UNIVERSITATEA ”ALEXANDRU-IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

Sistem software pentru gestiunea punctajelor

propusă de

Bodnar Ioan-Ciprian

Sesiunea: iulie, 2019

Coordonator științific

Lect. Dr. Frăsinaru Cristian

UNIVERSITATEA ”ALEXANDRU-IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

Sistem software pentru gestiunea punctajelor

Bodnar Ioan-Ciprian

Sesiunea: iulie, 2019

Coordonator științific

Lect. Dr. Frăsinaru Cristian

Avizat,

Îndrumător lucrare de licență,

Lect. Dr. Frăsinaru Cristian.

Data: ..... Semnătura: .....

Declarație privind originalitatea conținutului lucrării de licență

Subsemnatul **Bodnar Ioan-Ciprian** domiciliat în **România, jud. Suceava, sat Straja nr. 580**, născut la data de **06 iunie 1997**, identificat prin CNP **1970608330524**, absolvent al Facultății de informatică, **Facultatea de informatică** specializarea **informatică**, promoția 2019, declar pe propria răspundere cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art. 143 al. 4 și 5 referitoare la plagiat, că lucrarea de licență cu titlul **Sistem software pentru gestiunea punctajelor** elaborată sub îndrumarea domnului **Lect. Dr. Frăsinaru Cristian**, pe care urmează să o susțin în fața comisiei este originală, îmi aparține și îmi asum conținutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată prin orice modalitate legală pentru confirmarea originalității, consimțind inclusiv la introducerea conținutului ei într-o bază de date în acest scop.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări științifice în vederea facilitării falsificării de către cumpărător a calității de autor al unei lucrări de licență, de diplomă sau de disertație și în acest sens, declar pe proprie răspundere că lucrarea de față nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Data: ..... Semnătura: .....

Declarație de consimțământ

Prin prezenta declar că sunt de acord ca lucrarea de licență cu titlul **Sistem software pentru gestiunea punctajelor**, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test, etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de informatică.

De asemenea, sunt de acord ca Facultatea de informatică de la Universitatea ”Alexandru-Ioan Cuza” din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Absolvent **Bodnar Ioan-Ciprian**

Data: ..... Semnătura: .....

Cuprins

Introducere

În Romania, dar și nu numai, din princina birocrației, nevoia de automatism este în mare căutare. În multe cazuri ne dorim să modificăm un text vechi tipărit sau un text scris de mână care nu are o formă editabilă, iar singura soluție este rescrierea acestuia într-o formă editabilă, ceea ce este un proces destul de costisitor din punct de vedere al timpului. Soluția acestei probleme este rezolvată de către un recunoscător de caractere printate sau scrise, prescurtat OCR. Deci un utilizator al unui acest tip de aplicație trimite un input, de obicei sub forma unui text printat sau scris, iar OCR-ul întoarce varianta acestuia editabilă.

Scrisul uman de mână este dificil de înțeles sau de recunoscut de către un sistem automat deoarece fiecare persoană are stilul propriu de scris.

Motivație

De asemenea, acest tip de aplicație își are scopul și în cadrul facultății noastre. Prin prisma faptului că în cadrul prelegerilor de curs, dar și în cadrul activităților de laborator, respectiv seminar, adesea se cere monitorizarea prezenței studenților printr-o listă a acestora pe o coală de hârtie. Această practică este greu de gestionat de către cadrele didactice. De aceea o metoda mai automata de a realiza acest lucru este dezvoltarea unui aplicații web ce permite unui cadru didactic să încarce o foaie de prezență scanată, cu o anumită construcție tabelară în funcție de scopul ei (curs sau seminar/laborator).

Motivația pentru a alege o astfel de abordare a problemei este aceea că recunoașterea caracterelor este în atenția cercetătorilor din pricina folosirii acestei funcționalități în diferite arii de aplicare. Conversia caracterelor scrise de mână mai este importantă și în recunoașterea relatărilor istorice, de exemplu manuscrise, într-o formă editabilă.

Aplicații similiare

Din punct de vedere al unor aplicații asemănătoare, ne putem lega doar de partea din aplicație ce este bazată pe recunoașterea textelor scrise de mână. Unul dintre cele mai folosite sisteme de acest tip este Tesseract, ce este o aplicație gratuită, dezvoltată sub o licență Apache și a cărui sponsor major este Google. Tesseract însă nu a fost conceput pentru caractere scrise de mână cum este cazul de față, acestuia are nevoie de o antrenare cu astfel de caractere.

CuneiForm Cognitive este o alt sistem de tip OCR dezvoltat de către compania rusă Cognitive Technologies. Acesta acceptă un număr de 28 de limbi, printre care și cea română.

Tehnologiile folosite

Dupa mulți ani de la introducerea lui, REST a devenit una dintre cele mai importante tehnologi pentru aplicațiile web. Importanța acestui este apreciată că va crește în continuare astfel încat toate tehnologiile actuale migrează spre o implementare de tip API. Fiecare limbaj de programare include acum un framework pentru construcția unui serviciu de tip REST.

Plecând spre dorința de a face aplicația cât mai portabilă, s-a decis construirea unei aplicații web cu o arhitecura REST, aceasta prezentând avantajul accesibilității de oriunde și de pe diverse dispozitive, nefiind specifică unui singur sistem de operare, spre deosebire de o aplicație desktop, iar un alt avantaj este dat atât atractivitate din punct de vedere al aspectului, cât și practic, minimalismul unei astfel tip de aplicație.

Deoarece aplicația este de natură web, soluția noastră va fi implementată cu ajutorul tehnologiei React. React este o librărie realizată de către Facebook în limbajul JavaScript folosită pentru a construi interfețe pentru aplicațiile web. Acestea sunt alcătuite din componente încapsulate ce își gestionează singure starea. Deoarece logica acestor componente este scrisă în limbajul JavaScript, înlocuind astfel șabloanele clasice, este permisă transmiterea unui volum mare de informație cu mare ușurință de la o componentă la alta în cadrul aplicației, astfel gestionarea stării aplicației se realizează mult mai ușor.

Pe langă faptul că încă este cel mai popular limbaj de programare, Java are o serie de avantaje pentru a servi ca și server pentru orice aplicație web. Java are o gama foarte mare de librării open-source, aici putem include și principala librărie folosită și

în aplicație și care servește procesării imaginilor, OpenCV. Un alt aspect important ce face Java o alegere de implemetare a unui server este managementul de memorie cu ajutorul garbage collection și că este independent de platformă.

