UNIVERSITATEA “LUCIAN BLAGA” DIN SIBIU
FACULTATEA DE INGINERIE
DEPARTAMENTUL DE CALCULATOARE ŞI INGINERIE ELECTRICĂ

# PROIECT DE DIPLOMĂ

Coordonator științific: Prof. Univ. Dr. Ing. REMUS BRAD

Îndrumător: Asis. Univ. Drd. Ing. ALEXANDRU DOROBANȚIU

Absolvent: Lupu Andreea-Monica

Specializare: Calculatoare

- SIBIU 2020 -

**Cuprins**

# 1. PREZENTAREA TEMEI

## 1.1. Introducere

Recomăndrile sunt esențiale în zilele noastre, deoarece ajută utilizatorul să piardă cât mai puțin timp în a găsi ce își dorește. Deoarece timpul este limitat, oamenii caută orice variantă care îi poate ajuta să facă o decizie cât mai rapidă. Un sistem de recomandări ajută la construirea profilului unui utilizator, pe baza unor aprecieri făcute anterior.

Am ales aceasta temă deoarece eu pictez în timpul liber, iar atunci când oamenii doresc să facă o comandă pentru o pictură, de foarte multe ori, nu știu ce își doresc. Așteaptă propuneri, idei, variante din care pot alege. Fiecare pictură este unică. Chiar dacă aș încerca să fac aceeași pictură de 2 ori, nu ar ieși la fel. Având o istorie pe care sa mă pot baza, l-aș putea ajuta pe cel care dorește să comande o pictură, să primească una pe care și-o dorește și care îi place.

Sistemul de recomandări pe care am ales să-l implementez se bazează pe atributele asociate picturilor. Fiecare pictură are asociat un număr de atribute. Acestea sunt alese astfel încât să descrie cât mai bine pictura. Cu cât mai bine sunt alese atributele, cu atât mai precise vor fi recomandările ulterioare. Dacă atributele sunt alese greșit, evident că recomandările vor fi eronate, deci inutile.

Fără un sistem de recomandări, utilizatorul ar pierde mai mult timp până ar găsi ceva ce îi place. Mai mult decât atât, și-ar pute pierde interesul de a vizualiza sau cumpăra un produs. Acest lucru poate duce la scăderea numărului de utilizatori sau scăderea vânzărilor.

## 1.2. Descrierea temei

Sistemele de recomandare încearcă să ghicească interesele utilizatorilor și să recomande produse. Sunt cele mai puternice sisteme bazate pe învățare automată pe care distribuitorii le implementează, pentru a crește vânzările. Datele necesare pentru a genera recomandări provin din aprecierile date de utilizatori. Aceste aprecieri rezultă din interogările motoarelor de căutare, din achizițiile precedente sau alte cunoștințe despre utilizator.

**La ce folosesc sistemele de recomandări?**

Seth Godin (unul dintre cei mai apreciați autori de cărți de afaceri și orator) spunea în cartea sa „Toți marketerii sunt mincinoși" : „Confruntați cu mulțimea de informații, ne luptăm din răsputeri pană găsim o teorie despre ceea ce se întâmplă. Umplem spațiile libere și ghicim ceea ce vedem. De îndată ce suntem mulțumiți că predicția este destul de bună, ne putem relaxa."

Sistemul „umple spațiile libere" cu recomandări. Recomandările accelerează procesul de căutare și ajută utilizatorii să găsească ce îi interesează, fără a depune mult efort căutând. Astfel, utilizatorul se simte înțeles și probabilitatea ca acesta să cumpere produse cu conținut asemănător, crește. Prin urmare, vânzările pot crește cu ajutorul unui sistem de recomandări.

## 1.3. Scop și obiective

Aplicația are ca scop generarea unor picturi similare cu picturile apreciate anterior de un anumit utilizator. Pentru a face acest lucru posibil, a fost nevoie de o aplicație de tip client-server care constă în: o interfață cu utilizatorul (clientul) și o aplicație care gestionează datele (serverul).

Clientul trebuie să îndeplinească următoarele sarcini:

- permiterea fiecărui utilizator să se conecteze cu un cont unic sau să creeze un cont nou;
- afișarea picturilor corespunzătoare categoriei din care fac parte;
- permiterea utilizatorilor să vizualizeze evenimentele care au avut loc;
- afișarea recomandărilor;
- comunicare cu serverul.

Serverul trebuie să îndeplinească următoarele sarcini:

- gestionarea datelor din și în baza de date;
- recepția și transmiterea datelor către interfața cu utilizatorul;
- rularea aplicației care generează recomandările;
- comunicare cu clientul.

### 1.4. Tehnologii utilizate

- .NetCore (3.0.100) – limbajul de programare C#: cu ajutorul acestui limbaj se vor gestiona datele din și în baza de date;
- limbajul de programare Python (3.6.7) – cu ajutorul acestui limbaj se vor genera recomandările;
- Angular (CLI 9.0.7): cu ajutorul acestei tehnologii se va construi interfața cu utilizatorul

## 2. TEORIE

### 2.1. ASP.Net

Tehnologia Microsoft ASP.NET este folosită pentru a crea aplicații și servicii web. Această tehnologie este succesorul lui ASP *(„Active Server Pages”*), care beneficiază de puterea platformei de dezvoltare .NET, dar și de setul de instrumente oferite de mediul de dezvoltare al aplicației „Visual Studio .Net”

Avantaje ASP.NET :

- Conține un set mare de componente, care se bazează pe XML (acestea oferă un model de programare orientată pe obiect).
- Deoarece ASP.NET rulează cod compilat, performanța aplicației web crește.
- Există posibilitatea de a separa codul sursă în 2 fișiere: unul pentru executabil, unul pentru conținutul paginii (cod HTML și textul paginii).
- .NET este compatibil cu mai mult de 20 de limbaje.

Platforma .NET

.NET este o platformă de dezvoltare formată din instrumente, limbaje de programare și librării pentru posibilitatea formării aplicațiilor.

Platforma de bază oferă componente care se aplică mai multor aplicații distincte între ele.

ASP.NET extinde platforma .NET cu componente pentru construirea anumitor aplicații.

Ce include platforma .NET:

- Limbajele de programare: C#, F# și Visual Basic.
- Librarii de bază (de exemplu, librării pentru a putea lucra cu șiruri, date, fișiere I/O ...).
- Editoare și instrumente pentru Windows, Linux, macOS, Doker.

Ce adaugă ASP.NET platformei .NET:

- Framework de bază pentru procesarea cererilor în C# sau F#.
- Sintaxă pentru template-ul paginilor web (cunoscută și sub numele de Razor), pentru a construi pagini web dinamice, folosind C#.
- Biblioteci pentru modele web comune, cum ar fi MVC (*Model View Controller*).
- Sistem de autentificare care include librarii, bază de date, șabloane pentru gestionarea autentificărilor (inclusiv autentificarea externă cu Google și altele).
- Extensii pentru editor care oferă evidențierea sintaxei, completarea codului și alte funcționalități specifice dezvoltării paginilor web.

Cod pentru back-end

Atunci când ASP.NET este utilizat ca și cod pentru back-end, este scris utilizând C#, F# sau Visual Basic.

Deoarece ASP.NET extinde .NET, se pot utiliza pachete și biblioteci disponibile tuturor dezvoltatorilor .NET. Este posibilă autorizarea propriilor biblioteci care sunt partajate între orice aplicație scrisă pe platforma .NET.

Pagini dinamice care utilizează C#, HTML, CSS și JavaScript

Razor oferă o sintaxă pentru a crea pagini web dinamice, utilizând HTML și C#. Codul C# este evaluat în server și rezultatul HTML este trimis utilizatorului.

## 2.2. ASP .NetCore

ASP.NET Core este o versiune a platformei ASP.NET. Este mai performantă și mai rapidă decât celelalte framework-uri web în benchmark-urile TechEmpower independente.

ASP.NET Core are ca scop permiterea componentelor în timpul rulării (componentelor API, compilatoarelor și limbajelor) să evolueze rapid, cât timp oferă o platformă stabilă pentru a păstra rularea aplicațiilor.

Multiple versiuni ale ASP.NET Core pot exista una lângă alta în același server (asta înseamnă că o aplicație poate prelua ultima versiune, cât timp altă aplicație poate continua să ruleze pe versiunea pe care a fost testată).

ASP.NET Core oferă diverse opțiuni de asistență pentru ciclul de viață pentru a răspunde nevoilor unei aplicației. Se poate alege o versiune de asistență pe termen lung sau se poate rula cu cea mai recentă versiune.

Procesul de rulare ASP.NET Core pe care rulează o aplicație poate fi implementat ca parte a acestei aplicații sau instalat central pe serverul web.

## 2.3. Limbajul de programare C#

C# este un limbaj de programare orientat pe obiect. Acest limbaj a fost dezvoltat în jurul anilor 2000 de către Microsoft ca parte a inovației .NET. Ulterior a fost aprobat de către ECMA (*„European Computer Manufacturers Association”*) ca standard internațional.

Numele „C#” a fost inspirat din muzică, unde o notă însoțită de „#” indică faptul că acea notă ar trebui să fie cu un semiton mai ridicată, decât notele care nu sunt însoțite de acest semn.

## 2.4. Platforma Angular

Angular este o platformă și cadru pentru construirea aplicațiilor de tip client, care conțin o singură pagină. Aceasta utilizează HTML și se bazează pe TypeScript.

Arhitectura aplicației

Arhitectura aplicației este bazată pe câteva concepte de bază. Blocurile de bază sunt reprezentate de module și poartă numele de „NgModules”. Acestea oferă un context de compilare pentru componente. Aceste blocuri colectează codul aferent în seturi funcționale.

O aplicație Angular este alcătuită de un set de module. O aplicație are întotdeauna cel puțin un modul rădăcină care permite bootstrap-ul și este alcătuit din multe funcții.

Componentele definesc modul de vizualizare. Acestea sunt seturi de elemente de ecran.

Totodată, componentele folosesc și servicii. Acestea oferă funcționalități care nu sunt legate direct de modul de vizualizare.

Un modul declară un context de compilare pentru un set de componente dedicat unui domeniu de aplicație sau unui flux de lucru. Un modul poate importa funcții din alte module și poate fii exportat în alte module.

Ce conține orice aplicație Angular?

Fiecare aplicație Angular are un modul rădăcină care se numește „AppModule”. Acesta furnizează mecanismul de bootstrap de la care pornește aplicația.

Fiecare aplicație Angular este formata din cel puțin o componentă. Fiecare componentă definește o clasa căreia îi este asociată un șablon HTML. Acest șablon definește modul de vizualizare.

Partajarea datelor

Atunci când este nevoie ca o dată să fie partajată pentru a putea fi citită sau modificată de mai multe componente, se creează o clasa de servicii. Definirea clasei de servicii este însoțită de decoratorul „@Injectable()”. Acest decorator are ca scop permiterii injectării altor furnizori ca dependențe în clasă.

Redirecționarea

Angular Router NgModule pune la dispoziție un serviciu prin care să poate defini o cale de navigare între diferire stări ale aplicației.

Router-ul folosește căi asemănătoare cu cele URL, dar redirecționarea se face către moduri de vizualizare nu către pagini. Atunci când este făcută o redirecționare, routerul are sarcina de a afișa sau ascunde anumite componente.

## 2.5. Limbajul de programare Python

Python este un limbaj de programare de nivel înalt. Acesta este un limbaj curat și conține o sintaxă care permite utilizatorilor să își exprime ideile într-un mod mai simplu și mai clar decât în alte limbaje de programare.

Specific acestui limbaj este lizibilitatea codului prin utilizarea spatiilor goale. Construcția sa de limbaj și abordarea orientata pe obiect accentuează ușurința programatorilor de a scrie cod.

## 2.6. Keras vs Tensorflow vs Pytorch

Învățarea automată este un set de algoritmi care procesează date, învață din acestea și aplică ce au învățat, pentru a face decizii inteligente. Scopul programelor bazate pe învățare automată este de a găsi cea mai eficientă cale spre un rezultat dorit, prin utilizarea încercărilor repetate și prin verificarea rezultat obținut cu cel dorit.

Învățarea profundă este o varianta mai avansată a procesului de învățare automată . Scopul este analizarea continuă a datelor, pe baza unei structuri logice, asemănătoare cu modul de funcționare a conștiinței umane. Învățarea profundă folosește o rețea stratificată de algoritmi, care formează o Rețea Neurală Artificială (ANN- „*Artificial Neural Network"* ), care are un design inspirat din rețeaua neuronală a creierului uman. Învățarea profundă caută să înțeleagă informațiile pe care le primește și să învețe să ia decizii în mod independent ( Învățarea automată caută cea mai eficientă cale de a ajunge la rezultatul dorit )

Cele mai populare framework-uri pentru învățarea profundă sunt: Keras, Tensorflow și Pytorch.

Keras :

-rețea neurala, scrisă în Python;

- capabil să ruleze „peste" Tensorflow;

-experimentare rapidă cu *deep neural networks*;

-ușor de utilizat.

Tensorflow :

-librărie pentru a prelucra datele într-o serie de sarcini;

-este o biblioteca utilizată pentru rețele neurale;

-utilizat pentru design, construcția și a antrenarea modelelor de tip Deep Learning.

Pytorch :

-librărie pentru învățare automată pentru Python, care se bazează pe Torch;

-utilizată pentru aplicații, cum ar fi Procesarea Limbajului Natural ( NLP-*Natural Language Processing*);

-oferă flexibilitate și control mai mare.

Cele 3 au legături între ele și execută sarcini similare, dar sunt diferențiate de următorii parametrii:

- Nivel API ( „*Application Programming Interface”* );
- Viteză;
- Arhitectură;
- Ușurința de a scrie cod;
- Setul de date;
- Posibilitatea de debug;
- Suport comunitar;
- Tipuri de Rețele Neuronale Recurente;
- Popularitate.

