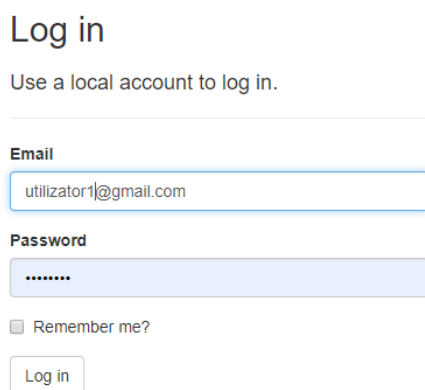


The screenshot shows a 'Register' form. At the top, it says 'Create a new account.' Below this, there are two red error messages: 'The Email field is required.' and 'The Password field is required.' The form has three input fields: 'Email' with the value 'utilizator1@gmail.com', 'Password' with masked characters '.....', and 'Confirm password' with masked characters '.....'. A 'Register' button is at the bottom.

Figura 3.2: Înregistrare utilizator nou

Acesta va fi redirecționat pe o pagină nouă, în care își poate introduce datele de autentificare, după cum se poate vedea în Figura 3.2. În cazul în care nu completează unul dintre câmpuri, utilizatorul va primi un mesaj de atenționare în acest sens. De asemenea, în cazul în care utilizatorul introduce o parolă care nu respectă cerințele de securitate sau parola introdusă în câmpul de confirmare nu corespunde cu cea aleasă mai sus, atunci acesta va primi un mesaj de atenționare și în acest sens.

În cazul în care utilizatorul are un cont deja creat, atunci acesta se poate autentifica utilizând butonul de *Log In* din pagina principală sau cel din bara de navigație.



The screenshot shows a 'Log in' form. It says 'Use a local account to log in.' Below this, there are two input fields: 'Email' with the value 'utilizator1@gmail.com' and 'Password' with masked characters '.....'. There is a checkbox labeled 'Remember me?' which is unchecked. A 'Log in' button is at the bottom.

Figura 3.3: Autentificare utilizator

În pagina apărută după accesarea butonului, utilizatorul își va putea introduce datele de autentificare pentru a putea accesa contul său din cadrul aplicației, după cum se poate vedea în Figura 3.3. În cazul în care datele de autentificare nu sunt recunoscute de către aplicație sau numele de utilizator nu corespunde cu parola introdusă, atunci utilizatorul va fi informat printr-un mesaj de atenționare, iar în cazul în care nu introduce datele corecte nu va avea acces la

funcționalitățile aplicației.

După efectuarea celor doi pași esențiali în accesarea contului, pe pagina principală, vizibilă utilizatorului vor fi disponibile funcționalitățile aplicației. Utilizând butonul *Choose files*, utilizatorului i se va deschide o fereastră de tip *explorer* pentru a naviga prin directoarele din computerul personal în vederea alegerii fișierelor pe care dorește să le încarce. (vezi Figura 3.4).

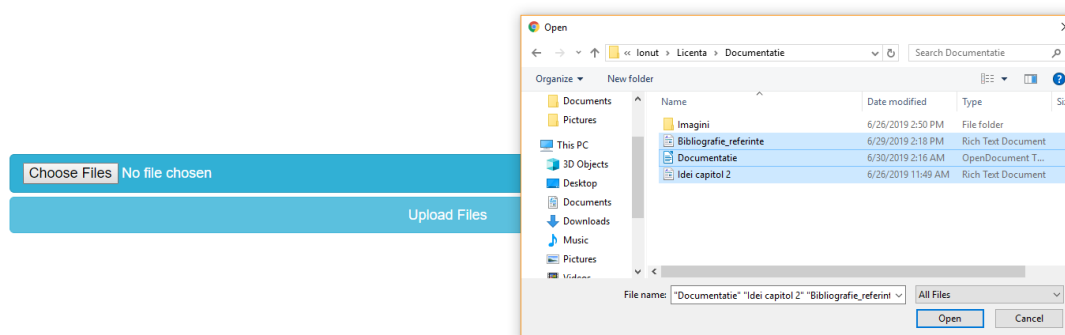


Figura 3.4: Selectare fișiere

După selectarea fișierelor pe care dorește să le încarce, utilizator va trebui să folosească butonul *Upload Files* pentru a realiza încărcarea efectivă a fișierelor în cloud. Odată cu încărcarea fișierelor, utilizatorul va fi redirecționat automat către o pagină specială în care va putea vizualiza fișierele pe care le-a încărcat, după cum se poate vedea în Figura 3.5.