Ca și server ajutator se folosește Python doar pentru partea clsificatorului compus din rețeaua neuronală care a fost construită cu ajutorul framework-ului Tensorflow și al API-ului Keras.

Tensorflow este un framework open source dezvoltat de Google și folosit pentru calcule numerice. Tensorflow oferă suport pentru învățare automată și deep learning. Folosind diverse toolkituri puse la dispoziție, utilizatorii pot construi modele de învățare automată la nivele diferite de abstractizare, de la modele construite pe baza unor serii de operații matematice la API-uri de nivel înalt pentru construirea de arhitecturi întregi prerealizate, cum ar fi rețele neuronale sau regresori liniari.

Keras reprezintă un API de nivel foarte înalt dedicat rețelelor neurale, scris în Python și care poate rula pe mai multe platforme de învățare automată cum ar fi TensorFlow, CNTK sau Theano. A fost dezvoltat pentru a abstractiza la maximum procesul de creare a unei rețele neurale, având ca scop atât experimentarea rapidă, cât și construcția într-un mod facil a unei arhitecturi solide. Keras poate fi văzut ca o interfață simplă, consistentă, folosită pentru cazurile uzuale, ce se pliază peste un back-end optimizat de deep learning și învățare automată. Este extrem de modular, permițând reutilizarea de blocuri configurabile, dar și extensibil, utilizatorii având posibilitatea de a crea noi straturi, funcții de cost sau arhitecturi pentru idei de cercetare.

Specificații funcționale

Descrierea problemei

Scopul aplicației, **Sistem software pentru gestiunea punctajelor**, este acela de a veni în ajutorul profesorilor și studenților. Aplicația este una de tip web, bazat pe un serviciu REST api, în care se diferențeză tipul de utilizator.

Un serviciu web de tip REST este de fapt o arhitectură REST bazat pe un serviciu web. Într-o arhitectură REST totul este o resursă, iar servicile sunt scalabile, mentenabile și foarte des folosite pentru a crea un API pentru aplicații web.

Aplicația încearcă folosirea unui recunoscător de caractere ce sunt scrise manual. Ea deservește la recunoașterea numelor și punctajele studenților dintr-o coală de hartie ce are rol de foaie de prezență sub o forma tabelară. Singura restricție este acea că literele trebuie să fie de tipar cu spațiere între ele.

Prezenta la cursul Programare avansata

	Nume	Grupa
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
20		
21		
22		
23		
24		
25		
26		
27		
28		
29		
30		
31		
32		
33		

Prezenta si activitate laborator Programare Avansata Grupa Data

Nr.	Nume	Gr.	this	prev	Bonus / Obs.
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					
22					
23					
24					
25					
26					
27					
28					
29					
30					
31					
32					

this = Activitatea din timpul laboratorului curent
prev = Activitatea propusa pentru laboratorul din saptamana precedenta

În partea stângă este un tabel standard pentru prezențele de curs, iar în partea

dreaptă este tabelul standard pentru prezențele și punctajele din cadrul unui laborator. Diferențierea dintre cele doua tipuri de tabele este făcută cu ajutorul numarului de coloane în cadrul aplicației.

Fluxuri de lucru

Toate fluxurile de intrare sunt date de către utilizator în modulul de frontend, ce interacționează direct cu utilizatorul.

Ca primă acțiune la care este supus utilizatorul, este acea de a introduce email-ul si parola acestuia din cadrul facultății. În acest moment aplicația deosebește daca utilizatorul aparține facultății și mai mult dacă acesta este student sau profesor.

Cazul în care un profesorul intră în contul personal al aplicației pe baza adresei de email a facultății, acesta selectează foaia de prezență. Foaia se procesează și este întors un tabel având forma celui de pe foaie, în care fiecare celulă poate să fie editată de către profesor daca vor apărea greșeli în recunoaștere a caracterelor. Acestă modalitate de a monitoriza punctajele și prezențele este mult mai eficientă și servește academic.

Dupa ce profesorul hotărăște că rezultatul obținut de către algoritmul OCR corespunde cu ce se așteaptă acesta, acesta poate opta ca datele sa fie salvate într-o bază de date care monitorizează prezențele și punctele la o anumită materie.

Pe lângă funcționalitatea oferită pentru un profesor, s-a optat pentru oferirea și unei statistici pentru studenți, unde aceștia își pot vizualiza prezențele în timp real la toate materiile din acel semestru fără a fi nevoit să acceseze paginile web ale materiilor respective.

Se presupune că aplicația folosește baza de date a facultății, deci accesul în contul fiecărui utilizator se realizează cu ajutorul emailului si a parolei deja existente.

Elemente de originalitate

Elementele de originalitate în cadrul aplicației sunt bazate pe tehnologiile care stau la baza construirii aplicației, dar și librăriile folosite în dezvoltarea acestei (OpenCV, Google Vision, Keras, Tensorflow, Reactjs). În plus, algoritmul de segmentare și metodele cu care trăsăturile unui caracter sunt extrase se pot adăga ca și elemente de originalitate.

Arhitectura proiectului

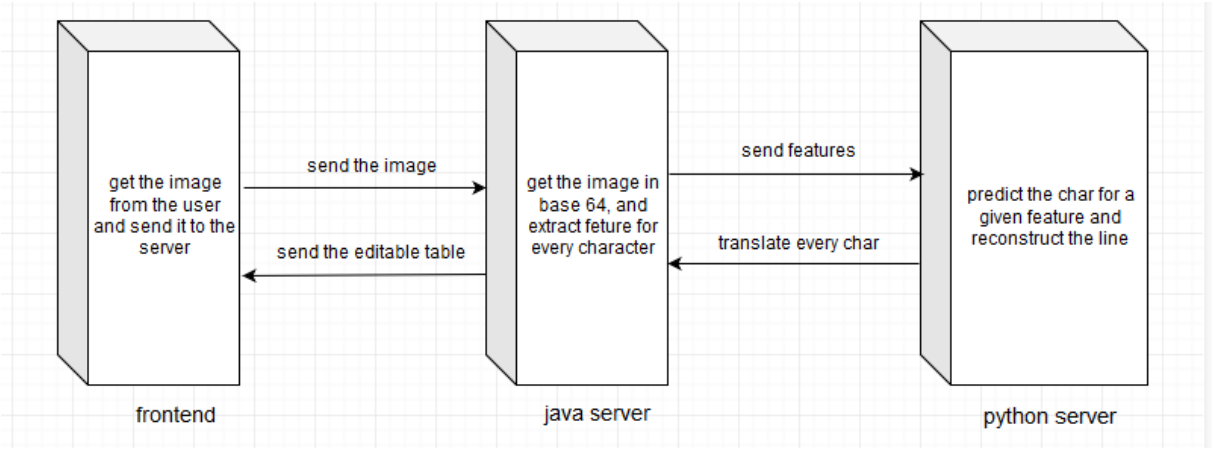


Fig 1. comunicarea dintre principalele servicii

Arhitectura proiectului este bazată pe schimbul de informații între trei componente ce reprezintă și serviciile aplicației ce alcătuiesc aplicația propriu zisă. Clientul trimite către serverul de java imaginea, acesta realizează parsarea și segmentarea, iar acesta trimite înspre recunoaștere către serverul de python. Ca și răspunsuri, direcția este inversă. Serverul de python trimite cate o linie din tabel, transpusă în text editabil, iar serverul de java transmite către frontend toate liniile primte și stocate într-o singură unitate, pentru a fi parcurse și afișate.