Nivel API

| Keras | Tensorflow | Pytorch |
|---|---|---|
| -nivel ridicat API<br>-ușor de utilizat<br>-simplist sintactic | -conține și un nivel scăzut, dar și unul ridicat API<br>-atenția este îndreptată asupra lucrul direct cu martici | -nivel scăzut API<br>-atenția este îndreptată asupra lucrul direct cu expresii |

*Tabel 1: Nivel API*

Viteza

| Keras | Tensorflow | Pytorch |
|---|---|---|
| -performanță lenta | -performanță rapidă | - performanță rapidă |

*Tabel 2: Viteza*

Arhitectura

| Keras | Tensorflow | Pytorch |
|---|---|---|
| -simplistă, ușor de folosit | -mai complexă decât arhitectura folosită de Keras<br>-nu e ușor de folosit | -complexă<br>-greu de citit, în comparație cu arhitectura ultilizată de Keras |

*Tabel 3: Arhitectura*

Ușurință de a scrie cod

| Keras | Tensorflow | Pytorch |
|---|---|---|
| -cate o singură linie de cod | -dimensiuni reduse<br>-precizie ridicată | -mai multe linii de cod |

*Tabel 4: Ușurință de a scrie cod*

Setul de date

| Keras | Tensorflow | Pytorch |
|---|---|---|
| -viteză redusă → seturi mici de date | -viteză ridicată → seturi mari de date ( necesita execuție rapidă ) | -viteză ridicată → seturi mari de date ( necesita execuție rapidă ) |

*Tabel 5: Setul de date*

Posibilitatea de a face debug

| Keras | Tensorflow | Pytorch |
|---|---|---|
| - din cauza rețelelor simpliste, necesitatea de a face debug este mică | -greu de făcut | -capabilități bune de a face debug<br>-greu de găsit linia exactă care creează probleme |

*Tabel 6: Posibilitatea de a face debug*

Suport comunitar

| Keras | Tensorflow | Pytorch |
|---|---|---|
| -suport comunitar redus, când vine vorba de depanarea erorilor | -susținut de un număr mare de companii | -suport comunitar puternic |

*Tabel 7: Suport comunitar*

Tipuri de Rețele Neuronale Recurente

| Keras | Tensorflow | Pytorch |
|---|---|---|
| Librăria Keras conține starturi recurente.<br>Câteva nivele importante:<br>-*SimpleRNN* : RNN complet conectat, în care datele de ieșire vor fi din nou date de intrare<br>-*LSTM(„Long-Short Term Memory”)*: potrivit pentru clasificarea, procesarea și realizarea predicțiilor pe baza datelor din seriile de timp ( pot exista decalaje de timp necunoscut între evenimentele importante ).<br>-GRU (*„Gated Reccurent Unit Layer”)*: este ca o memorie LSTM cu poartă de uitare, dar are mai puțini parametrii, deoarece nu conține și o poartă de ieșire. | Deține un modul **tf.nn.rnn_cell** care ajută la gestionarea RNN standard.<br>Câteva clase importante:<br>- Clasa *MultiRNNCell*: utilizată pentru a stoca diverse celule, pentru a crea *deep RNN*<br>-Clasa *DropoutWrapper*: utilizată pentru implementarea abandonului (*dropout)*<br>- Clase de tip *Cell level:* utilizate pentru definirea unei singure celule RNN. | Oferă 2 nivele de clase pentru construirea rețelelor recurente:<br>-Clase *multi-layer*: obiectele pot reprezenta rețele neurale recurente bidirecționale<br>*-cell-level classes:* obiectele pot reprezenta o singură celula care poate gestiona o data de intrare într-o unitate de timp. |

*Tabel 8: Tipuri de Rețele Neuronale Recurente*

Popularitate

Keras a devenit cel mai popular datorită ușurinței cu care se utilizează, în comparație cu Tensorflow și Pytorch.

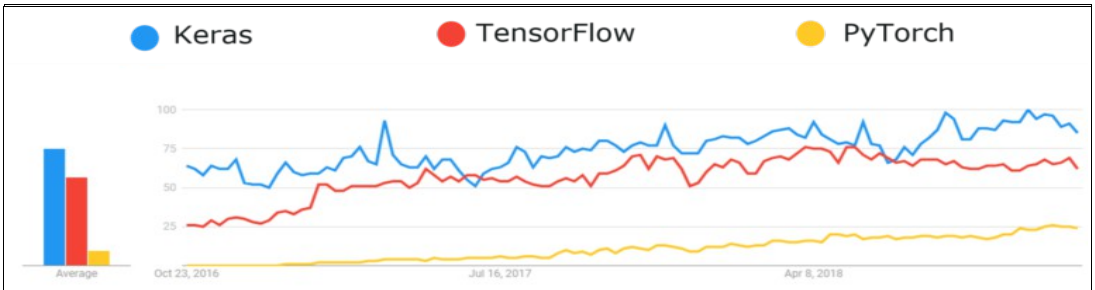



*Figura 1: Reprezentarea grafică a popularității celor 3 framework-uri*

## 2.7. Încorporările

Încorporările se pot folosi pentru maparea elementelor.

Mai departe este prezentat un exemplu, în care, folosind Filtrarea colaborativă vom recomanda filme.

### 2.7.1. Încorporarea unidimensională

În extremitatea stângă avem filme pentru copii, iar în extremitatea dreaptă avem filme pentru adulți.

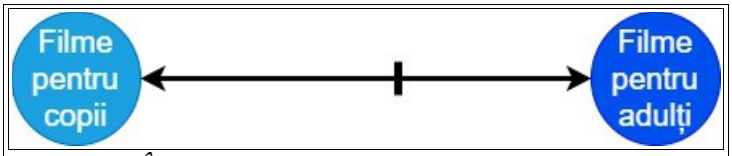



*Figura 2: Încorporarea unidimensională*

Când încercăm să aranjăm filmele unidimensional, nu vom avea precizie mare, deoarece sunt filme pentru copii cărora le plac și adulților (ex: Harry Potter) și sunt filme pentru adulți care le plac și copiilor (ex: Singur Acasă)

### 2.7.2. Încorporarea bidimensională

Pe axa Ox: în partea negativă sunt filmele pentru copii, iar în partea pozitivă sunt filme pentru adulți. Pe axa Oy: în partea pozitivă sunt filme de mare succes, iar în partea negativă sunt filme artistice.

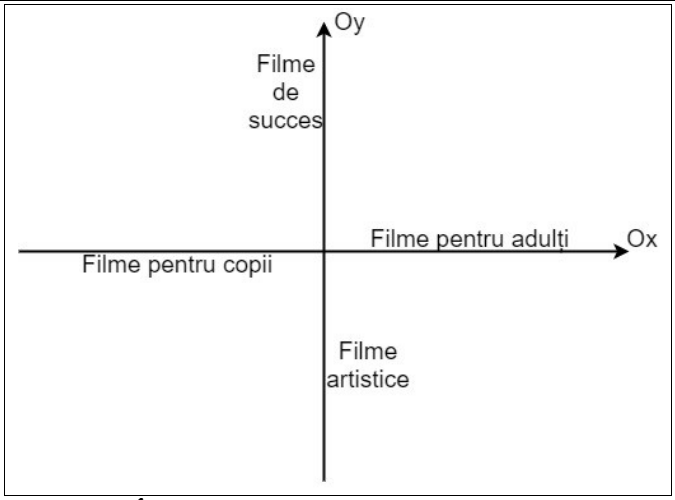



*Figura 3: Încorporarea bidimensională*

Astfel, filmele apropiate sunt mai similare decât în cazul incorporării unidimensionale.

Fiecare film reprezintă un punct de coordonate reale (ex: Harry Potter (-0.25, 0.95) )

În realitate, se lucrează cu d-dimensiuni, deoarece este greu să avem precizie bună pentru recomandări cu 2 dimensiuni. Astfel, fiecare punct bidimensional devine un punct d-dimensional, iar valoarea punctului reprezintă cat este de potrivit pentru un anumit utilizator.

### 2.7.3. Învățarea încorporărilor în rețele neurale

Pentru învățarea încorporărilor în rețele neurale folosim Backpropagation. Acest algoritm găsește valorile optime ale ponderilor dintre unitățile de învățare dintr-o rețea pe mai multe nivele, cu un număr fix de unități de învățare.

Dorim ca eroarea dintre valoarea obținută la ieșirea rețelei neurale și valoarea pe care o dorim sa fie cât mai mică.

Nu este nevoie de antrenare separată. Stratul de incorporări va fi un strat ascuns cu o unitate per dimensiune.

Intuitiv, unitățile ascunse descoperă cum să organizeze elementele în spațiul d-dimensional în așa fel încât să optimizeze cât mai bine obiectivul final.

Aranjăm datele sub forma unei matrici, astfel încât pe coloana să avem filmele, iar pe linie să avem utilizatorii.

Mapăm fiecare caracteristică într-un întreg între 1 și numărul_filmelor-1, astfel pentru un utilizator care a vizionat Shrek și Harry Potter, vom avea vectorul (1, 3 ), unde 1 și 3 reprezintă coloanele pe care sunt poziționate filmele (indexarea începând de la 0).

Folosind Filtrarea Colaborativă pentru a recomanda filme, este necesară următoarea împărțire: 30% din filme le vom folosi ca date de test, iar restul de 70% din filme vor fi datele de intrare.

Fiecare unitate ascunsă corespunde unei dimensiuni. După antrenare legăturile dintre filme și nivelul ascuns, sunt ponderi.

**De câte niveluri este nevoie?**

Un număr mare de niveluri poate conduce la o învățare mai bună a relațiilor dintre valorile de la intrare. Totodată, un număr mare de dimensiuni poate conduce la supra-învățare, învățare lentă sau necesitatea mai multor date de intrare.

## 2.8. Corectitudinea datelor

Este necesar ca datele folosite să fie alese corect. În caz contrar, recomandările vor fi incorecte.

### 2.8.1. Tipuri de biasuri

- Bias de raportare : apare atunci când frecvența evenimentelor, proprietăților și/sau rezultatelor incluse într-un set de date, nu reflectă frecvența reală.

- Bias de automatizare: reprezintă o tendință de a favoriza rezultatele generate de sistemele automate față de sistemele care nu sunt automate, fără a lua în calcul erorile obținute.

- Bias de selecție: apare atunci când exemplele setului de date sunt alese în așa fel încât nu reflectă lumea reală.

  Acest bias poate avea forme diferite:

  -bias de acoperire (datele nu sunt selectate într-o forma reprezentativă)

  -bias fără răspuns/ bias participant (datele ajung să fie nereprezentative din cauza lipsei participării la procesul de colectare a datelor)

  - bias de eșantionare (aleatorizarea adecvată nu este utilizată în timpul colectării datelor)

- Bias de atribuire unui grup: reprezintă tendința de a generaliza. Ce este adevărat pentru anumite date, va fi adevărat pentru tot grupul din care face parte. Există doua tipuri de manifestări ale acestui bias: bias în grup (o preferință pentru membrii unui grup de care aparțin sau caracteristici pe care le împărtășesc) și bias de omogenitate în afara grupului (tendința de a stereotipa membrii care aparțin unui alt grup).

- Bias implicit: apare atunci când presupunerile sunt făcute pe baza experiențelor personale, care nu se aplică neapărat într-un caz general. O formă comună a biasului implicit este biasul de confirmare în care se procesează datele astfel încât să afirme credințele și ipotezele apriori. În anumite cazuri, modelul este antrenat până când rezultatul confirmă ipotezele inițiale; această abordare se numește prejudecata experimentatorului.

### 2.8.2. Identificarea biasurilor

Ce trebuie evitat în cadrul datelor de antrenament: valori lipsă, valori neașteptate, variația datelor.

Valori lipsă

Dacă setul de date are una sau mai multe caracteristici care au valori lipsă pentru un număr mare de exemple, acesta ar putea indica faptul că anumite caracteristici cheie ale setului de date sunt sub-reprezentate. Este recomandat ca înainte de a antrena modelul, trebuie observat dacă există valori care lipsesc, iar dacă există, trebuie identificat motivul.

Valori neașteptate

Trebuie căutate valorile care ies în evidență (zgomote), deoarece acestea pot indica probleme apărute în timpul colectării datelor.

Variația datelor

Orice tip de variație a datelor, în care anumite grupuri de caracteristici pot fi subreprezentate sau supra-reprezentate în raport cu relevanța lor în lumea reală, pot introduce biasuri în model.

### 2.8.3. Evaluarea biasurilor

Când se evaluează modelul (pe set întreg sau un set de validare), rezultatele reflectă cât de corect este acest model.

Să presupunem că avem un model care prezice prezența tumorilor. Acest model este evaluat pe baza unui set de validare de 1000 înregistrați ale pacienților (500 femei și 500 bărbați).

Rezultatele sunt:

| | |
|---|---|
| Pozitiv adevărat: 16 | Fals pozitiv: 4 |
| Fals negativ: 6 | Negativ adevărat: 974 |

*Tabel 9: Rezultate*

Putem calcula :

*Precizie*: P = $\frac{TP}{TP+FP} = \frac{16}{16+4} = 0{,}800$; *Rapel*: R= $\frac{TP}{TP+FN} = \frac{16}{16+6} = 0{,}727$.

Dacă facem calculele separat pentru femei și bărbați, putem observa mai bine diferențele de performanță.

| Femei | | Bărbați | |
|---|---|---|---|
| Pozitiv adevărat: 10 | Fals pozitiv: 1 | Pozitiv adevărat: 6 | Fals pozitiv: 3 |
| Fals negativ: 1 | Negativ adevărat: 488 | Fals negativ: 5 | Negativ adevărat: 486 |

*Tabel 10: Diferențele de performanță separat pentru femei și bărbați*

Pfemei= $\frac{TP}{TP+FP} = \frac{10}{10+1} = 0{,}909$ Pbarbati= $\frac{TP}{TP+FP} = \frac{6}{6+3} = 0{,}667$

Rfemei= $\frac{TP}{TP+FN} = \frac{10}{10+1} = 0{,}909$ Rbarbati= $\frac{TP}{TP+FN} = \frac{6}{6+5} = 0{,}545$

| Concluzii în legătura cu pacienții de sex feminin: | Concluzii în legătura cu pacienții de sex masculin: |
|---|---|
| - din cele 11 femei care au tumori, modelul prezice corect pozitiv pentru 10 pacienți(rata de rapel: 90.9%). Modelul ratează o diagnosticare în 9.1% din cazuri.<br><br>- atunci când modelul returnează pozitiv, predicția este corectă în 10 din 11 cazuri (rata de precizie: 90,9%). Modelul prezice incorect în 9.1% din cazuri. | - pentru bărbații care au tumori, modelul prezice corect pozitiv pentru doar 6 pacienți (rata de rapel: 54,5%). Modelul ratează o diagnosticare în 45.5% din cazuri.<br><br>- atunci când modelul returnează pozitiv, predicția este corectă în doar 6 din 9 cazuri ( rata de precizie: 66,7%). Modelul prezice incorect în 33.3% din cazuri. |

*Tabel 11: Concluzii în legătura cu pacienții*

## 2.9. Concepte despre învățarea automată

### 2.9.1. Încadrarea problemei

-eticheta reprezintă ce prezicem

-caracteristica reprezintă  date de intrare

-modelul reprezintă relația dintre etichetă și caracteristică

Un model are 2 faze: antrenarea (învățarea sau crearea unui model) și testarea (un model aplică ce a învățat).