### My files

- 1615200\_709753969055853\_335720135\_n.jpg
- 2011\_S1\_Nr. 1.doc
- 2019-01-13 (1).png
- 2019-01-13 (12).png
- 2019-01-13 (4).png
- Bibliografie\_referinte.rtf
- button\_create-new-project.png
- Captură ecran (126).png
- Documentatie.odt
- Idei capitol 2.rtf
- Laborator 13 - Grafuri 2.pdf
- Laborator11-Relatii4.pdf
- Laborator12-Grafuri.pdf
- Laborator13-Teorianumerelor.pdf
- Laborator3-Legileargumentatiei.pdf
- Laborator5-Multimi1.pdf
- Laborator6-Multimi2.pdf
- Laborator7-Functii.pdf
- P1040294.JPG
- P1040295.JPG
- P1040296.JPG
- Screenshot\_2.png
- \_DSC6817.jpg

Figura 3.5: Pagina dedicată fișierelor

De asemenea, în cazul în care utilizatorul nu dorește să mai încarce niciun fișier nou, ci doar să descarce unul încărcat anterior, atunci acesta poate accesa direct pagina *Files* utilizând cu butonul cu același nume disponibil în bara de navigație.

Pentru a realiza descărcarea efectivă a fișierului, utilizatorul nu trebuie decât să îl caute în

lista afișată pe pagină și să apese o singură dată pe el. În momentul respectiv va fi apelată funcția de descărcare, o copie a fișierului fiind disponibilă apoi în dispozitivul utilizatorului.

În cazul în care utilizatorul nu mai dorește să folosească aplicația pentru moment, acesta poate accesa butonul *Log Out*, disponibil în bara de navigație.

De asemenea, exista posibilitatea ca utilizatorul să dorească modificarea datelor de autentificare. Acest lucru este posibil apăsând direct pe numele lui, în bara de navigație. După accesare, utilizatorul va fi redirecționat către o pagină în care poate face aceste modificări.

The screenshot shows a web interface titled "Manage your account" with the subtitle "Change your account settings". On the left, there is a vertical menu with three options: "Profile", "Password" (which is highlighted with a blue background), and "Two-factor authentication". On the right, under the heading "Change password", there are three input fields labeled "Current password", "New password", and "Confirm new password". Below these fields is a button labeled "Update password".

Figura 3.6: Modificare date de autentificare

După cum se poate vedea în Figura 3.6, utilizatorul își poate modifica parola sau unele date din profilul său.



## Capitolul 4. Testarea aplicației

În momentul de față, aplicația se află pe un server local. Astfel încât, pentru rularea acesteia este necesară executarea codului din mediul de dezvoltare *Visual Studio*. Pentru a fi disponibilă pe internet, aplicației îi trebuie un domeniu și un serviciu de găzduire.

Pentru punerea în funcțiune a acestei aplicații nu este necesară configurarea sau instalarea altor elemente suplimentare.

Pentru a testa funcționarea corectă a aplicației, s-au creat mai multe scenarii de test.

### 4.1. Testarea realizării directoarelor individuale

În vederea testării acestei funcții, s-au creat o serie de conturi de test și s-au încărcat câteva fișiere de pe fiecare dintre aceste conturi.