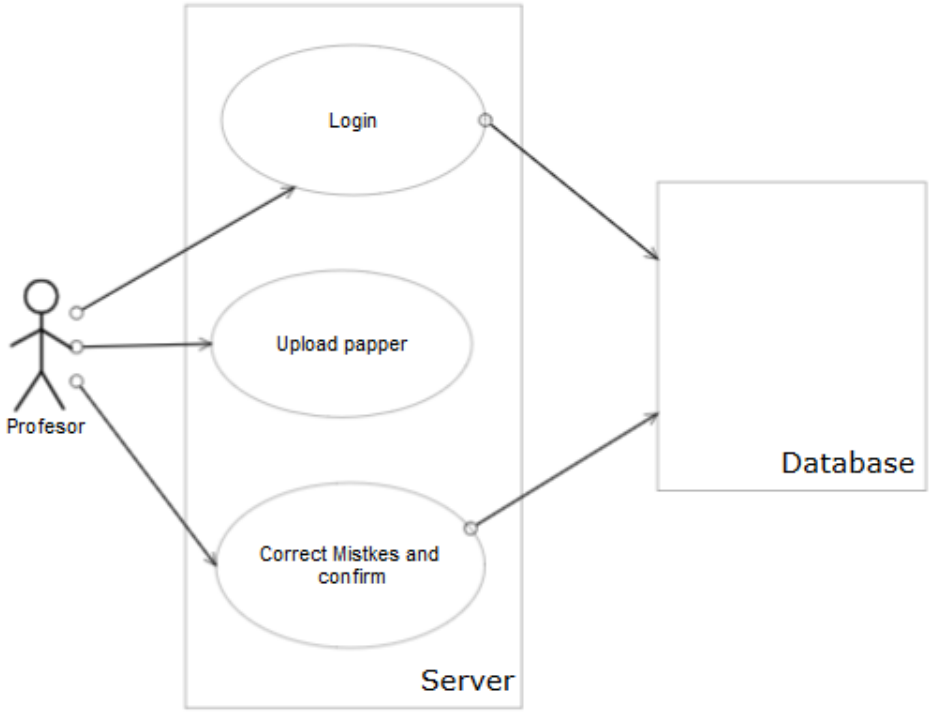
Use casul aplicației

Din punct de vedere al utilizatorilor, aplicația se poate împărți în doua categorii de use casuri ce împrtășesc aceași bază de date și același server.

1. Cazul în care profesorul utilizează aplicația.

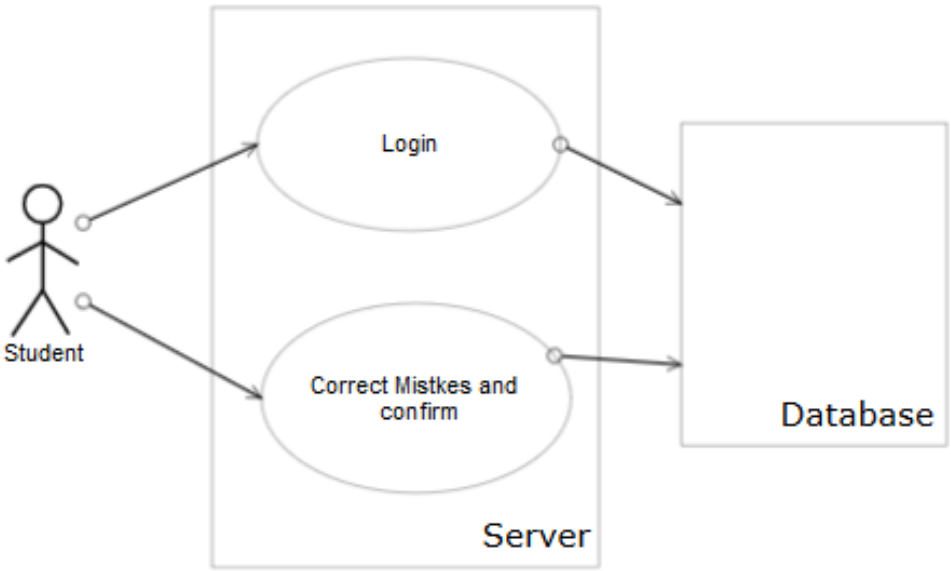
   Ca prima acțiune la care este supus profesorul este acea de a se înregistra. Acesta este nevoit să introducă email-ul și parola, acestea sunt trimise către backend unde sunt interogate și validate de către baza de date. După ce înregistrarea s-a realizat cu succes, utilizatorul de tip profesor este nevoit să încarce în aplicație o

foaie de hârtie standard, scanată. Aplicația preprocesează imaginea și o parsează, iar ca și rezultat se întoarce tabelul în formă editabilă. După corectarea greșelilor de recunoaștere ale caracterelor, profesorul își dă acordul de a trimite datele din tabel către server, acesta le stochează în baza de date aferentă aplicației.