Regresia determină modelul prezice valori continue.

Clasificarea determină modelul prezice valori discrete.

### 2.9.2. Detalierea problemei

-ecuația modelului: y' = b+w1*x1, unde y'= ieșirea dorită

b= bias

w1= panta (greutate)

x1= data de intrare

Antrenarea reprezintă învățarea valorilor corecte pentru toate ponderile(w) și biasurilor(b) din exemple.

Pierderea reprezintă penalizara pentru predicție greșită

### 2.9.3. Reducerea pierderilor

De obicei, graficul pierderilor este sub forma de „U” și trebuie aflat punctul în care pierderea este minimă.

Un gradient este un vector cu 2 caracteristici:

- direcție
- mărime.

Începem dintr-un anumit punct, iar gradientul ne arată în ce direcție trebuie să o luam și cât de mare să facem pasul în acea direcție.

Dacă pasul (mărimea gradientului) este prea mic, determinarea pierderii minime durează foarte mult. În schimb, dacă pasul este prea mare, putem rata valoarea minimă, trecând peste ea.

*Stochastic gradient descent (SGD)*= folosește doar un exemplu (un lot de dimensiune 1) pentru fiecare iterație. După ce se realizează suficiente iterații, SGD funcționează bine(dar zgomotos).

*Mini-batch stochastic gradient descent (mini-batch SGD)*= este un compromis între iterația cu un lot întreg și SGD. Un astfel de lot este de obicei între 10 și 1000 exemple, alese la întâmplare. Acest lot reduce zgomotul care era prezent la SGD și este mult mai eficient decât alegerea unui lot întreg.

### 2.9.4. Generalizare

Setul de date trebuie divizat în 2 subseturi: unul pentru antrenare și unul pentru testare.

O performanță bună pe setul de test este un indicator bun al performantei pe date noi dacă: setul de test este destul de mare și nu utilizam același set de test de fiecare dată.

### 2.9.5. Antrenare și testare

Datele se pot împarți în 3 seturi:

- set de antenare = subset pentru a antrena modelul
- set de testare = subset pentru a testa modelul antrenat

Cel mai important aspect este asupra datelor de test, deoarece antrenarea modelului nu trebuie făcută pe setul de test. Dacă modelul este antrenat pe aceste date, el va supra-învăța si rezultatele nu vor mai fi relevante.

### 2.9.6. Setul de validare

Setul de validare este folosit pentru a evalua rezultatele din setul de antrenament. După ce modelul a „trecut” de setul de validare, se utilizează setul de testare pentru a verifica evaluarea făcută.

### 2.9.7. Reprezentare

Datele „crude” se mapează în datele de intrare prin: *Feature engineering* (transformarea de la date ”crude” într-un vector, ca dată de intrare).

Această reprezentare se numește „*one-hot encoding”* (când o singură poziție din vector are valoarea 1) sau „*multi-hot encoding”* (mai multe valori sunt 1).

### 2.9.8. Încrucișarea caracteristicilor

Apare o problemă atunci când pe grafic, o sigură liniile nu poate să separe 2 clase.

Încrucișarea caracteristicilor se formează atunci când se încrucișează 2 sau mai multe date de intrare.

Această încrucișare poate oferi abilitați de prezicere mai mari decât datele de intrare individuale.

Putem avea mai multe tipuri de caracteristici încrucișare:

[A X B]: valorile a 2 caracteristici se multiplică

[A x B x C x D x E]: valorile a 5 caracteristici se multiplică

[A x A]: ridicare la pătrat a unei caracteristici.

### 2.9.9. Regularizare

Regularizare L2: se definește termenul de regularizare ca suma pătratelor ponderilor. Ponderile se vor apropia asimptotic de 0.

Se poate ajusta impactul general al regularizării prin multiplicarea valorii sale cu lambda (numită și rata de regularizare) astfel:

minimizare(pierderi(data|model) + lambda*complexitate(Model))

Dacă lambda este prea mare, modelul va fi prea simplu și există riscul de subapreciere a datelor. Dacă lambda e prea mic, modelul va fi complex și există riscul de supraapreciere a datelor (modelul va învăța prea mult particularitățile datelor de antrenament).

### 2.9.10. Regresia logistică

Metodă de predicție din care rezultă probabilități bine calibrate.

Funcția pentru pierderi a Logistic Regresion este numită Logloss și este definită:

$$\text{Log Loss} = \sum_{(x,y)\in D} -y\log(y') - (1-y)\log(1-y')$$

### 2.9.11. Clasificare

*TP(true positive)*= modelul a prezis corect o clasă pozitivă

*FP(false positive)*= modelul a prezis incorect o clasă pozitivă

*TN(true negative)*= modelul a prezis corect clasă negativă

*FN(false negative)*=modelul a prezis incorect o clasă negativă

Acuratețea= folosită pentru evaluarea clasificării modelelor

A=(TP+TN)/(TP+TN+Fp+FN)

Precizia = ce proporție din identificările pozitive sunt corecte

P=TP/(TP+FP)

Reapelare *(Recall)*= ce proporție din ce am identificat pozitiv este corect

R=TP/(TP+FN)

*TPR(True Positive Rate)* = R

*FPR(false Positive Rate)* = FP/(FP+TN)

Curba ROC *(receiver operating characteristic curve)* = este un grafic care arată performanța unui model pentru toate pragurile de clasificare și este definită de TPR și FPR

Aria de sub ROC se numește AUC*(Area Unter the ROC)*.

Predicția biasului măsoară cat de departe sunt cele 2 medii și se calculează :

bias_de_predictie = media_predictiilor - media_datelor

### 2.9.12. Regularizare

| Regularizare L1 | Regularizare L2 |
|---|---|
| • Penalizează \|w\|, unde w=pondere<br>• derivata lui L1 = k, unde k este o constantă, independentă de w | • Penalizează $w^2$<br>• derivata lui L2 = 2*w |

*Tabel 12: Diferența dintre L1 și L2*

## 2.10. Tipuri de învățare automată

- Învățare Supravegheată: modelul reiese din datele etichetate

- Învățare Nesupravegheată**:** găsește tipare (care nu provin din setul de date)

- Învățare Prin Recompensare: pentru fiecare sarcină dusă la bun sfârșit, se oferă o recompensă.

Învățarea automată cuprinde :

- Clasificare: se alege 1 din N posibilități
- Regres: prezice valori numerice
- Grupare: grupează exemplele similare
- Asociere: identifică modele care se pot asocia
- Ieșire structurată: creează date complexe de ieșire
- Clasament: identifică poziția pe o scară/status

Matplotlib este folosit pentru vizualizarea datelor.
Seaborn este folosit pentru funcții *heatmap.*
Pandas este folosit pentru manipularea datelor.
NumPy este folosit pentru operații matematice simple.
Învățarea Scikit este folosită  evaluarea valorilor.

## 2.11. Sisteme bazate pe învățare automată în lumea reală

*Predicția pacienților bolnavi de cancer*

Un model este antrenat să prezică probabilitatea unui pacient de a avea cancer din fișele medicale.

Datele de intrare conțin: vârsta, sex, condiții medicale, numele spitalului, semne vitale și rezultatele testelor.

Modelul generează o performanță foarte bună pe datele de test, dar când au fost introduși pacienți noi, rezultatul modelului nu a fost satisfăcător.

Numele spitalului influenta foarte mult decizia modelului. Anumite spitale sunt specializate în tratarea cancerului și majoritatea pacienților din aceste spitale au cancer.

Atunci când apar pacienți noi, care nu aparțin de nici un spital, apare problema. Acest fenomen este cunoscut sub numele de *„label leakage”.*

*Literatura*

Un profesor a vrut sa prezică afilierea politică a autorilor bază doar pe metaforele folosite.

O echipă de cercetători a făcut un set de date care conține mai multe opere ale mai multor autori (propoziție cu propoziție) și a împărțit setul în set de antrenare, de validare, de test.

Modelul a avut rezultate aproape perfecte pe datele de test, dar cercetătorii au crezut că acest lucru e suspicios de exact.

Împărțirea datelor se poate face în 2 feluri:

*A*. Câte un exemplu de la fiecare autor se va împărți în date de antrenament, de validare și de test.

*B.* Toate exemplele unui autor vor fi incluse într-un singur set. Operele unui autor au fost plasate ca date de antrenare, operele altui autor ca date de validare și operele altui autor ca datele de test.

Ca și rezultat final, varianta A a avut o precizie mult mai bună decât varianta B.

Concluzie: chiar dacă am avut aceleași date și în cazul A și în cazul B, împărțirea diferită, a condus la precizii diferite.

*Orientare în cadrul învățării automate*

- Primul model trebuie să fie simplu.
- Concertarea trebuie îndreptata înspre corectitudinea datelor.
- Utilizarea unei metrici simple și observabile pentru antrenare și evaluare.
- Deținerea și monitorizarea datelor de intrare.
- Tratarea configurației modelului ca pe un cod: trebuie revizuit și verificat.

Notarea rezultatelor, în special a eșecurilor.

Învățarea automată conține (pe lângă antrenarea datelor):

- colecția datelor
- extracția datelor de intrare
- verificarea datelor
- monitorizare
- analizarea datelor

Antrenarea datelor

| Antrenare statică (*off-line*) | Antrenare dinamică (*on-line*) |
|---|---|
| Antrenarea se face o dată, înainte de a fi folosit modelul.<br><br>Avantaj: este ușor de utilizat și testat.<br><br>Dezavantaj: necesită monitorizarea datelor de intrare.<br><br>Se folosește când presupunem că datele nu se vor schimba prea mult în timp. | Date noi apar continuu în sistem și sunt incorporate în model.<br><br>Avantaj: se adaptează la datele care se schimbă în timp<br><br>Dezavantaj: are nevoie de monitorizare<br><br>Se folosește când datele se schimbă. |

*Tabel 13: Diferența dintre antrenarea datelor on-line și off-line*

Deducție statică sau dinamică

| Deducție statică (off-line) | Deducție dinamică (on-line) |
|---|---|
| Se calculează toate predicțiile posibile dintr-un lot. După ce se face predicția, aceasta se scrie într-un tabel.<br><br>Avantaj: după predicție, se verifică datele pentru a le valida.<br><br>Dezavantaj: se poate prezice doar ce este cunoscut.<br>Dezavantajul apare la un set mare de date. | Predicția se face la cerere, utilizând un server.<br><br>Avantaj: Capabil să facă predicția pentru orice element nou. Este util pentru set mare de date.<br><br>Dezavantaj: sensibil la latență (poate limita complexitatea) și necesită mai multă monitorizare. |

*Tabel 14: Diferența dintre deducția statică și dinamică*

Dependențe de date

Dependențele de date în învățarea automată sunt similare cu dependențele de date din programare. Este de preferat ca numărul dependențelor să fie cât mai mic.

Datele de intrare determină comportamentul sistemului. Fiecare set de date de intrare, determină alt comportament la sistemului.

Datele de intrare trebuie să răspundă la următoarele întrebări:

1. Este de încredere?

   Semnalul (datele de intrare) va fi întotdeauna disponibil sau provine dintr-o sursă nesigură?

2.Se va schimba în timp?

   Dacă da, trebuie să ne gândim la versiuni ale acelei date.

   Se schimbă sistemul care calculează aceste date? Dacă da: Cât de des?

3. Este necesară?

   Utilitatea funcției justifică costul includerii acesteia?

4. Există corelații?

   Există corelații atât de strânse între semnalele mele, astfel încât avem nevoie de strategii suplimentare de eliminare?

5. Bucle de reacție (*feedback loops*).

   Care dintre semnalele mele de intrare pot fi afectate de rezultatele modelului meu? Uneori, un model își poate afecta propriile date de antrenare. De exemplu, rezultatele unor modele, la rândul lor, sunt caracteristici de introducere directă sau indirectă la același model.

## 2.12. Rețele Neurale

O Rețea Neurală este un set de noduri, organizate în straturi. Legăturile dintre straturi se fac cu ajutorul seturilor de greutăți. Conține și un set de biasuri pentru fiecare nod și o funcție de activare care transformă ieșirea fiecărui nod într-un strat.

ReLU*(rectified linear unit)*= funcție de activare

### 2.12.1. Antrenarea Rețelelor Neurale

Cel mai frecvent algoritm de antrenarea rețelelor neuronale este *Backpropagation*. Acesta face realizabilă descrierea gradientului pentru rețelele neuronale cu mai multe straturi.

### 2.12.2. Rețele neurale cu mai multe clase

*„One vs. All”*: modelul trece printr-o secvență de clasificări binare, antrenând fiecare clasificare să răspundă la întrebări de clasificare separate.

*„Softmax”* asignează probabilități decimale pentru fiecare clasă dintr-o problemă cu mai multe clase.

*„Softmax”* poate fi:

- *„full Softmax”* : calculează probabilități pentru fiecare clasă în parte
- *„candidate sampling”*: calculează probabilități pentru toate etichetele pozitive, dar doar pentru o etichetă negativă aleasă la întâmplare.

### 2.12.3. Incorporările

Ajută la translatarea vectorilor multidimensionali în vectori unidimensionali.
Filtrarea colaborativă: face preziceri pe baza preferințelor majoritare.

## 2.13. Sisteme de recomandări

Sistemele de recomandare încearcă să ghicească interesele utilizatorilor și să recomande produse. Sunt cele mai puternice sisteme de tip învățarea automată pe care distribuitorii le implementează, pentru a crește vânzările. Datele necesare pentru a genera recomandări provin din aprecierile date de utilizatori. Aceste aprecieri rezultă din interogările motoarelor de căutare, din achizițiile precedente sau alte cunoștințe despre utilizator.