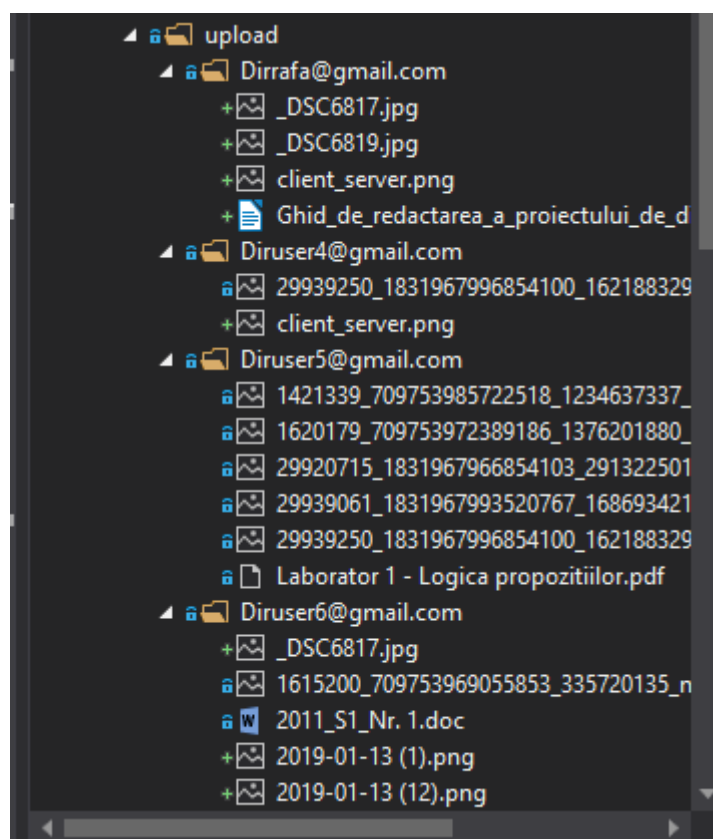


Figura 4.1: Directoare utilizatori

După cum se poate observa în Figura 4.1, s-a creat câte un director special pentru fiecare cont creat. De asemenea, fiecare director conține doar fișierele adăugate de pe contul utilizatorului de care aparține.

Tot aici s-a verificat și corectitudinea denumirii conturilor. Astfel încât, fiecare director creat o dată cu crearea contului respectă tiparul impus (șirul de caractere „Dir”, urmat de adresa de email folosită de către utilizator la crearea contului).

## 4.2. Testarea afișării individuale a fișierelor încărcate de fiecare utilizator

S-a realizat acest test cu scopul de a verifica dacă fiecare utilizator își poate accesa fișierele încărcate. Tot în cadrul acestui test s-a verificat dacă fiecare utilizator are acces doar la fișierele lui, fără a avea acces la fișierele încărcate de către alți utilizatori.

Pentru a realiza acest lucru, s-au creat câteva conturi de test și s-au încărcat câteva fișiere de pe fiecare dintre aceste conturi.