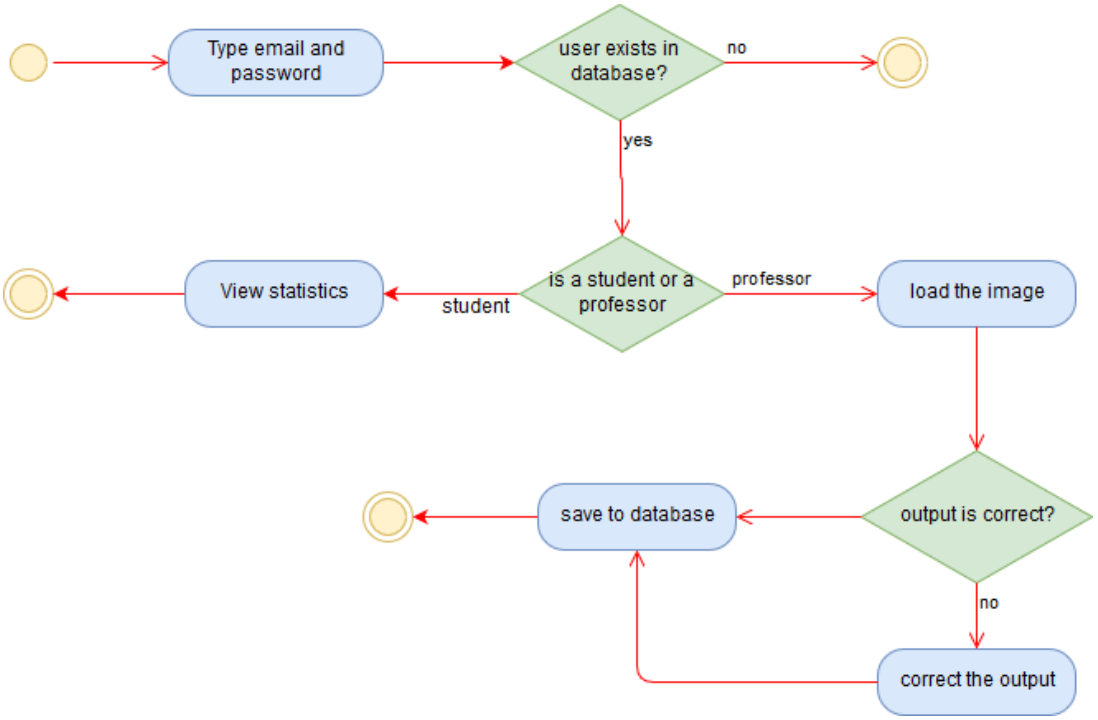


2. Cazult în care studentul utilizează aplicația.

   Se foloește aceași funcționalitate de logare în aplicație ca și la profesor. Funcționalitate ce diferă fată de un profesor este acea că studentul își poate vizualiza statistica acestuia ce include punctaje și prezențe la materiile acestuia din anul și semestrul curent.

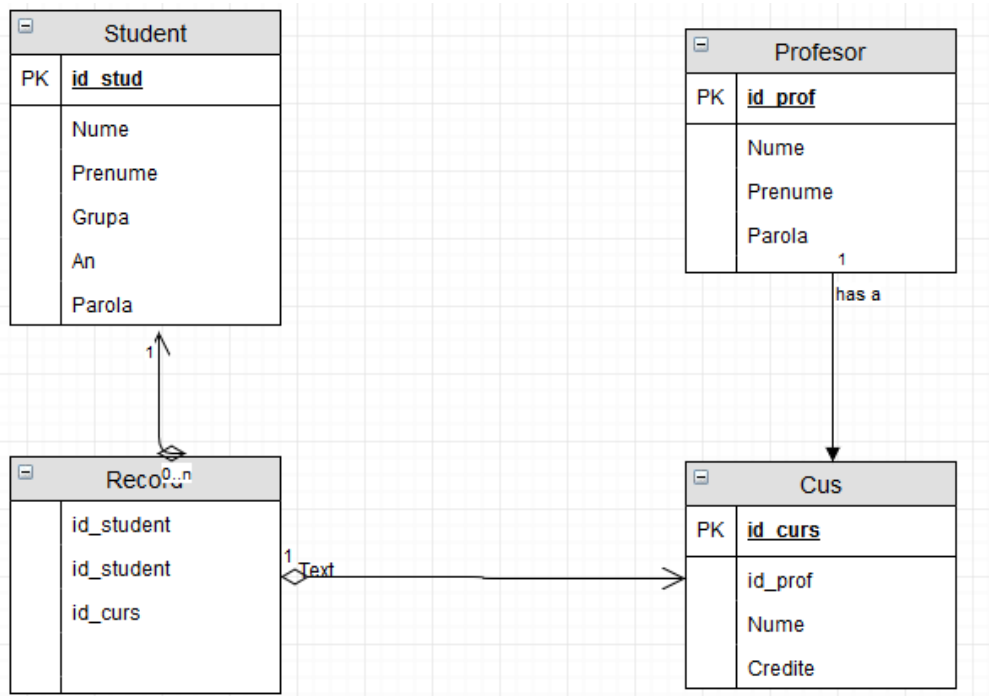


Vizualizarea tuturor activităților, din punct de vedere a ambelor tipuri de utilizator, este integrată în diagrama activităților. Acest tip de diagramă reunește cele doua use-casuri pentru o întelegere deplină a modului de funcționare a sistemului.



Arhitecura bazei de date

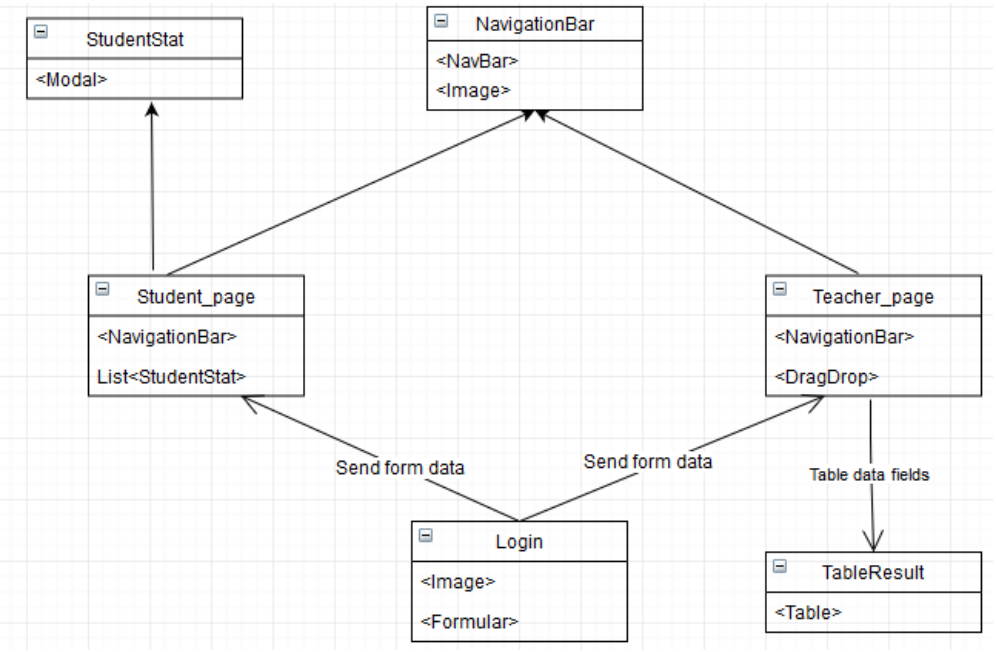
Din punctul de vedere a bazei de date, structura acesteia este constituită din 4 tabele. Tabelele Student, Profesor și Curs ce au în componența lor o cheie unică și primară, iar tabela Record conține ca și cheie primară o cheie compusă din id-ul studentului în combinație cu id-ul cursului. Baza de date se presupune a fi o extensie a bazei de date ce este folosită în cadrul facultății, din acest motiv sunt preluate doar cele mai de interes tabele și modelate astfel încât să ajute aplicația.