### 2.13.1. Funcționalitatea sistemelor de recomandări

Aceste sisteme funcționează cu 2 tipuri de informații:

- informație caracteristică: constă în informații despre produse și utilizatori;
- interacțiune utilizator-produs: constă în evaluări, număr de achiziții, aprecieri.

Pe baza celor 2 tipuri de informații, se pot distinge 3 algoritmi utilizați în sistemele de recomandare:

- Sistem bazat pe conținut: folosește informații caracteristice;
- Sistem de filtrare colaborativ: bazat pe interacțiuni de tip utilizator-produs;
- Sisteme hibride: combină cele două tipuri de informații.

#### 2.13.1.1. Sisteme bazate pe conținut

Recomandările se fac pe baza produselor apreciate de utilizator și caracteristicile profilului acestuia. Dacă un utilizator a fost interesat de un obiect în trecut, acesta va fi interesat de acel obiect și în viitor. Obiectele similare se grupează, în funcție de caracteristicile lor.

*Probleme:*

1. dacă utilizatorul a fost interesat de câteva categorii, sistemul nu poate recomanda în afara acelor categorii, chiar dacă utilizatorul ar putea fi interesat și de altele.
2. utilizatorii noi nu au trecut pe care să se poată baza sistemul.

#### 2.13.1.2. Sistem de filtrare colaborativ

În prezent este unul din cele mai utilizate sisteme și oferă rezultate mai bune decât recomandările bazate pe conținut. Sistemul folosește interacțiunea utilizatorilor pentru filtrarea produselor.

*De exemplu*: doi utilizatori (u1 și u2) au apreciat aceleași produse (p1 și p2). Dacă u1 apreciază și un al treilea produs, p3, atunci u2 ar putea fi interesat de p3.

Pentru a putea ajunge la un rezultat, se pot urma 2 metode: bazate pe memorie și bazate pe model

Bazate pe memorie : exista doua abordări:

- Prima identifică cluster-ele utilizatorilor și se folosește de interacțiunile unui utilizator, pentru a prezice comportamentul altui utilizator.
- A doua, identifică cluster-ele produselor, care au fost evaluate de un utilizator și folosite să prezică interacțiunea dintre acel utilizator și oricare altul.

Problema apare la lucrul cu matrici mari (numărul de interacțiuni utilizator-produs poate fi prea mic pentru generarea de cluster-ele de înaltă calitate)

Bazate pe model: folosește tehnici de învățarea automată și data mining. Scopul este de a antrena modele pentru a putea face predicții cât mai bune.

Avantajul adus de acest model este capabilitatea de a face multe recomandări pentru multi utilizatori (are o acoperire mare, chiar dacă lucrează cu matrici mari)

*Problemele sistemului de filtrare colaborativ:*

1. Start rece: sistemul nu poate funcționa până când un utilizator nu are suficientă interacțiune (utilizator-produs sau utilizator-utilizator);
2. Adăugarea unui utilizator/produs nou: nu contează dacă este utilizator sau produs nou, nu există informații precedente asupra lor, deoarece nu au fost interacțiuni.

### 2.13.2. Tehnici folosite pentru construirea unui sistem de recomandări

Cele 2 tehnici prin care se poate construi un sistem de recomandări sunt: Rețea neurală complet conectata și Item2vec.

#### 2.13.2.1. Rețea neurală complet conectată

Utilizatorului și produsului le corespund vectori de dimensiuni diferite. Acest lucru înseamnă că trebuie să învățăm reprezentările utilizatorilor și produselor, numite încorporări (deoarece încorporam aceste concepte într-un spațiu vectorial). Începem cu o valoare aleatoare, pentru ca nu avem alte valori pentru acești vectori.

Pentru fiecare interacțiune utilizator-produs vom concatena atât încorporarea utilizatorului cât și a produsului, pentru a rezulta un singur vector. Deja cunoaștem rezultatul acestei interacțiuni utilizator-produs, deci putem forța ieșirea rețelei să fie la fel.

În pasul următor, rețeaua va folosi *backpropagation* (algoritm utilizat pentru antrenarea conexiunilor neurale) pentru ajustări, astfel încât rezultatul să obținut să fie la fel cu cel pe care îl așteptam. Rețeaua va învață cea mai bună modalitate de a reprezenta utilizatorii și elementele.

Un rezultat al acestei abordări este faptul că încorporările conțin informații asemănătoare și rețeaua va fi utilă pentru a prezice interacțiunile noi.

#### 2.13.2.2. Item2vec

Se folosește de produsele cumpărate, ca informații contextuale.

Această abordare nu implică în mod direct utilizatorii și nici nu îi ia în considerare atunci când se generează o recomandare.

Dacă obiectivul este de a arăta utilizatorului și alte alternative pentru produsul ales (din categorii diferite), atunci această abordare este foarte utilă.

*Problemă:* necesită foarte multe date pentru a produce încorporarea bună.

### 2.13.3. Condițiile necesare pentru construirea unui sistem de recomandări

Organizarea datelor este cel mai important aspect. Trebuie să fie cunoscute detalii despre utilizatori și produse. Cu cât mai multe seturi de date, cu atât mai bine va funcționa sistemul.

Ce trebuie reținut despre interacțiunile utilizator-produs:

- definirea interacțiunilor trebuie făcută astfel încât sistemul să extragă ușor datele;
- interacțiunile pot fi definite explicit (caracterizat de situații) sau implicit (interesul este determinant din acțiuni);
- mai multe interacțiuni pentru fiecare utilizator și produs, ajută sistemul să funcționeze mai bine;
- sistemele de recomandare funcționează bine cu produsele populare.

### 2.13.4. Evaluarea unui sistem de recomandări

Evaluarea unui astfel de sistem se poate face prin doua tipuri de metode:

1. Metode on-line (sau metode de testare A / B): reacțiile utilizatorilor sunt analizate luând în considerare recomandările făcute. Această abordare este ideală, dar singura modalitate de rulare a experimentelor este prin interacțiunea cu un sistem care este deja în producție. Folosirea clienților reali pentru experimente va fi mai lenta decât dacă datele există deja.

2. Metode off-line: utilizatorul nu este direct implicat, iar sistemul nu trebuie să fie în producție. O parte a datelor va fi folosită pentru construirea sistemului (date pentru antrenament), iar cealaltă parte va fi folosită pentru evaluarea acestuia (date de evaluare).

## 3. Descrierea formală a aplicației

Aplicația are ca scop recomandarea unor picturi similare cu cele apreciate de un anumit utilizator.

Pentru a face acest lucru, am avut nevoie de o interfață cu utilizatorul prin care acesta poate interacționa cu baza de date.

Fiecare pictură are un set de atribute, pe baza cărora se vor face recomandările.

După ce utilizatorul se conectează, acesta poate aprecia oricâte picturi dorește (dacă nu apreciază nici o pictură, îi sunt recomandate 2 picturi, care vor fi ulterior înlocuite).

După ce picturile au fost apreciate, sistemul de recomandări calculează similaritatea fața de celelalte picturi și recomandă 2, cele mai similare.

### 3.1. Actori si use-case

Aplicația suportă 2 actori:

- Vizitator
- Utilizator

Use-case arată interacțiunea dintre aplicație și utilizatorii acesteia. Pentru această aplicație, pot fi prezenți 2 tipuri de actori: vizitatori și utilizatori.

Vizitator:

Vizitatorul poate vedea picturile din orice categorie, poate citi despre evenimente și vizualiza pozele corespunzătoare evenimentului. Dacă dorește să aprecieze o pictură (butonul fiind vizibil), va apărea o eroare care atenționează vizitatorul că trebuie sa fie înregistrat pentru a face acest lucru. Vizitatorul are posibilitatea de a-și crea un cont nou. Odată ce își creează un cont, acesta părăsește statutul de vizitator și devine utilizator. Dacă numele pe care vizitatorul dorește să îl aleagă atunci când își crează un nou cont, a fost deja folosit, va apărea un mesaj de eroare și vizitatorul trebuie să își aleagă alt nume.

Utilizator:

Utilizatorul are un cont cu care se înregistrează, folosind numele și parola pe care și le-a ales atunci când și-a creat contul. Dacă una din ele este greșită, va apărea un mesaj de eroare și utilizatorul nu poate intra în contul său. Spre deosebire de vizitator, utilizatorul poate aprecia picturi

și beneficiază de recomandările făcute pe baza picturilor apreciate anterior. Dacă utilizatorul nu a apreciat nici o pictură (acest caz apare atunci când este făcut un cont nou, sau dacă utilizatorul nu a găsit nici o pictură pe placul lui) vor fi recomandate 2 picturi. Acestea vor fi ulterior înlocuite, dacă utilizatorul apreciază picturi.

### 3.1.1. Vizitator use-case

| Actor | Vizitator |
|---|---|
| Descriere | Vizitatorul poate vedea picturile, citi despre evenimente și își poate crea un cont nou. |
| Precondiții | Necesitatea folosirii unui browser. |
| Postcondiții | Se pot vizualiza picturi, evenimente și se poate crea cont. |
| Scenariu | Vizitatorului îi este permis să vizualizeze atât picturile, cât și evenimentele. Dacă acesta încearcă să aprecieze o pictură, va apărea o eroare (care spune că trebuie să fie conectat pentru a aprecia picturile).<br>Utilizatorul are posibilitatea de a-și crea un cont nou (devenind utilizator). |

*Tabel 15: Vizitator use-case*

### 3.1.2. Utilizator use-case

| Actor | Utilizator |
|---|---|
| Descriere | Utilizatorul poate vizualiza picturile, evenimentele, dar poate aprecia picturi și se poate bucura de recomandările făcute. |
| Precondiții | Necesitatea folosirii unui browser. |
| Postcondiții | Se pot aprecia picturi și vedea recomandările. |
| Scenariu | Utilizatorului îi este permis să vizualizeze atât picturile, cat și evenimentele.<br>Poate aprecia oricâte picturi dorește și se poate bucura de recomandările făcute. |

*Tabel 16: Utilizator use-case*

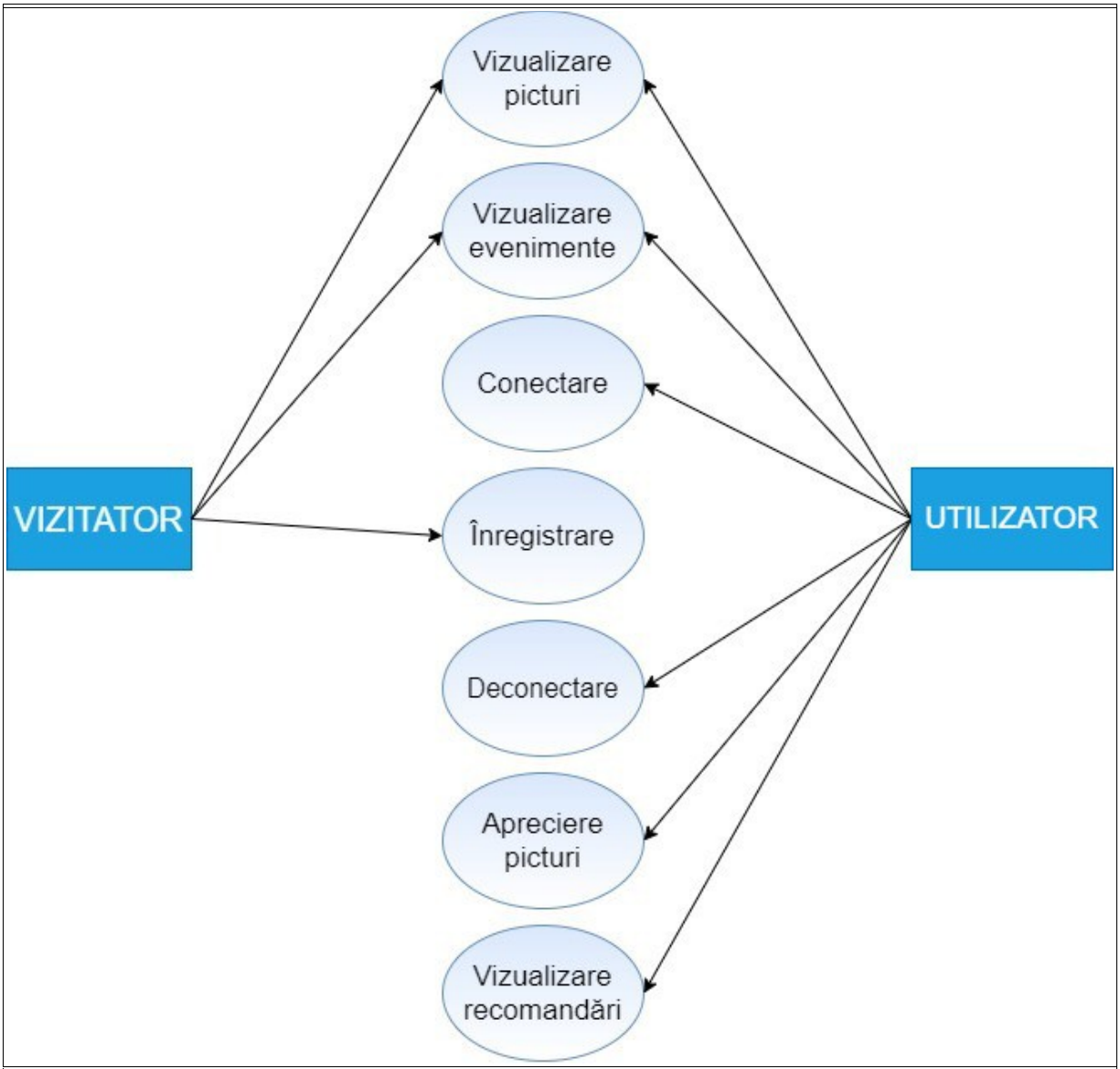



*Figura 4: Diagrama corespunzătoare use-case-urilor*

Din această diagramă se pot observa ce acțiuni sunt atribuite fiecărui actor.

Atunci când vizitatorul alege să se înregistreze, acesta va deveni utilizator și va avea permisiunea de a aprecia picturi, a vedea recomandările făcute pentru el, iar în final, a se deconecta și reconecta oricând dorește, cu contul pe care și l-a creat.

## 4. Detalii de implementare

Baza de date a aplicației este una simplă, care constă în 2 fișiere: Picturi.csv și User.csv.