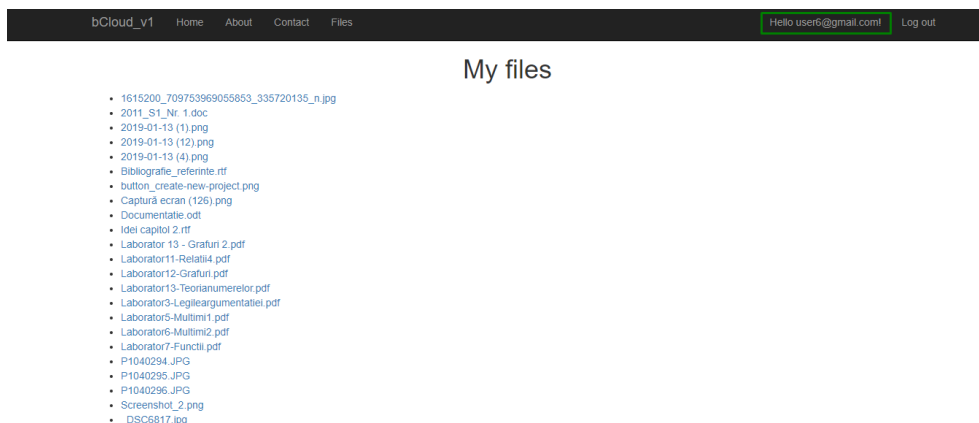


Figura 4.2: Fișiere utilizator test 1

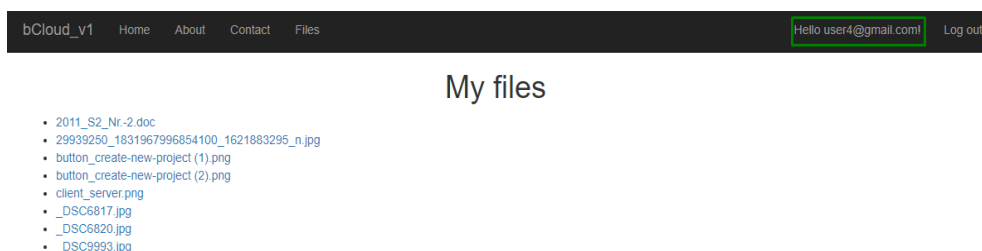


Figura 4.3: Fișiere utilizator test 2

În Figura 4.2 și Figura 4.3, în chenarul verde, se poate observa că testul a fost făcut pe două conturi diferite. De asemenea, se observă cum fiecare dintre acești doi utilizatori au acces doar la fișierele pe care le-au încărcat.

Având în vedere că acest test a fost un succes, se poate spune că aplicația oferă un grad ridicat de securitate în ceea ce privește datele personale ale fiecărui utilizator ce dorește să beneficieze de avantajele oferite de această aplicație.

### 4.3. Testarea încărcării mai multor fișiere în același timp

Opțiunea de a adăuga mai multe fișiere în același timp reprezintă un avantaj pentru fiecare utilizator, fiind o soluție de economisire a timpului necesar acestui proces.

În contextul acestei verificări, a fost necesară conectarea la un cont de test și s-au selectat un număr de fișiere pentru încărcare, după cum se poate vedea în Figura 4.4.

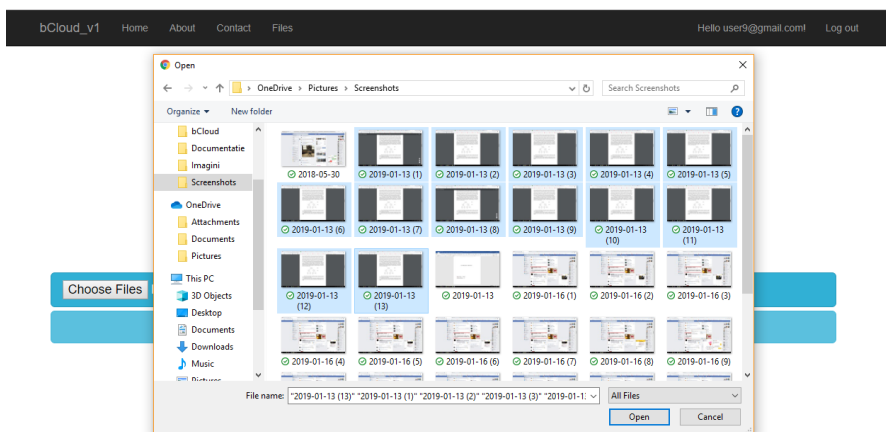


Figura 4.4: Selectare fișiere multiple

După selectarea fișierelor din fereastra de explorare a computerului personal, s-a verificat dacă acestea au fost selectate și în aplicație. (Vezi Figura 4.5)