În cadrul tabelelor sunt reținute doar campurile ce sunt considerate de interes în cadrul aplicației.

Arhitectura clientului

Arhitectura clientului sau frontend-ul sistemului este compus din 6 clase ce comunică cu serverul și îsi transmit între ele o serie de proprietăți în funcție de desfășurarea acțiunilor de către utilizator. Unele clase se pot refolosi în cadrul altor clase, iar modularizarea acestora este aplicată cu scopul de a păstra un cuplaj adegvat între clase. Clasa cu care aplicația pornește și de unde restul acțiunilor au loc este cea de *Login*. Modalitatea de transmitere a datelor din cadrul unei componente către altă componentă este prin intermediul props-urilor.



Clasele care sunt legate direct de DOM-ul aplicației și rutează spre o anumită adresă din indexul clientului sunt: Login, Student_page și Teacher_page.

Arhitectura serverului

Arhitectura serverului este impărțită în doua componente ce comunică între ele prin cereri de tip post. Componenta ce se ocupă de gestionarea imaginei primite de la modulul de frontend este alcătuită din clase de tip servlet ce comunică cu clientul prin schimbul de informații, dar și din clase obișnuite ce alcătuiesc algoritmi necesari pentru preprocesare și segmentare. Servletul ce comunică cu marea parte din componența backendului este cel care primește imaginea în format base64. Celelalte servlete realizează interogări directe către baza de date. După ce imagiea este decodată aceasta este trimisă înspre preprocesare și segmentare, de unde se extrag liniile iar mai apoi caracterele. Se extrag anumite trăsături din fiecare imagine ce reprezintă un caracter iar aceste trăsături sunt trimise înspre serverul ce găzdiește rețeaua neuronală.

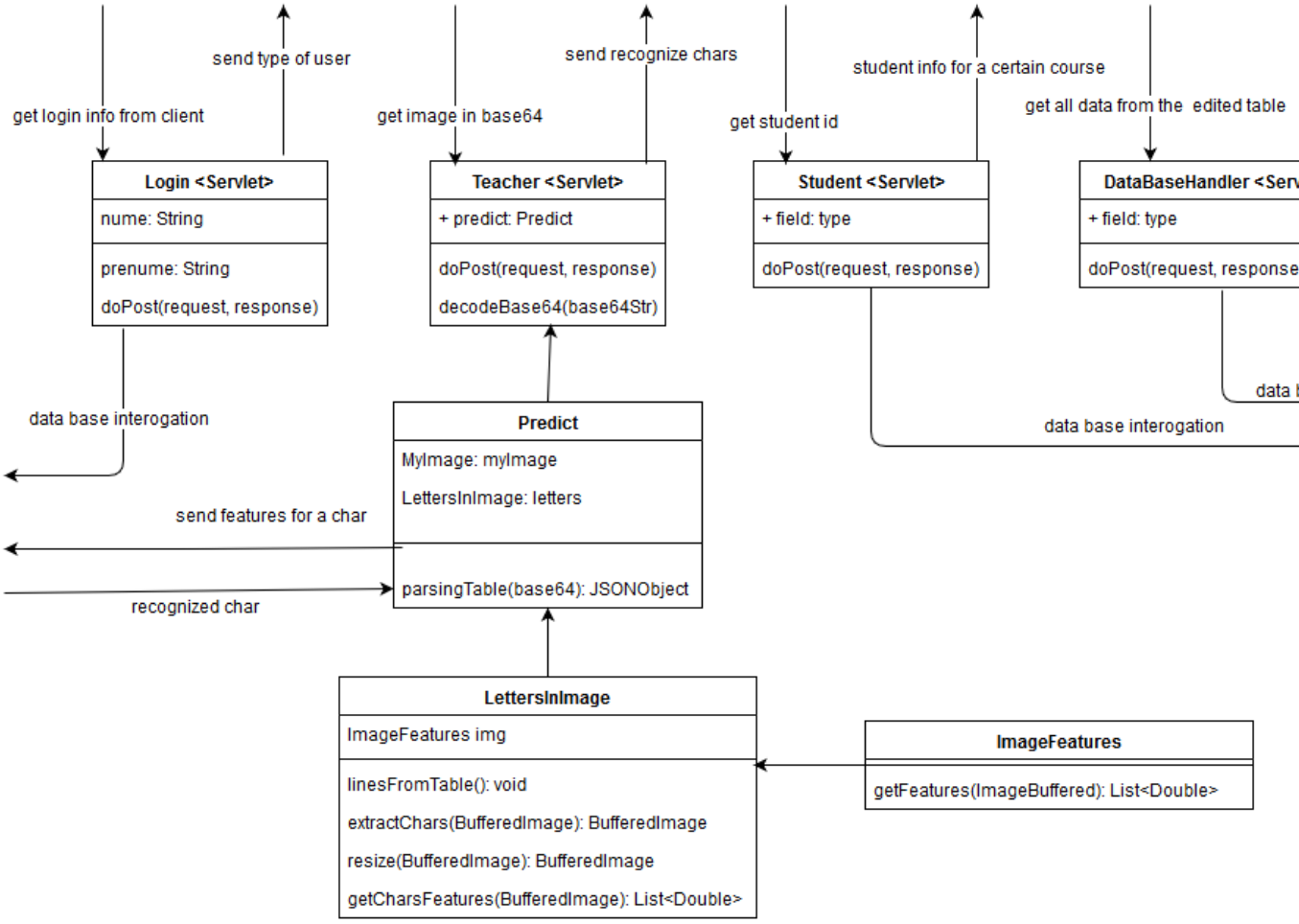


Fig. 2 arhitectura serverului în java

Clasa predict realizează o cerere de tip post către a doua componentă ce este și serverul ce găzduiește rețeaua neuronală. Se întoarce ca și răspuns către serverul java, textul reconstruit în urma clasificării caracterelor, iar acesta are rolul de a-l transmite în

continuare către modulul de frontend.