Fișierul Picturi.csv conține câmpurile:

- id_pictura: reține id-ul fiecărei picturi;
- cale_poza: reprezintă calea completă către fișierul în care se află poza, împreună cu numele acesteia;
- denumire: conține titlul picturii;
- categorie: conține categoria din care face parte pictura;

| *codificare* | *categorie* |
|---|---|
| 1 | picturi pe pânza |
| 2 | picturi pe perete |
| 3 | picturi pe căni/ pahare |
| 4 | felicitări |
| 5 | mărțișoare |
| 6 | meniuri |

*Tabel 17: Codificarea categoriilor*

- taguri: reprezintă atributele care caracterizează cel mai bine pictura;
- disponibilitate:  indica dacă pictura este disponibilă sau nu ;

| *codificare* | *disponibilitate* |
|---|---|
| 0 | indisponibil |
| 1 | disponibil |

*Tabel 18: Codificarea disponibilității*

- preț: conține prețul picturii;
- dimensiune: conține dimensiunea picturii (lungime x lățime [cm]);
- descriere: conține descrierea picturii;

User.csv conține:

- id_user: conține id-ul fiecărui utilizator;
- username: conține numele pe care și l-a ales utilizatorul;
- mail: conține adresa de mail a utilizatorului;
- parola: reține parola aleasă de utilizator;
- picturi_apreciate: conține picturile apreciate de utilizator;

Câmpurile acestor fișiere sunt delimitate de caracterul „#”. Nu se putea folosi ca delimitator virgula, punctul sau spațiul, deoarece aceasta apare de multe ori în câmpul „descriere” și s-ar fi produs erori la separarea câmpurilor.

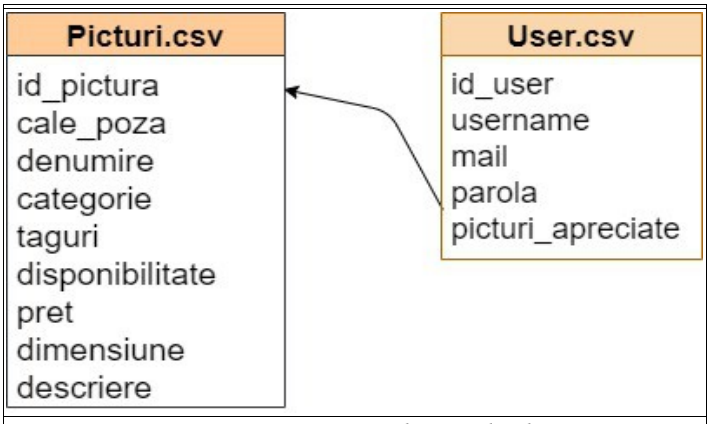



*Figura 5: Structura bazei de date*

Alte 3 fișiere importante, fără de care aplicația nu poate funcționa, sunt: picturi_apreciate.csv, lista_picturui_recomandate.csv, picturi_recomandate.txt.

Folosind aceste fișiere, aplicația citește picturile apreciate de un anumit utilizator (din picturi_apreciate.csv), generează recomandările (și le scrie în lista_picturui_recomandate.csv), apoi se citesc recomandările (din listă sunt extrase id-urile picturilor și sunt scrise în picturi_recomandate.txt).

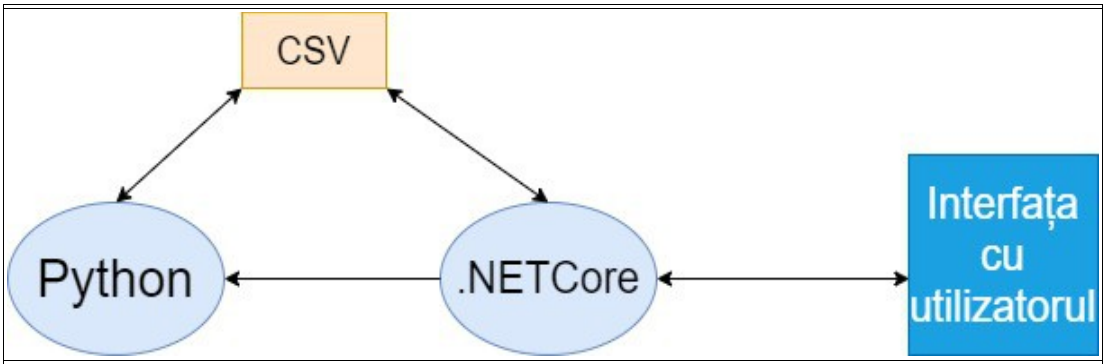


*Figura 6: Structura aplicației*

Structura aplicației constă într-o interfață cu utilizatorul, care îl ajută să interacționeze cu baza de date. Aceasta bază de date este alcătuită din mai multe fișiere CSV. Gestionarea datelor este realizată cu ajutorul a 2 aplicații.

Prima aplicație este cea scrisa în .NETCore, care are ca scop prelucrarea datelor introduse de utilizator, citirea/scrierea din/în baza de date.

Cea de-a doua aplicație constă într-o aplicație scrisă în Python, care are ca scop generarea recomandărilor.

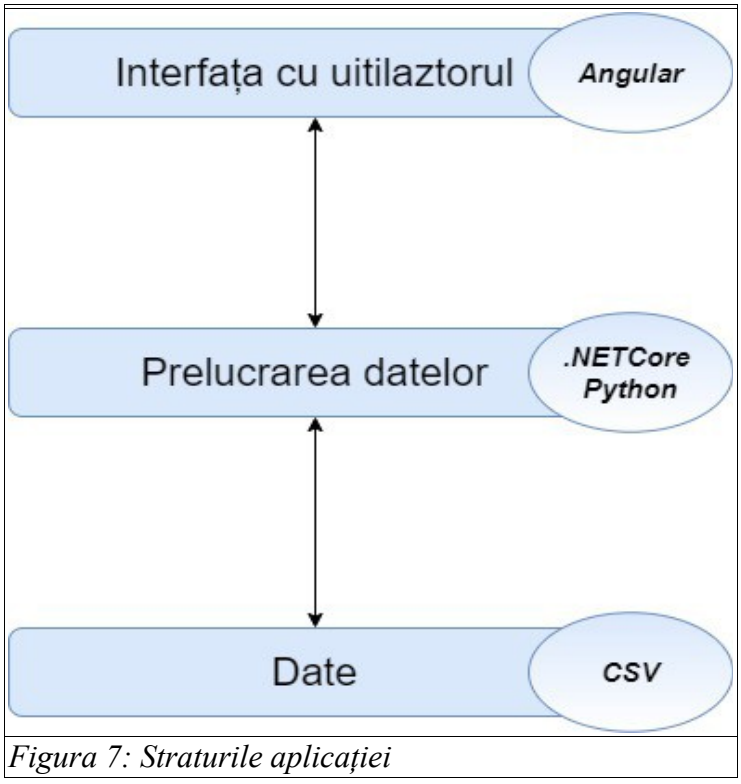


*Figura 7: Straturile aplicației*

Aplicația conține 3 straturi: date, prelucrarea datelor și interfața cu utilizatorul.

Stratul de date conține baza de date, fișierele CSV, care vor fi prelucrate de stratul superior, adică aplicațiile .NETCore și Python. Acestea scriu/citesc în/din aceste fișiere, în momentul în care apare o cerere de la stratul superior (de la utilizator).

Ultimul strat constă în interfața cu utilizatorul. Aceasta permite utilizatorului să interacționeze cu baza de date, într-un mod transparent, fără a vedea cum sunt gestionate datele.

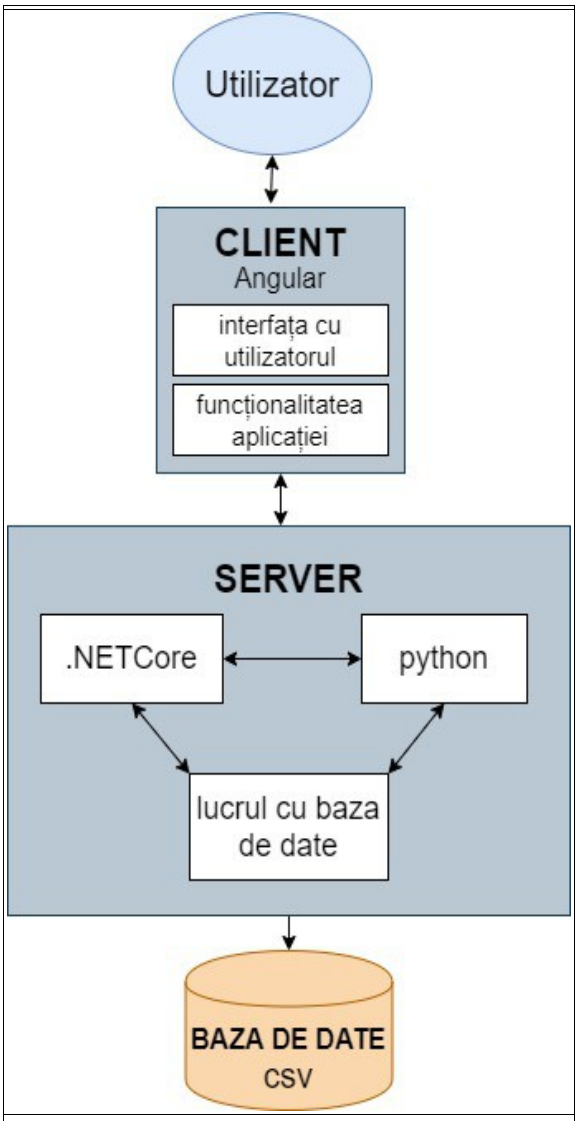


*Figura 8: Structura aplicație client-server*

Aplicația client-server este alcătuită din 2 programe principale: clientul și serverul. Aceste 2 programe trebuie să comunice între ele.

Serverul constă într-un program care rulează neîncetat. Acesta are ca scop gestionarea datelor de la client și furnizarea datelor către acesta.

Clientul este programul care trimite date către server și interpretează datele pe care le primește de la acesta.

Pentru a face comunicarea posibilă, aceste programe trebuie sa se găsească reciproc:

- clientului trebuie specificat localhost-ul serverului: „localhost:51476”;
- serverului trebuie specificat localhost-ul clientului: „localhost:4200”.

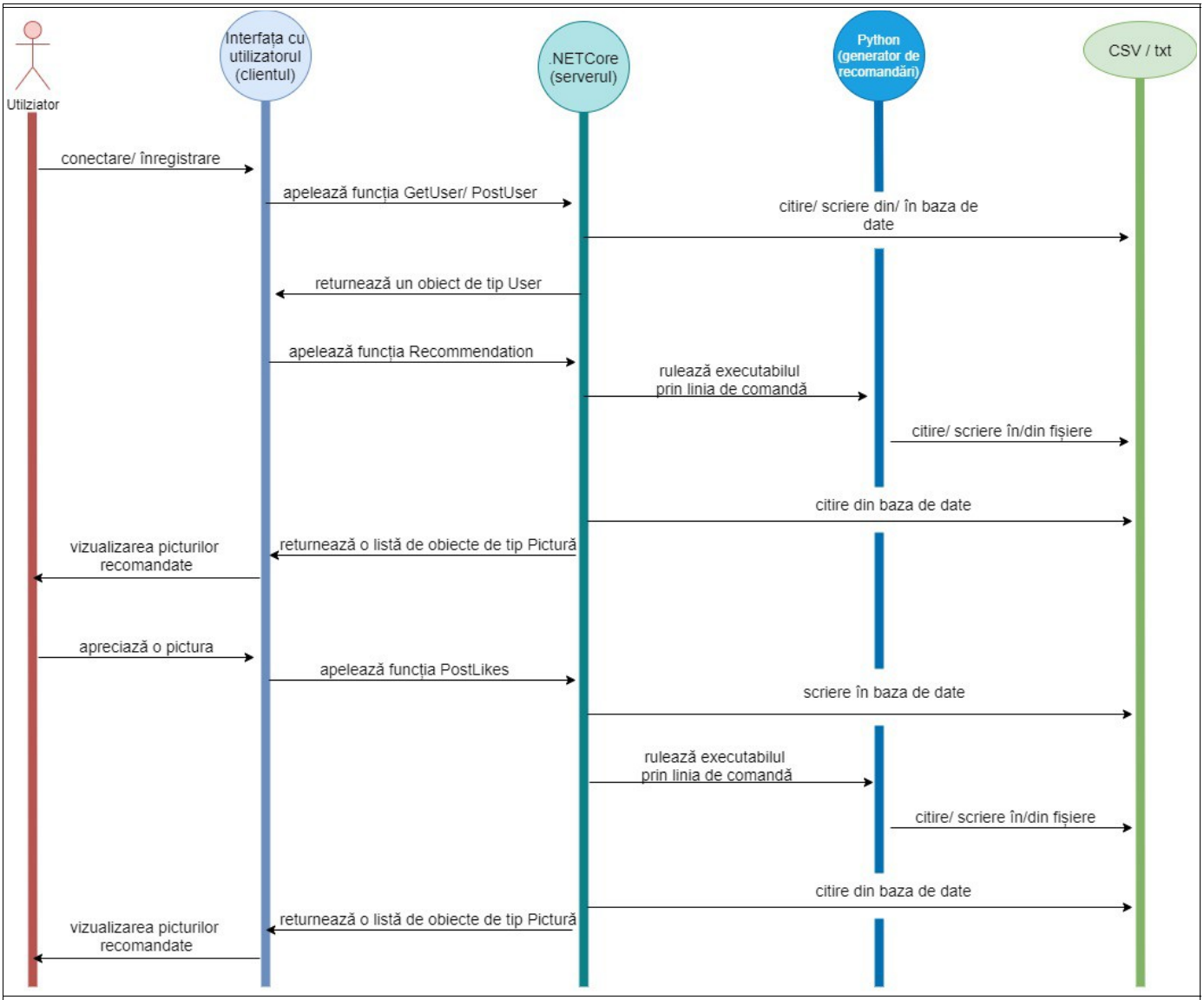



*Figura 9: Diagrama de secvențe*

În imaginea de mai sus este prezentată diagrama de secvențe. O acțiune a utilizatorului, declanșează o reacțiune din partea serverului. Acțiunile utilizatorului care declanșează o reacție în lanț sunt: înregistrarea/conectarea sau aprecierea unei picturi.