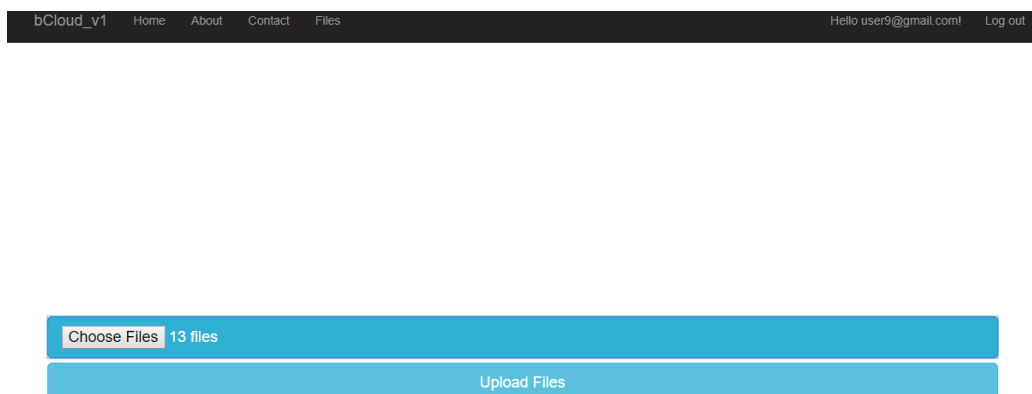


Figura 4.5: Verificare număr fișiere în aplicație

Ultimul pas în realizarea acestui test a fost verificarea paginii de vizualizare a fișierelor, pentru a vedea dacă acestea au fost încărcate.

În Figura 4.6 se poate observa că aplicația a încărcat cu succes toate fișierele selectate.

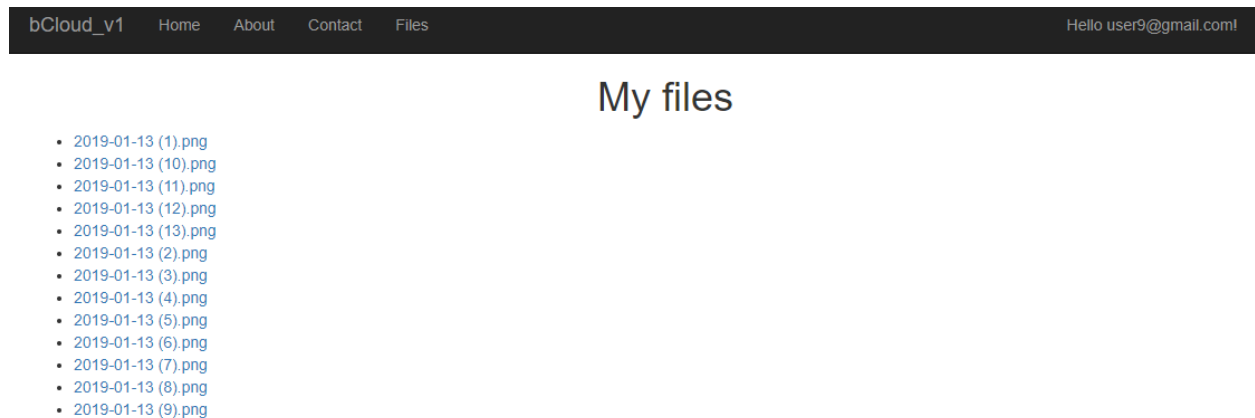


Figura 4.6: Pagină vizualizare fișiere multiple

## Concluzii

Având în vedere obiectivul propus atunci când s-a ales această temă în vederea realizării lucrării de licență și anume dezvoltarea unei aplicații web care să permită utilizatorilor să încarce, să stocheze și să descarce fișiere, se poate spune că rezultatul final este unul satisfăcător.

Inițial, s-a propus crearea unei aplicații simple, care să permită încărcarea și descărcarea unor fișiere. Odată realizat acest obiectiv, s-a trecut la implementarea funcționalităților suplimentare, precum asocierea fiecărui cont cu un anumit director și afișarea în pagină doar a fișierelor încărcate de utilizatorul curent.

De un foarte mare ajutor a fost alegerea făcută în sensul tehnologiei care s-a utilizat pentru a dezvolta această aplicație. Framework-ul folosit a oferit un punct de plecare solid în ceea ce privește baza de date în care se stochează datele de autentificare ale utilizatorilor, cât și un schelet al interfeței grafice.

În momentul de față, există numeroase aplicații ce oferă servicii de stocare a fișierelor. Privind ce funcționalități oferă acele servicii, se poate spune că aplicația dezvoltată este una simplă și reprezintă un punct de plecare, peste care se mai pot implementa multe alte funcționalități.