În cazul în care aplicația folosește în locul rețelei neuronale, API-ul Google Vision, atunci singura schimbare majoră se realizează la schimbarea rețelei cu API-ul propriul zis.

Implementare

Tehnologiile folosite pentru implementarea proiectului, atat că vorbim despre client, server sau baze de date, sunt unele dintre cele mai utilizate, rapide și performante momentan în domeniul IT.

Pentru implementarea clientului se folosește librăria pentru JavaScript a celor de la Facebook, Reactjs. Mediul de lucru pentru client este editorul de text Visual Studio Code.

Pentru principalul server al aplicației se folosește limbajul java, iar mediul de lucru este Intellij IDEA. Deoarece baza de date ObjectDB depinde de limbajul java, acelasi mediu de lucru este folosit și în cadrul bazei de date.

Deși librăria tensorflow este realizată sub o arhitectură c++, pentru motive evidente de rapiditate, aceasta este mapată ca și frontend în python pentru ușurința utilizării, de aici și constrangerea de a utiliza python în construirea rețelei neuronale. Serverul este unul Flask ce răspunde la cererile de tip post din partea serverului mare ce are rol de client. Alternativa rețelei este dată de API-ul Google Vision.

Backend

În acest capitol este prezentată partea de backend a aplicației care este tot odată și web serverul aplicației ce este construit folosind Apache Tomcat pentru serverul în Java, iar pentru cel în Python este folosit un framework web numit Flask.

Apache Tomcate este o implementare software open source a servletelor în Java și a tehnologiei de websockets de către cei de la *Apache software foundation*. Acesta are în componența sa un container web numit Catalina care încarcă toate cererile de tip http și are obligația de a instanția metodele post și get. De asemenea, Tomcate folosește și un conector http numit cynote.

Principalele motive pentru care s-a ales un webserver de tip Tomcat este stabilitatea acestuia și faptul că oferă doar cele mai importante și necesare funcționalități pentru a rula un server.

Flask este un un framework web dezvoltat în Python. Acesta este clasificat ca și un microframework deoarece nu necesită o librărie particulară. Acesta suportă extensii ce pot fi adăugate aplicației ca și cum acestea ar fi fost implementate în Flask de la început.

Motivele pentru care s-a ales rularea unui server în Python cu ajutorul microframework-ul Flask este acela că Flask este gândit pentru o aplicație de tip REST și este ușor de folosit și de integrat cu alte librării.

Funcționalitățiile serverului aplicației sunt:

1. distincția dintre un student sau profesor
2. primirea printr-o cerere a principalei date de intrare, adăugată de către utilizator, care este imaginea unei foi de prezență sub formă tabelară în format base64 si preprocesarea ei
3. extragerea a fiecărui caracter ce intră în componența tabelului
4. recunoașterea caracterelor

5. oferirea de statistici

Tehnologia folosită pentru punctele 1,2,3,5 este **Java** cu un server de tip **Tomcat**, iar pentru punctul 4 este folosit un server ajutător în **Python** de tip **Flask**.

Toate firele de comunicare cu frontend-ul sunt inițiate de către client prin intermediul unei cereri de tip Post, iar serverul are obligația de ai transmite un răspuns. În servletul de tip tomcat acest lucru este posibil prin intermediul metodelor puse la dispoziție (doPost, doGet, etc.). Cererea de tip post este stocată într-un obiect de tip JSON, chiar daca această cerere este realizată dinspre client-server sau server-server, iar răspunsul este stocat sub aceași formă și trimis înapoi celui ce inițializează schimbul de informații.

Pentru a transforma clasele adnotate în clase de tip servlet si să primească anumite date la o adresă stabilită, se crează un fișier de configurare a aplicației web pe partea serverului. Acest fișier poartă numele de *web.xml* și se găsește în folderul web din cadrul stucturii proiectului.

Fișierul web.xml este un fișier de configurare a unei aplicații web în java. Instruiește containerul servletului pentru a decide ce clase să fie încărcate și să se comporte ca și un servlet. Fișierul web.xml mai setează ce fel de cereri să accepte din browser. De aici și clasa CrossFilter creată și introdusă în cadrul fisierului, ce specifică clar ce cereri sunt acceptate din partea browserului. Aceste cereri sunt: *DELETE, HEAD, GET, OPTIONS, POST, PUT*. Mecanismul Cross-origin (CORS), permite comunicarea dintre un client și un server ce au domeniu diferit. Pentru a realiza acest lucru în Java, este necesar declararea unei clase ce are rol de filtru și specifică metodele prin care va comunica serverul și clientul.

Serverul Flask rulează la un port diferit de cel în Tomcat, iar acesta primește de asemenea cereri de tip post și întoarce ca și răspuns, o listă pentru fiecare linie din tabel în care textul este reconstruit într-o formă editabilă.

Prin urmare se propune o implementare proprie al unui recunoscător automat de caractere scrise de mână și prin doua metode diferite de a extrage trăsăturile. Algoritmul de OCR propus se va compara cu API ul Google Vision, criteriul fiind acuratețea.

Ca și librărie de funcții informatice folosită semnificativ în cadrul aplicației pentru prelucrarea de imagini este **OpenCV**.

Preprocesarea imaginei

Preprocesarea este un pas important în recunoașterea de caractere. Acest procedeu include eliminarea zgomotelor, binarizarea și normalizarea.

Deoarece inputul este primit din cadrul modulului de front-end într-un format codat a imaginei în *base64*, prima acțiune este acea de a transforma imaginea în formatul *Mat*, care este o clasă din *OpenCV* alcătuită din doua părți: un pointer către o matrice ce conține valoarea pixelilor și a doua parte care constituie informații despre matrice (dimensiune, adresa matricii stocate etc.). Tot odată prin această acțiune, imaginea este binarizată prin transformarea tuturor pixelilor din format RGB în pixeli de tip GRAYSCALE cu ajutorul funcției *Imgcodecs.IMREADGRAYSCALE*. Pentru a reduce zgomotul din imagine se folosește un prag adaptiv, dilatarea și mai apoi erodarea pe întreaga imagine. Aceste preprocesări facandu-se cu ajutorul funcțiilor:

1. *adaptiveThreshold* care aplică pragul adaptiv peste o listă și transformă o imagine de tip grayscale intr-o imagine conform formulei

$$dst(x,y) = \begin{cases} \texttt{maxValue} & \text{if } src(x,y) > T(x,y) \\ 0 & \text{otherwise} \end{cases}$$

2. *dilation* este un procedeu ce crește aria unui obiect. Dilatarea poate elimina zgomotele albe din acea imagine, dar micșorează și imaginea, astfel dupa erosiune se poate aplica dilatarea pentru o mai bună eliminare a zgomotului.