Atunci când utilizatorul se înregistrează sau se conectează, trebuie trimise datele de la client către server. Serverul verifică baza de date (și introduce un utilizator nou, dacă este cazul înregistrării), iar apoi răspunde clientului printr-un obiect care conține toate datele utilizatorului.

Una din cele mai importante atribute ale obiectului este cel care conține picturile apreciate anterior. Pe baza acestora, se generează recomandările și îi sunt comunicate clientului, acesta urmând să le afișeze.

Atunci când utilizatorul apreciază o pictură, trebuie actualizată lista picturilor recomandate. Pentru a face acest lucru, trebuie actualizat câmpul picturilor apreciate al utilizatorului conectat. Odată ce acesta este actualizat, trebuie generate noile recomandări și comunicate clientului.

Partea de client a aplicației alcătuiește interfața cu utilizatorul. Această interfața este făcută în Angular și are ca scop punerea la dispoziția utilizatorului un mediu prin care acesta poate interacționa cu baza de date, printr-un mod transparent și cat mai captivant.

Serverul, aplicația .NETCore, comunică cu clientul și cu baza de date. Comunicația cu clientul reprezintă preluarea datelor și cererilor făcute de acesta. De exemplu când un utilizator dorește să se conecteze, se trimite o cerere pentru a verifica numele utilizatului și parola acestuia. Împreună cu această cerere, trebuie trimise și datele necesare pentru a putea fi verificate.

Pentru afișarea recomandărilor, serverul prin linia de comandă, rulează fișierul scris în Python, care generează recomandările în funcție de picturile apreciate anterior, de utilizatorul conectat.

Pentru a putea rula aplicația pe orice calculator sunt necesare următoarele resurse:

-Windows 10
-Visual Code 2019
-Angular CLI 9.0.7
-Pyhton 3.6.7
-Visual Studio 2019
-.NETCore 3.0.100
-acces la baza de date
-pornirea și rularea serverului (aplicația .NETCore)
-pornirea și rularea clientului (aplicația Angular)

Clientul, interfața cu utilizatorul, conține următoarele componente: Home, Galerie, Evenimente, Login. Componenta Galerie conține 6 sub-componente: Picturi pe pânza, Picturi pe perete, Picturi pe căni sau pahare, Mărțișoare, Felicitări și Meniuri (acestea apar în funcție de ce alege utilizatorul).

Serverul constă într-o aplicație .NETCore, care se ocupă cu gestionarea datelor din și în baza de date.

Cele mai importante componente din aceasta aplicație sunt:

- fișierul Controllers, care conține ValuesConstroller.cs (face legătura cu angular prin funcțiile definite);

- fișierul Models, care conține clasele Pictura și User;
- ReadCsv.cs, care conține toate funcțiile folosite;
- startup.cs care face legătura cu angular (specifică localhost-ul clientului "http://localhost:4200" )

Clientul este format dintr-o aplicație Angular

Cele mai importante componente din aceasta aplicație sunt:

- componentele: Home, Galerie, Evenimente, Login
- fișierul „shared” care conține: servicii (pentru partajarea datelor și trimiterea datelor către server) și clasele Pictura și User.

### 4.1. Componenta Home

Această componentă conține un scurt rezumat al paginii.

Prezintă componentele galeriei. Dacă utilizatorul dorește, poate apăsa pe butonul „vezi mai multe”, va fi redirecționat în galerie și va putea vizualiza picturile din categoria aleasă.

Pe lângă componentele galeriei, sunt prezentate pe scurt evenimentele, însoțite de un buton, pe care utilizatorul îl poate apăsa, dacă dorește să citească mai mult despre acele evenimente.

Pentru utilizatorii care s-au înregistrat, în prima parte a paginii apar recomandările.

Atunci când este apăsat butonul „vezi mai multe”, este apelata funcția *Redirect_evenimente*, care primește ca parametrii calea(care poate fi galerie sau evenimente) și id-ul categoriei (dacă se dorește redirectarea către evenimente, acest id va fi 0).


*Figura 10: Apelarea funcției Redirect_evenimente*

```
Redirect_Evenimente(cale, id_categorie){
            this.cale=cale;
            this.id_categorie = id_categorie;
            this._galerieservice.nextId(this.id_categorie);
            console.log("id_cat", this.id_categorie);
            this.router.navigate([cale]);
            window.scrollTo(0, 0);
    }
```

*Corpul funcției Redirect_Evenimente*

Redirectarea se face cu ajutorul metodei „*navigate*” din clasa „*Router*”. Acesta primește ca parametru calea către pagina la care dorim redictarea. Înainte de redirectare, trebuie trimis id-ul categoriei. Acest lucru se face cu ajutorul unui serviciu, numit *GalerieService* astfel:

```
export class GalerieService{
            private id_categorie = new BehaviorSubject(0);
            sharedCategorieId = this.id_categorie.asObservable();
            constructor(){}
            nextId(id_categorie: number) {
            this.id_categorie.next(id_categorie);
            }
    }
```

*Corpul serviciului GalerieService*

În acest caz, *id_categorie* este o dată partajată. Aceasta este modificată/citită cu ajutorul metodelor:

- *asOservable*: preia valoarea de la sursă și o trimite la 1 sau mai mulți destinatari (tuturor celor care s-au subscris) . Cu alte cuvine, această metoda ajută la citirea unei date partajate.
- *nextId*: modifică data partajată (se setează o nouă valoare).

### 4.2. Componenta Galerie

Această componentă este alcătuită din 2 părți: meniul și 6 sub-componente, care au ca scop afișarea picturilor conform cu categoria aleasă din meniu.

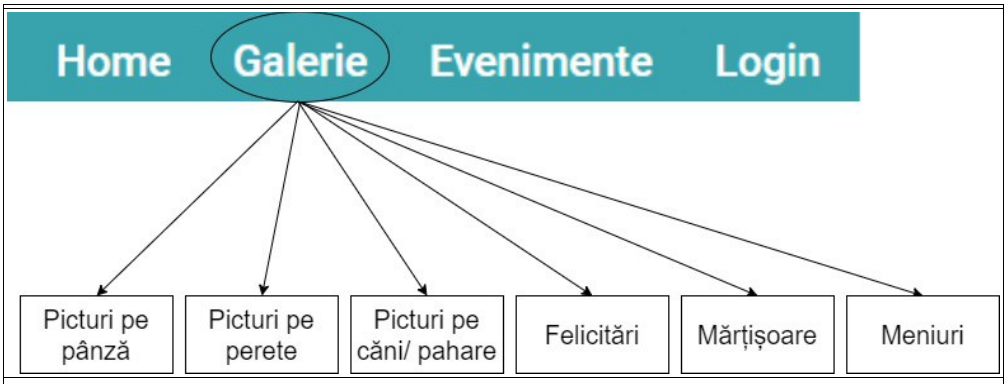



*Figura 11: Sub-componentele componentei Galerie*

Din meniu, utilizatorul poate alege ce categorie dorește sa vizualizeze. Fiecare categorie are asociat un id pe baza căruia se va afișa o sub-componentă (care va afișa toate picturile care au id-ul egal cu cel ales).


*Imaginea 1: Componenta Galerie*

Există 2 variante pentru a accesa componenta Galerie: din Home (apăsând pe butonul corespunzător unei categorii, iar odată cu redirectarea se trimite și id-ul categoriei) sau din meniul principal. În cazul în care alegem galeria din meniul principal, id-ul categoriei va fi 0.

| *codificare* | *semnificație* |
|---|---|
| 0 | nimic |
| 1 | picturi pe pânza |
| 2 | picturi pe perete |
| 3 | picturi pe căni/ pahare |
| 4 | mărțișoare |
| 5 | meniuri |

*Tabel 19: Codificarea categoriilor picturilor*

După ce id-ul categoriei a fost ales, se afișa sub-componenta corespunzătoare, care se va ocupa de afișarea picturilor care au id-ul egal cu cel ales de utilizator.

De exemplu, dacă utilizatorul alege din meniu „*Picturi pe pânză*”, aceasta corespunde id-ului 1.

```
<ng-template [ngIf]="nr_categorie==1" >
       <app-picturi-panza></app-picturi-panza>
</ng-template>
```

*Directiva structurală NgIf*

În interiorul șablonului, este verificat id-ul categoriei. Această verificare se face cu ajutorul directivei structurale „*NgIf*”, care include condiționat un șablon bazat pe valoarea unei expresii booleene. Când expresia este evaluată ca fiind adevărată, componenta care este inclusa în acest șablon va deveni vizibilă.

Componenta „picturi-pânză” trebuie să afișeze toate picturile care au id-ul categoriei egal cu id-ul ales de utilizator. Acest lucru este făcut cu ajutorul serverului (aplicația .NETCore).

Componenta „*picturi-pânză*” va apela funcția *getPictura* inclusă într-un serviciu („*Service*”) care primește ca parametru id-ul categoriei și va returna o listă de obiecte de tip Pictură (Pictură este o clasă prezentă și în angular și în .NETCore).

În serviciul *Service* avem:

```
   getPictura(id_categorie: string):Observable<any>{
        return this.http.get(this.baseUrl+'values/'+ id_categorie);
    }
```

*funcția getPictura din serviciu*

unde *baseURL* este adresa serverului *(„localhost:51476/api/”).*

Se vor apela următoarele funcții: *Get* din ValluesControler.cs, care apelează *GetPictura* din Service.cs. La rândul ei, *GetPictura* apelează funcția *Read* din ReadCSV.cs.

Activitatea serverului este încheiată cu returnarea unei liste de picturi care au id-ul egal cu id-ul categoriei alese de utilizator.

```
public List<Pictura>  GetPictura(int id_categorie)
{
     return reader.Read().Where(a => a.categorie == id_categorie).ToList();
}
```

*Funcția GetPictura*

```
public List<Pictura> Read()
    {
        string path = "C:\\Users\\Andreea\\facultate\\licenta\\lup\\BD\\Picturi.csv";
        List<Pictura> pictura = new List<Pictura>();
        string[] lines = System.IO.File.ReadAllLines(path);

        for (int i = 1; i < lines.Length; i++) //linii; sare peste capul tabelului
        {
            string[] fields = lines[i].Split('#') ;

            pictura.Add(new Pictura
            {
                id_pictura = Int32.Parse(fields[0]),
                cale_poza = fields[1],
                denumire = fields[2],
                categorie = Int32.Parse(fields[3]),
                taguri = fields[4],
                disponibilitate = Int32.Parse(fields[5]),
                pret = Int32.Parse(fields[6]),
                dimensiune = fields[7],
                descriere = fields[8]

            });
        }
        return pictura;
    }
```

*Funcția Read*

Pentru a prelua datele de la server, clientul folosește o lambda expresie pentru a prelua rezultatul și a-l salva într-o lista cu obiecte de tip Pictura.

```
constructor(private service:Service, private _dataservice:DataService) {
  this.service.getPictura('1').subscribe(res => {
    this.pictura = res;
  },
    err => {
      console.log(err);
    })
 }
```

*Preluarea listei de obiecte*

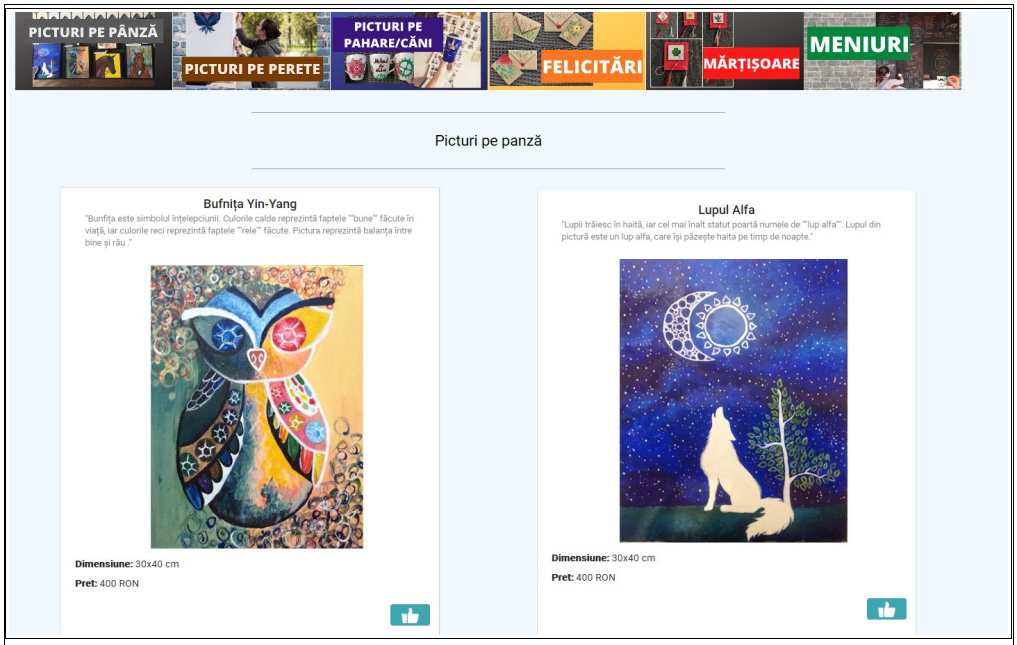
Pentru aranjarea picturilor am folosit „*mat-grid-list”*, o componentă din Angular Material, care ajută împărțirea spațiului în 2 coloane (liniile depind de cate picturi sunt).

```
<mat-grid-list cols="2"  [gutterSize]="'10px'">
  <mat-grid-tile *ngFor="let pictura of pictura">
      <mat-card>
          <mat-card-header>
            <mat-card-title >{{pictura.denumire}}</mat-card-title>
            <mat-card-subtitle>{{pictura.descriere}}</mat-card-subtitle>
          </mat-card-header>

          <img mat-card-image src={{pictura.cale_poza}} alt="">

          <mat-card-content>
              <p> <b>Dimensiune: </b> {{pictura.dimensiune}} cm </p>
              <p> <b>Pret: </b> {{pictura.pret}} RON </p>
          </mat-card-content>

          <mat-card-actions class="dreapta">
            <button mat-button (click)="Like(pictura.id_pictura)">
              <img src="../../assets/images/like.png">
            </button>
          </mat-card-actions>

          <mat-card-footer>
            <p></p>
          </mat-card-footer>
      </mat-card>
  </mat-grid-tile>
</mat-grid-list>
```

*Afișarea obiectelor de tip Pictura*

Pentru afișarea picturilor am folosit „*mat-card*”, care este tot o componentă din Angular Material. Aceasta afișează picturile sub forma unor carduri, care au atributele: titlu, subtitlu, imagine, conținut și buton. Titlul conține denumirea picturii, subtitlul este alcătuit din descrierea acesteia, imaginea conține poza picturii, conținutul este compus din detalii despre aceasta (dimensiune și preț), iar butonul reprezintă aprecierea picturii.