Un plus al aceste aplicații față de actualele servicii de stocare a fișierelor este reprezentat de dimensiunea spațiului de stocare oferit gratuit. Un alt avantaj este reprezentat de interfața simplistă și ușor de utilizat.

Ca și dezvoltarea ulterioară, se poate implementa un sistem prin care utilizatorii își pot partaja fișierele între ei, oferind acces asupra acestora prin intermediul unui link. O altă funcționalitate ce poate fi implementată pe viitor constă în deschiderea anumitor tipuri de fișiere direct în aplicație, în vederea vizualizării conținutului și de ce nu, a modificării acestora, fără ca utilizatorii să mai fie nevoiți să le descarce în computerul personal.

## Bibliografie

- [1] Bill Stewart, The Living Internet, 2000
- [2] Keith D. Foote, A brief history of Cloud Computing, 2017, Disponibil la:  
<https://www.dataversity.net/brief-history-cloud-computing/>
- [3] Arif Mohamed, A history of Cloud Computing, 2019 Disponibil la:  
<https://www.computerweekly.com/feature/A-history-of-cloud-computing>
- [4] Krissi Danielson, Distinguishing Cloud Computing from Utility Computing, 2008, Disponibil la:  
[http://www.ebizq.net/blogs/saasweek/2008/03/distinguishing\\_cloud\\_computing/](http://www.ebizq.net/blogs/saasweek/2008/03/distinguishing_cloud_computing/)
- [5] Ujwala, 10 Benefits Of Data Storage In A Cloud, 2019 Disponibil la:  
<https://www.milesweb.com/blog/hosting/cloud/10-benefits-data-storage-cloud/>
- [6] IBM, The DeepQA Research Team, 2019 Disponibil la:  
[https://researcher.watson.ibm.com/researcher/view\\_group.php?id=2099](https://researcher.watson.ibm.com/researcher/view_group.php?id=2099)
- [7] Keith D. Foote, Cloud Computing: Latest Trends, Issues, and Innovations, 2019 Disponibil la :  
<https://www.dataversity.net/cloud-computing-latest-trends-issues-innovations/>
- [8] Darko Angelovski, The Main Cloud Storage Market Challenges — That Sparked Innovation, 2019  
Disponibil la: <https://medium.com/iagon-official/cloud-storage-market-challenges-c3ac0459925f>
- [9] Mehvish, How Does Google Backup and Sync Work: A Comprehensive Guide, 2019, Disponibil la:  
<https://www.guidingtech.com/google-backup-sync-guide/>
- [10] Jonathan Strickland , How Google Docs Works , 2008, Disponibil la:  
<https://computer.howstuffworks.com/internet/basics/google-docs5.htm>
- [11] Dropbox, Can I upload large files to Dropbox?, 2019 Disponibil la:  
<https://help.dropbox.com/installs-integrations/sync-uploads/upload-limitations>
- [12] Todd Hoff, 6 Lessons from Dropbox - One Million Files Saved Every 15 minutes , 2011, Disponibil  
la: <http://highscalability.com/blog/2011/3/14/6-lessons-from-dropbox-one-million-files-saved-every-15-minu.html>
- [13] Dan Wheeler, Ziga Mahkovec, Chris Varenhorst, Dropbox dives into CoffeeScript, 2012, Disponibil  
la: <https://blogs.dropbox.com/tech/2012/09/dropbox-dives-into-coffeescript/>
- [14] Wired, The Epic Story of Dropbox's Exodus From the Amazon Cloud Empire, 2016, Disponibil la:  
<https://www.wired.com/2016/03/epic-story-dropboxs-exodus-amazon-cloud-empire/>
- [15] Dropbox, Security, 2019 Disponibil la:  
[https://www.dropbox.com/en/security?\\_locale\\_specific=en](https://www.dropbox.com/en/security?_locale_specific=en)
- [16] Microsoft, Visual Studio Details, 2019 Disponibil la:  
<https://www.microsoft.com/en-us/download/details.aspx?id=55984>
- [17] Microsoft, What is SQL Server Management Studio (SSMS)?, 2019 Disponibil la:  
<https://docs.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-2017>
- [18] Deborah Lee Soltesz, What Does JavaScript Do?, 2019 Disponibil la:  
<https://www.techwalla.com/articles/what-does-javascript-do>
- [19] Meyer Eric , Cascading Style Sheets 2.0 Programmer's Reference, 2001
- [20] World Wide Web Consortium, First mention of HTML Tags on www-talk mailing list, 1991
- [21] Tahir Naushad, Upload/Download Files in ASP.NET Core, 2017 Disponibil la:  
<https://www.codeproject.com/Articles/1203408/Upload-Download-Files-in-ASP-NET-Core>