3. *erosion* este procedeul învers dilatării prin care se elimină micile zgomote albe din imagini. Aceasta însă poate să fie folosită pentru a despărți doua elemente conectate. Procedeul începe prin formarea unui kernel, ce este o matrice. Un pixel din imagine este ales. Pixelul va fi ales 1 doar dacă toți pixelii de sub kernel sunt 1, altfel pixelul va fi erodat. Astfe grosimea imaginei scade, deoarece toți pixelii de la granițe sunt șterși.

```
private void Adaptiv() {
    System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
    Imgproc.adaptiveThreshold(matFormatImage, matFormatImage,  maxValue: 255, Imgproc.ADAPTIVE_THRESH_MEAN_C, Imgproc.THRESH_BINARY,  blockSize: 7,  C: 10);
    Photo.fastNlMeansDenoising(matFormatImage, matFormatImage,  h: 3,  templateWindowSize: 7,  searchWindowSize: 21);
    Imgproc.dilate(matFormatImage, matFormatImage, Imgproc.getStructuringElement(Imgproc.CV_SHAPE_CROSS, new Size( width: 1,  height: 1)));
    Imgproc.erode(matFormatImage, matFormatImage, Imgproc.getStructuringElement(Imgproc.CV_SHAPE_CROSS, new Size( width: 1,  height: 1)));

}
```

Componenta de recunoaștere a textului

În urma preprocesări imaginei obținem un element modificat în scopul extrageri meta datelor din el. Se clonează imaginea preprocesată, iar din imaginea clonată se decupează pe rând liniile, iar mai apoi din fiecare linie se extrag caracter cu caracter prin același procedeu de decupare ca și la linii. Fiecare caracter decupat este redimensionat. Se extrag o serie de trăsături cu ajutorul metodelor *discret cosin transform, haar wavelet* ce ne furnizeaza o listă de valori din intervalul [-1,1], iar mai apoi aceste trasături sunt trimise către modulul rețea ce decodifică lista de valori într-un caracter alfa numeric. Toate aceste caractere ce sunt transpuse se reconstruiesc pentru fiecare linie a tabelului și sunt trimise ca și răspuns, sub forma unei liste, către serverul în java. Reconstrucția cuvintelor se reazlizează cu ajutorul faptului că fiecare celulă este separată printr-o linie verticală, iar prin această informație putem deduce exact unde este limita fiecărei celule din linia respectivă.

În continuare sunt prezentate stările prin care recunoscătorul este alcătuit. Se începe prin segmentarea textului din imagine, mai apoi metodele folosite pentru a extrage trăsături, rețeaua neuronală și API-ul produs de Google.

Segmentare text

Din momentul în care avem imaginea preprocesată, aceasta fiind intr-un format standard, sub formă tabelară, ca prima acțiune pentru extragerea datelor este acea de a identifica poziția liniilor. Acest lucru este posibil prin setarea unui prag negativ pentru a evidenția liniile tabelului. Aceasta acțiune se desfășoară pe o clonă a imaginei originale și preprocesate. Procedeul prin care se obține imaginea cu prag negativ este tot prin folosirea funcției *adaptiveThreshold*, doar că de această dată se optează pentru parametru negativ pentru a avea rezultatul dorit.

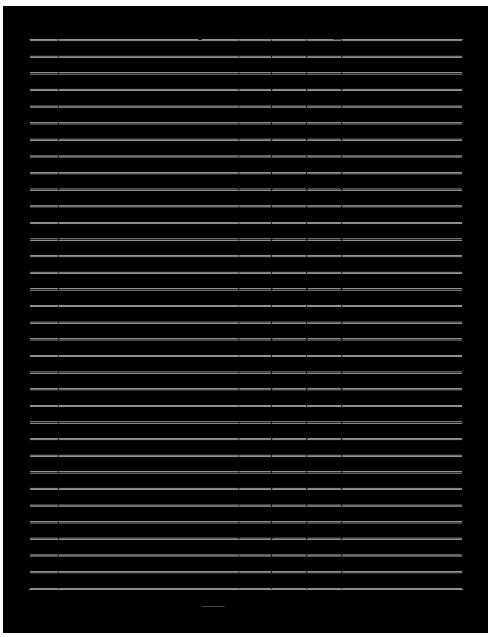


Fig.3 clona imaginei ce deservește la identificarea liniilor

Se parcurge imaginea în care sunt evidențiate liniile tabelului în plan vertical și se reține pentru fiecare linie a tabelului, pixelii albi din patea superioară, respectiv inferioară. Deoarece imaginea coinicide cu cea originală din punct de vedere a structurii tabelului, acele poziții ale liniilor sunt folosite pentru a decupa fiecare linie din imaginea originală.

Dupa ce linia este decupată din imaginea originală și preprocesată, aceasta este rotită 90° în sensul acelor de ceas. Procedeul de rotire se produce cu ajutorul clasei *AffineTransform* care mapează o matrice într-o alta matrice. Metoda *rotate* din acestă clasă realizează rotirea în jurul unei ancore, care în situația actuală este dată de mijlcul cartezian al coloanei în poziție orizontală.

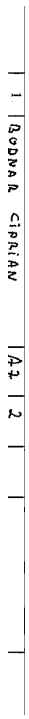


Fig.4 rezultatul extragerii liniilor

Pentru a extrage fiecare caracter separat din liniile verticale, se cauta primul și ultimul pixel diferit de culoarea negru pentru a afla în acest fel lațimea finală a literei, deoarece înălțimea este fixă ce este exact cât înălțimea coloanelor din tabel. Căutarea se face parcurgand imaginea, iar daca indexul este diferit de înăltimea coloanei rotite și am găsit cel puțin un pixel diferit de culoarea albă, atunci putem considera că avem coordonatele ce încadrează litera. Deoarece am lucrat pe linia rotită 90° în sesnul acelor de ceas, litera ce urmează a fi decupată trebuie rotită invers acelor de ceas prin folosirea funcției, *rotate* a clasei *AffineTransform*, doar cu parametrul ce sugerează direcția.

Urmează uniformizare a tuturor literelor spre o dimensiune fixă și egală între înălțime și lățime. Dimensiunea fixă este impusă de către rețeaua ce recunoaște, ce dorește un număr egal de trăsături pentru fiecare caracter. S-a ales ca dimensiunea 40x40 pixeli datorită și faptului înălțimii coloanelor din tabel.