În imaginea de mai jos sunt prezentate primele 2 picturi care apar atunci când este ales din meniu opțiunea „Picturi pe pânză”:




*Imaginea 2: Sub-componenta "Picturi pe pânză"*

### 4.3. Componenta Evenimente

Această componentă conține descrierea evenimentelor care au avut loc. Împreuna cu această descriere, sunt atașate și poze de la eveniment.

Componenta aceasta are rolul de a informa utilizatorul cu privire la evenimentele care au avut loc.

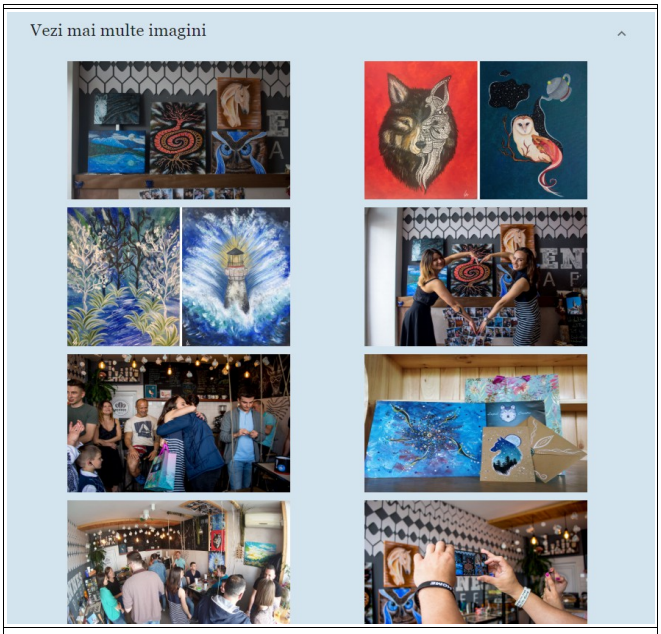

*Imaginea 3: Al doilea eveniment*

Pe lângă informarea utilizatorului în legătura cu evenimentele care au avut loc deja, această componenta va afișa detalii despre următoarele evenimente, astfel încât dacă exista utilizatori care doresc să participe, să fie informați despre eveniment.

Utilizatorul are posibilitatea de a vedea mai multe imagini apăsând pe butonul „vezi mai multe imagini”.

Odată apăsat, acesta va extinde o galerie creată pentru acel eveniment.

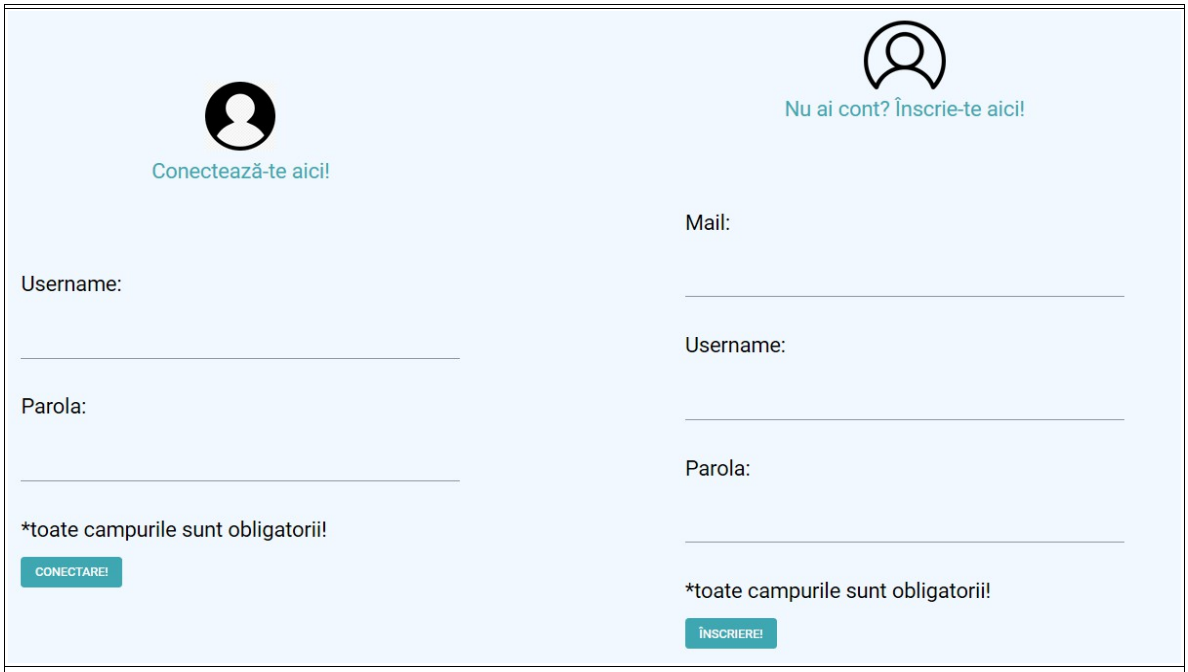
Aceasta galerie are rolul de a îndruma utilizatorul către picturile care au fost apreciate de peroanele care au participat la expoziție și le-au văzut în realitate.




*Imaginea 4: Galeria corespunzătoare celui de al doilea eveniment*

### 4.4. Componenta Login

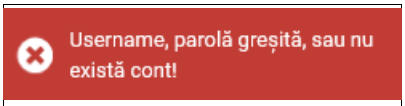
Această componentă pune la dispoziția utilizatorului 2 variante: înscriere (pentru cei care au cont) sau înregistrare (pentru cei care doresc sa își creeze un cont).




*Imaginea 5: Componenta Login*

Dacă formularul nu este completat corect, vor apărea erori, care atenționează utilizatorul că trebuie sa acorde o atenție sporită completării formularului:

- dacă nu sunt completate toate câmpurile și utilizatorul încearcă să se conecteze sau să se înscrie, va apărea un mesaj de eroare care semnalează utilizatorului că toate câmpurile trebuie completate;
- dacă este greșit numele sau parola, va apărea o eroare, atenționând utilizatorul;




*Imaginea 6: Completarea greșită*

- dacă la înregistrare utilizatorul dorește să-și facă un cont cu un nume deja existent în baza de date, utilizatorul va fi atenționat și va fi nevoit sa aleagă alt nume;


*Imaginea 7: Același nume*

Aceste atenționări sunt necesare, deoarece la apăsarea butonului pentru înscriere sau conectare se va apela o funcție care trimite datele către server. Fără a fi completate toate câmpurile, s-ar produce erori la încercarea de a citi sau a introduce un nou utilizator în baza de date.

După completarea tuturor câmpurilor, aceeași funcție este folosită pentru transmite datele către server. La apăsarea butonului de conectare sau înscriere este apelată funcția *Login* ,care primește ca și parametrii numele utilizatorului, parola și mailul (în cazul butonului de înregistrare, câmpul *mail* va fi null).

Asemănător cu preluarea id-ului categoriei de la galerie, funcția Login va apela funcția *getUser* inclusă într-un serviciu („*Service*”) care primește ca parametru numele utilizatorului, parola și mailul. Aceasta va returna un obiect de tip User (User este o clasa prezenta și în aplicația client și în aplicația server).

Diferența dintre înregistrare și conectare se face prin câmpul „mail” astfel: în server se apelează funcția *LoginOrRegister*. Aceasta verifică câmpul „mail”. Dacă acesta este null, se apelează *GetUser* (însemnând ca este este o conectare), altfel se va apela *PostUser* (reprezentând o operație de înregistrare).

Pentru conectare se vor trimite numele utilizatorului și parola către următoarele funcții:

1. *GetUser* din Service, care va apela la funcția *ReadUser* din ReadCsv.
2. Funcția *ReadUser* verifică CSV-ul corespunzător utilizatorilor. Aceasta verifică dacă numele utilizatorului și parola sunt corecte. Dacă sunt corecte, se populează obiectul user cu câmpurile corespunzătoare din baza de date, iar apoi returnat acest obiect. În caz contrar se va returna tot un obiect de tip User, dar toate câmpurile vor fi nule. Acesta indica o eroare în verificarea corectitudinii datelor și înseamna ca numele utilizatorului sau parola sunt greșite.

Pentru înregistrare se vor trimite numele utilizatorului, parola și mailul către următoarele funcții:

1. *PostUser* din service, care va apela la rândul ei funcția *PostUser* din ReadCsv.
2. Aceasta funcție introduce o noua înregistrare în CSV-ul corespunzător utilizatorilor. La sfârșit, returnează un obiect de tip User.

Am avut nevoie ca aceste funcții să returneze un obiect de tip User, deoarece fiecare acțiune pe care o va face utilizatorul de acum înainte, va fi salvată.

Totodată, majoritatea componentelor au nevoie de id-ul utilizatorului. Acest lucru a fost realizat printr-un serviciu:

```
export class DataService{

    private id_loggedin = new BehaviorSubject(0);
    sharedId = this.id_loggedin.asObservable();

    constructor(){}

    nextId(id_loggedin: number) {
        this.id_loggedin.next(id_loggedin);
    }

}
```

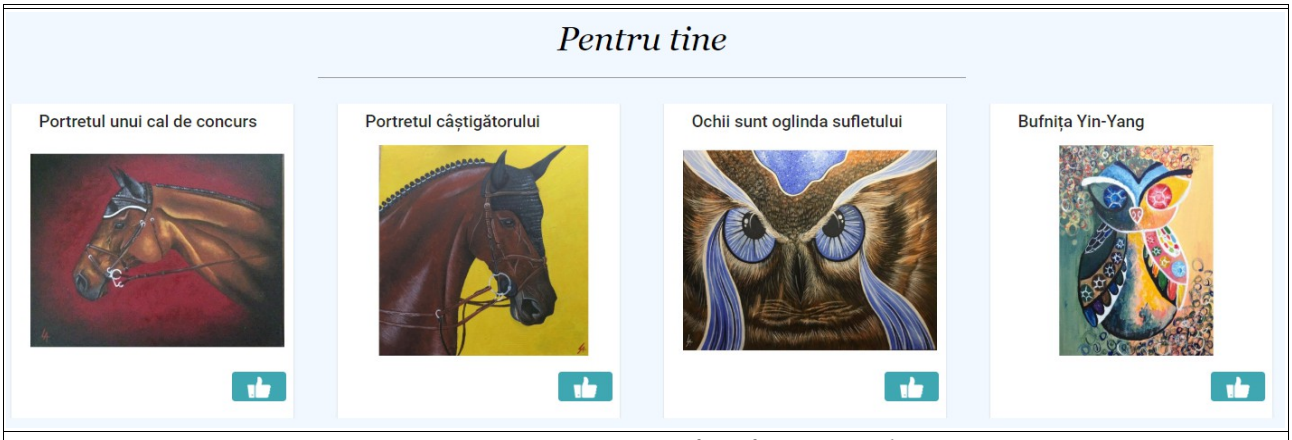
*Serviciul DataService*

În acest caz, *id_loggedin* este o dată partajată. Aceasta este modificată/citită cu ajutorul metodelor:

- *asOservable*: preia valoarea de la sursă și o trimite la 1 sau mai mulți destinatari (tuturor celor care s-au subscris) . Cu alte cuvine, această metoda ajută la citirea unei date partajate.
- *nextId*: modifică data partajată (se setează o nouă valoare).

Modificarea acestei valorii se face doar în momentul înregistrării, conectării sau deconectării. Inițial, aceasta valoare este 0.

Odată conectat utilizatorul, id-ul este setat cu valoarea corespunzătoare numelui utilizatorului, apoi este redirecționat la pagina de început (Home). Aici apare prima schimbare, față de vizitator. Apare o porțiune numită „Pentru tine”. În această porțiune apar picturi recomandate pentru utilizator. Aceste picturi sunt similare cu cele pe care le-a apreciat anterior. În cazul în care utilizatorul este nou, sau nu a apreciat nici o pictură, vor fi recomandate 2 picturi, acestea urmând să fie înlocuite, atunci când utilizatorul apreciază una sau mai multe picturi.




*Imaginea 8: Porțiunea specifică fiecărui utilizator*

În imaginea anterioară, utilizatorul conectat a apreciat anterior 2 picturi: una cu portretul unui cal („Portretul lui Sara-Z”) și o pictură cu o bufniță („The night's watch”). Pe baza atributelor picturilor apreciate, au fost generate aceste recomandări, care sunt cele mai similare cu cele 2 picturi apreciate.

Următoarea schimbare care apare pentru utilizatori este că pot aprecia orice pictură. Vizitatorii nu puteau face acest lucru. Chiar dacă butonul este vizibil și pentru cei care nu sunt conectați, dacă aceștia apasă pe butonul pentru apreciere, vor fi atenționați că nu sunt conectați și operațiunea pentru aprecierea picturilor nu merge mai departe. La nivelul vizitatorilor este inutil să fie permisă aprecierea picturilor , pentru că neavând cont, nu am putea reține picturile apreciate anterior, așadar recomandarea altor picturi, ai fi incorectă și inutilă.

O altă schimbare care apare pentru utilizatori este în meniu: butonul de „Login” se schimbă în „Logout”. Cu ajutorul variabilei partajate despre care am vorbit mai devreme (*id_loggedin*), am realizat acest lucru. Dacă nu avem nici un utilizator conectat, valoarea variabilei *id_loggedin* este

egală cu 0. În acest caz, textul afișat în meniu este „Login”. După înregistrare/conectare, odată cu schimbarea id-ului, se schimbă și textul din meniu, acesta devenind „Logout”.

La apăsarea acestui buton („Login”/”Logout”), este verificat textul butonului. Dacă acesta este „Login”, atunci utilizatorul este redirecționat către pagina de înregistrare/conectare.

Dacă butonul are textul „Logout”, se resetează id-ul userului (redevine 0).

### 4.5. Sistemul de recomandare

Sistemul de recomandări este realizat cu ajutorul limbajului Python. Acesta este executat din linia de comandă, prin server, pentru a actualiza recomandările la fiecare conectare a unui utilizator, sau atunci când un utilizator (deja conectat) a apreciat alte picturi și dorește sa vadă recomandările.