- [22] Network Cloud Solutions, 2019 Disponibil la :  
<http://www.rock-cafe.info/suggest/network-cloud-solutions-6e6574776f726b.html>
- [23] Cloud Computing, 2019 Disponibil la:  
[https://ro.wikipedia.org/wiki/Cloud\\_computing](https://ro.wikipedia.org/wiki/Cloud_computing)
- [24] Iagon, Revolutionizing the Cloud, 2019 Disponibil la:  
<https://busy.org/@oivas/iagon-revolutionizing-the-cloud>
- [25] A complete rewrite of ASP.NET, 2017, Disponibil la:  
<https://www.danielcrabtree.com/blog/333/asp-net-core-a-complete-rewrite-of-asp-net>
- [26] Everything you need to know Before Starting to work with JavaScript & SEO, 2019 Disponibil la: <https://www.amfastech.com/2018/08/everything-about-javascript-seo-google.html>

## Anexe

### *Anexa 1. Stabilire conexiune cu baza de date*

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=DESKTOP-MSB9195;Database=aspnet-bCloud_v1-297C53C6-9228-47A1-8501-888DFFD803BF;Trusted_Connection=True;MultipleActiveResultSets=true"
  },
  "Logging": {
    "IncludeScopes": false,
    "LogLevel": {
      "Default": "Warning"
    }
  }
}
```



## *Anexa 2. Creare directoare individuale*

```
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Register(RegisterViewModel model, string
returnUrl = null)
{
    ViewData["ReturnUrl"] = returnUrl;
    if (ModelState.IsValid)
    {
        var user = new ApplicationUser { UserName = model.Email, Email = model.Email };
        var result = await _userManager.CreateAsync(user, model.Password);
        if (result.Succeeded)
        {
            DirectoryInfo di = new DirectoryInfo("wwwroot/upload/" );
            di.CreateSubdirectory($"Dir{user}");
            _logger.LogInformation("User created a new account with password.");
            var code = await _userManager.GenerateEmailConfirmationTokenAsync(user);
            var callbackUrl = Url.EmailConfirmationLink(user.Id, code, Request.Scheme);
            await _emailSender.SendEmailConfirmationAsync(model.Email, callbackUrl);
            await _signInManager.SignInAsync(user, isPersistent: false);
            _logger.LogInformation("User created a new account with password.");
            return RedirectToLocal(returnUrl);
        }
        AddErrors(result);
    }

    return View(model);
}
```

*Anexa 3. Funcționalități*

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net.Http.Headers;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.FileProviders;
using bCloud_v1.Models;
using bCloud_v1.Models.Home;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNet.Identity;
using static Microsoft.EntityFrameworkCore.DbLoggerCategory;
using Microsoft.EntityFrameworkCore.Metadata.Internal;
using bCloud_v1.Models.AccountViewModels;
using System.Security.Claims;

namespace bCloud_v1.Controllers
{
    public class uploadmultipleController : Controller
    {
        private readonly IFileProvider fileProvider;
        private readonly Microsoft.AspNetCore.Identity.UserManager<ApplicationUser>
        _userManager;

        //class constructor

        public uploadmultipleController(IFileProvider fileProvider,
        Microsoft.AspNetCore.Identity.UserManager<ApplicationUser> userManager)
        {
            this.fileProvider = fileProvider;
            _userManager = userManager;
        }

        [HttpPost]

        [HttpPost]
        public async Task<IActionResult> UploadFiles(List<IFormFile> files)
        {
            System.Security.Claims.ClaimsPrincipal currentUser = this.User;

            var user = await _userManager.GetUserAsync(User);
            var email = user.Email;
            var dirName = $"Dir{email}";

            if (files == null || files.Count == 0)
                return Content("files not selected");
        }
    }
}