De fiacre dată când un utilizator se conectează, aplicația scrisă în Python trebuie rulată, pentru a se actualiza fișierele din care se citesc picturile apreciate anterior de utilizator și fișierul în care sunt scrise picturile pe care programul le va recomanda.

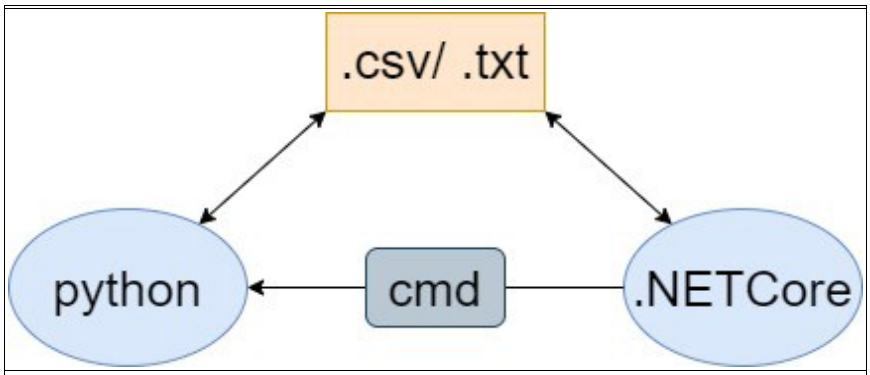



*Figura 12: Structura funcționării sistemului de recomandări*

Pe lângă fișierele care alcătuiesc baza de date, alte 3 fișiere importante (în afara celor care alcătuiesc baza de date) fără de care generarea recomandărilor nu ar fi posibilă sunt: picturi_apreciate.csv, lista_picturui_recomandate.csv, picturi_recomandate.txt. Aceste fișiere se actualizează la fiecare conectare a unui utilizator și atunci când utilizatorul apreciază una sau mai multe picturi.

- picturi_apreciate.csv: acest fișier este rescris atunci când se conectează un utilizator. Fișierul conține picturile anterior apreciate de utilizatorul care tocmai s-a conectat.
- lista_picturi_recomandate.csv: în acest fișier sunt scrise recomandările făcute pe

baza picturilor apreciate, care sunt salvate în fișier prezentat anterior.

- picturi_recomandate.txt.: în acest fișier sunt recomandările finale, fără dubluri.

Pentru a crea recomandările au fost urmați 3 pași:

1. Scrierea în fișier a id-urilor picturilor apreciate

   Pentru utilizatorul conectat, se extrage din CSV-ul corespunzător utilizatorilor, câmpul „picturi apreciate”.

2. Generarea recomandărilor

   Din server se execută fișierul Pyhton, prin linie de comandă.

```
public void run_python()
{
            Process cmd = new Process();
            cmd.StartInfo.FileName = "cmd.exe";
            cmd.StartInfo.RedirectStandardInput = true;
            cmd.StartInfo.RedirectStandardOutput = true;
            cmd.StartInfo.CreateNoWindow = true;
            cmd.StartInfo.UseShellExecute = false;
            cmd.Start();

            cmd.StandardInput.WriteLine("conda activate tensorflow");
           cmd.StandardInput.WriteLine("python"+
                      "C:\\Users\\Andreea\\facultate\\licenta\\rec_sys\\RECOMANDARI.py");

            cmd.StandardInput.Flush();
            cmd.StandardInput.Close();
            cmd.WaitForExit();
            Console.WriteLine(cmd.StandardOutput.ReadToEnd());
}
```

*Rularea fișierului Python din aplicația client*

Execuția programului care generează recomandările începe cu citirea CSV-ului corespunzător picturilor apreciate de un anumit utilizator, introducând datele într-o matrice. Apoi se construiește o nouă coloană formată din caracteristicile pe baza cărora se vor face recomandările (în cazul nostru caracteristicile sunt: categoria din care face parte pictura și atributele acesteia). După ce a fost creată noua coloană, se va calcula similaritatea fiecărei picturi față de cele deja apreciate (în funcție de caracteristicile alese mai devreme) și vor fi introduse într-o listă. După sortarea listei(în funcție de similaritate) în ordine descrescătoare, sunt alese primele 2 (cu cea mai mare similaritate) și vor fi scrie în fișierul lista_picturi_recomandate.csv sub forma: [(id_pictura1, similaritate1), (id_pictura1, similaritate1), ...].

```
cv = CountVectorizer()
count_matrix = cv.fit_transform(df["combined_features"])
cosine_sim = cosine_similarity(count_matrix)

likedPaintingsMatrix = pd.read_csv(r'C:\Users\Andreea\facultate\licenta\
    sistem_recomandari\picturi_apreciate.csv', sep=',',header=None)
picturi = pd.read_csv(r'C:\Users\Andreea\facultate\licenta\lup\BD\Picturi.csv',
          sep='#',
          names=['id_pictura','cale','denumire','categorie','taguri','disponibil'
                 ,'pret','dimeniune','descriere'])

id_likedPaintings = likedPaintingsMatrix.values[0]
similar_paintings_list=[]

for id_p in id_likedPaintings:
    liked_painting=get_title_from_index(id_p)
    painting_index = get_index_from_title(liked_painting)
    similar_painting =  list(enumerate(cosine_sim[painting_index]))
    sorted_similar_paintings = sorted(similar_painting,key=lambda x:x[1],reverse=True)
    similar_paintings_list.append(sorted_similar_paintings[1:3])

print(similar_paintings_list)

with open(r'C:\Users\Andreea\facultate\licenta\sistem_recomandari\
    lista_picturi_recomandate.txt', 'w') as f:
    for item in similar_paintings_list:
        f.write("%s\n" % item)
```

*Recomandarea picturilor*

Unde:

- funcția *CountVectorizer* este folosită pentru a converti o colecție de documente text într-o matrice de numere.

- funcția *cosine_similatity* este folosită pentru a determina cât de similare sunt șirurile de atribute. Din punct de vedere matematic, aceasta măsoară cosinusul unghiului dintre 2 vectori proiectați într-un spațiu multidimensional. În acest context, vectorii reprezintă tablouri care conțin numărul de cuvinte ale șirurilor de atribute . Cu cât unghiul este mai mic, cu atât similaritatea crește.

Aceste două funcții fac parte din modulul Python numit „sklearn”. Acest modul integrează algoritmii clasici de învățare automată.

3. Citirea recomandărilor

Primul pas din această etapă constă în citirea fișierului în care a fost scrisă lista cu picturile recomandate și similaritatea. Se extrage din acea listă doar id-ul picturilor, se elimină dublurile (dacă există), iar apoi se scriu în fișierul picturi_recomandate.txt.

Următorul pas constă în citirea fișierului care conține recomandările, construirea obiectelor de tip Pictura (în concordanta cu id-urile citite) și introducerea lor într-o listă.

Ultimul pas constă trimiterea de la server către client a acestei liste de obiecte și afișarea picturilor recomandate.

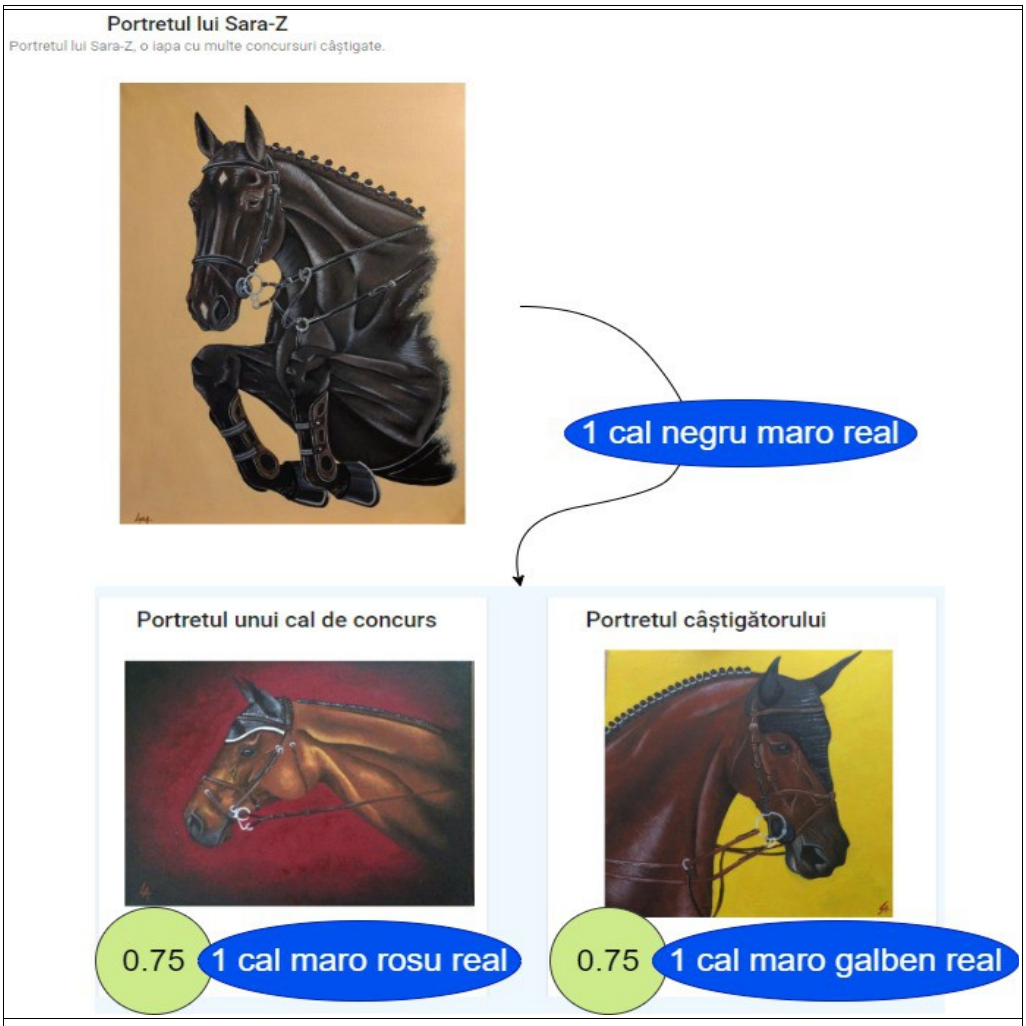



*Figura 13: Recomandarea picturilor*

În figura de mai sus am exemplificat cum funcționează sistemul de recomandări.

După ce s-a conectat, utilizatorul a apreciat pictura „Portretul lui Sara-Z”, care face parte din categoria 1, adică picturi pe pânză și are asociate atributele: cal, negru, maro, real. În urma acestor caracteristici, au fost recomandate alte 2 picturi, cu o similaritate de 0.75.

În acest caz, similaritatea celor 2 picturi este egală, dar acest lucru se întâmpla rar. Aceste picturi recomandate au în comun categoria și caracteristicile: cal, maro și real, iar diferența între cele 3, o fac cuvintele: negru, rosu și galben.

# 5. Rezultate, dezvoltări ulterioare și concluzii

## 5.1. Rezultate

Scopul aplicației a fost atins: generarea unor picturi similare cu picturile apreciate anterior de un anumit utilizator.

Interfața cu utilizatorul a îndeplinit sarcinile:

- permite fiecărui utilizator să se conecteze cu un cont unic sau să creeze un cont nou;
- afișarea picturilor corespunzătoare categoriei din care fac parte;
- permite utilizatorilor să vizualizeze evenimentele care au avut loc;
- afișarea recomandărilor.

Serverul a îndeplinit sarcinile:

- gestionarea datelor  din și în baza de date;
- recepția și transmiterea datelor către interfața cu utilizatorul;
- rularea aplicației care generează recomandările, prin linia de comandă.

Sistemul de recomandări a îndeplinit sarcinile:

- citirea picturilor apreciate de un utilizator;
- calcularea similarității picturilor, în funcție de atributele picturilor apreciate
- generarea recomandărilor (pentru fiecare pictura apreciată, se aleg alte 2 cele mai similare picturi)

### 5.2. Dezvoltări ulterioare

- Atunci când un utilizator se conectează, acesta să primească un mail pentru confirmare;
- Să se poată conecta mai mulți utilizatori simultan;
- Atunci când apare un eveniment nou, toți utilizatorii să primească un mail cu informațiile legate despre eveniment, împreună cu o invitație;
- Posibilitatea cumpărării unei picturi;
- Atunci când un utilizator crează un cont nou, acesta sa treacă printr-un chestionar în care să aprecieze picturi (pentru a elimina recomandările inițiale, care nu sunt personalizate);
- Posibilitatea plasării unei comenzi pentru o pictură (crearea picturii să reiasă în urma completării unui chestionar);
- Posibilitatea utilizatorului de a lăsa un comentariu la fiecare pictură, dar și la evenimente;
- Sistemul de recomandări (fișierul Python) sa fie apelat printr-un serviciu direct în loc de a fi apelat prin linia de comandă;
- Introducerea învățării automate în sistemul de recomandări pentru a avea recomandări mai concrete;

### 5.3. Concluzii

Realizând această aplicație, am învățat cum se construiește un sistem de recomandări, aplicând ce am asimilat pe parcursul celor 4 ani de facultate, dar și învățând multe lucruri noi. Am învățat cum se folosesc conceptele învățării automate pentru a crea un sistem de recomandări, m-am familiarizat cu limbajul Python, am învățat cum să construiesc o aplicație Angular și nu în ultimul rând, am învățat cum să fac legătura între 3 aplicații diferite, astfel încât aplicaţia să funcționeze cum m-am dorit.

## 7. Bibliografie


1. https://dotnet.microsoft.com/apps/aspnet
2. https://dotnet.microsoft.com/learn/aspnet/what-is-aspnet-core
3. https://developers.google.com/machine-learning/crash-course/exercises
4. https://angular.io/docs
5. https://material.angular.io/components/categories
6. http://d2l.ai/chapter_recommender-systems/recsys-intro.html
7. https://tryolabs.com/blog/introduction-to-recommender-systems/
8. https://www.edureka.co/blog/keras-vs-tensorflow-vs-pytorch/
9. https://towardsdatascience.com/building-a-book-recommendation-system-using-keras-1fba34180699
10. https://medium.com/@jdwittenauer/deep-learning-with-keras-recommender-systems-e7b99cb29929