```

```
        foreach (var file in files)
        {
            var path = Path.Combine(
                Directory.GetCurrentDirectory(), $"wwwroot/upload/{dirName}/",
                file.GetFilename());

            using (var stream = new FileStream(path, FileMode.Create))
            {
                await file.CopyToAsync(stream);
            }
        }

        return RedirectToAction("Files");
    }

    public IActionResult Files()
    {
        System.Security.Claims.ClaimsPrincipal currentUser = this.User;

        var model = new FilesViewModel();

        foreach (var item in
            this.fileProvider.GetDirectoryContents($"Dir{currentUser.Identity.Name}"))
        {
            model.Files.Add(
                new FileDetails { Name = item.Name, Path = item.PhysicalPath });
        }
        return View(model);
    }

    public async Task<IActionResult> Download(string filename)
    {
        System.Security.Claims.ClaimsPrincipal currentUser = this.User;

        var user = await _userManager.GetUserAsync(User);
        var email = user.Email;
        var dirName = $"Dir{email}";
        if (filename == null)
            return Content("filename not present");

        var path = Path.Combine(
            Directory.GetCurrentDirectory(),
            $"wwwroot/upload/{dirName}/", filename);

        var memory = new MemoryStream();
        using (var stream = new FileStream(path, FileMode.Open))
        {
            await stream.CopyToAsync(memory);
        }
        memory.Position = 0;
        return File(memory, GetContentType(path), Path.GetFileName(path));
    }
```

```

    }

    private string GetContentType(string path)
    {
        var types = GetMimeTypes();
        var ext = Path.GetExtension(path).ToLowerInvariant();
        return types[ext];
    }

    private Dictionary<string, string> GetMimeTypes()
    {
        return new Dictionary<string, string>
        {
            { ".txt", "text/plain" },
            { ".pdf", "application/pdf" },
            { ".doc", "application/vnd.ms-word" },
            { ".docx", "application/vnd.ms-word" },
            { ".xls", "application/vnd.ms-excel" },
            { ".xlsx", "application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet" },
            { ".png", "image/png" },
            { ".jpg", "image/jpeg" },
            { ".jpeg", "image/jpeg" },
            { ".gif", "image/gif" },
            { ".csv", "text/csv" }
        };
    }
}

```

### Anexa 4. Interfață

```

@using Microsoft.AspNetCore.Identity
@using bCloud_v1.Models

@inject SignInManager<ApplicationUser> SignInManager
@inject UserManager<ApplicationUser> UserManager

<link rel="stylesheet" href="~/css/site.css" />

@if (SignInManager.IsSignedIn(User))
{
    <form asp-controller="uploadmultiple" asp-action="UploadFiles" method="post"
        enctype="multipart/form-data">
        <div class="upload_container">
            <div class="col-lg-12 ">
                <input class="btn btn-info btn-lg btn-block" type="file"
name="files" multiple />
                <button class="btn btn-info btn-lg btn-block" type="submit">Upload
Files</button>
            </div>
        </div>
    </form>
}
else
{
    <div class="container-fluid ">
        <div class="upload_container ">
            <div class="col-lg-12 ">
                <form asp-area="" asp-controller="Account" asp-action="Register">
                    <button class="btn btn-info btn-lg btn-block">Get
started</button>
                </form>
            </div>
            <div class="col-lg-12 ">
                <form asp-area="" asp-controller="Account" asp-action="Login">
                    <button class="btn btn-info btn-lg btn-block">Log in</button>
                </form>
            </div>
        </div>
    </div>
}

```