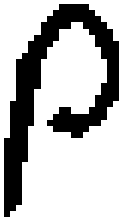


Fig.5 rezultatul extragerii caracterelor

Extragerea trăsăturilor

Structura generală a unui OCR este alcătuită din parte de preprocesare și partea de extragere a trăsăturilor. Alegerea metodei prin care se extrag trăsăturile este probabil cea unul din cei mai importanți factori în atingerea unui procent de recunoaștere ridicat. Considerând aria largă de metode de a extrage trăsături existente, o problemă importantă este deciderea ce fel de trăsături sunt potrivite pentru o anumită aplicație.

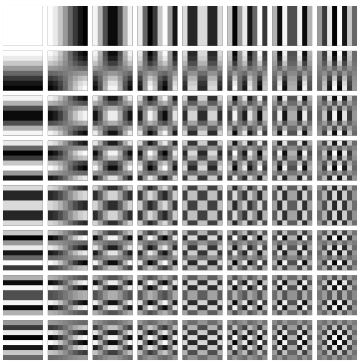
Transformările fourier și wavelet sunt doua metode populare verificate în multe aplicații de tip OCR. Rezultatele fiecărei metode depinde în mare parte de setul de date al problemei.

În cadrul acestui modul se vor folosi cele doua metode în comparație.

Discret cosine transform

Un algoritmul folosit este *discrete cosine transform (DCT)*. DCT este o transformare de tip Fourier, acest tip de transformare descompune și procesează un tip de semnal digital într-o suma a unei funcții trigonometrice.

Similar ca și datele de tip audio, imaginile pot fi modelate de asemena prin intermediul undelor. O transformare de tip *discrete cosine transform* exprimă o secvență finită de puncte în termeni a unei sume al cosinusului, acestea oscilând la frecvențe diferite. Utilizarea funcțiilor de cosinus în schimbul sinusoidalelor, este critică pentru comprimare, deoarece sunt necesare mai puține funcții de cosinus pentru a aproxima un anume semnal, în timp ce pentru ecuațiile diferențiale cosinusurile exprimă o anumită alegere pentru condițiilor limită.



În domeniul spațial (înainte de a aplica DCT) datele sunt descrise prin valori digitale pentru fiecare pixel și imaginea este reprezentată ca o listă de pixeli. După transformare, imaginea este descrisă de coeficenții din spațiul frecvențial pentru axa orizontală și verticală.

Primind litera scalată la o dimensiune fixă de 40x40, sub forma unei matrici, iar peste această matrice este aplicată formula de bază a acestei metode. Numărul de trăsături pentru o literă este formată din 16 numere întregi cuprinse între [-128, 128].

$$X_{k_1,k_2} = \sum_{n_1=0}^{N_1-1} \left(\sum_{n_2=0}^{N_2-1} x_{n_1,n_2} \cos\left[\frac{\pi}{N_2} \left(n_2 + \frac{1}{2} \right) k_2 \right] \right) \cos\left[\frac{\pi}{N_1} \left(n_1 + \frac{1}{2} \right) k_1 \right]$$

Aplicarea formulei pe fiecare celulă a matricei rezultă o compresie a acestei imagini, astfel încât compresia imaginei s-a redus doar intr-un bloc de 8x8 cu cele mai semnificative trăsături ce vor fi trimise înspre antrenare, pentru a putea clasifica imaginea.

Scopul aplicării algoritmului este acela de a diminua numarul de trăsături astfel încât sa nu trimitem întreaga imagine înspre antrenare.

Haar wavelet

Transformările wavelet sunt o serie de tehnici prin care o imagine este reprezentată în diferite nivele de rezoluție.

Haar wavelet este cea mai simplă metodă din familia transformărilor wavelet. Dezavantajul tehnic al acestei metode este acela că acest tip de metodă nu este continuă și nici diferențiabilă. Această metodă este definită astfel:

$$\psi(t) = \begin{cases} 1 & \text{if } t \in \left[0, \frac{1}{2}\right) \\ -1 & \text{if } t \in \left[\frac{1}{2}, 1\right) \\ 0 & \text{otherwise} \end{cases}$$

Pentru fiecare imagine, mulțimea de trăsături este cuprinsă în intervalul [-1,1], iar lungimea este dată de dimensiunea imaginei, în cazul de față vorbim despre o dimensiune de 40px, adica un set complet de 1600 de valori trimise înspre antrenare.

Antrenarea datelor

Scopul acestui modul este de a prezice litera (caracterul) corespunzătoare, în funcție de vectorul de caracteristici furnizat de către modulul de extragere a trăsăturilor. Pentru aceasta se alcătuieste o rețea neuronlă în Python cu ajutorul API-ului Keras și al framework-ului Tensorflow.

O rețea neurală artificială este un tip de învățare automată inspirat din rețelele neurale existente în creierul uman. O rețea neurală este formată din neuroni interconectați, organizați în straturi, care comunică între ei pentru a ajusta performanța întregii rețele. Fiecare conexiune dintre neuronii artificiali funcționează ca o sinapsă dintr-un creier biologic, transmițând un semnal de la un neuron la altul. Un neuron poate să transmită mai departe semnalul trimis, sau să efectueze o procesare asupra lui și apoi să semnaleze mai departe altor neuroni.

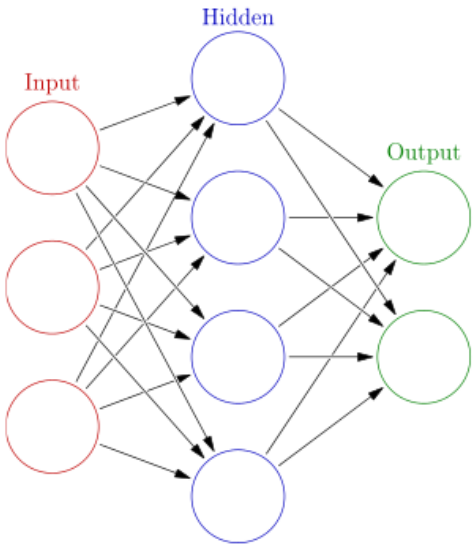
O rețea neuronală are următoarele straturi:

1. Stratul de input Stratul de input este format din neuroni de intrare, care preiau datele de intrare și le transmit mai departe rețelei. De cele mai multe ori, neuronii de intrare nu au ponderi și nici funcții de activare, având un rol pasiv în cadrul rețelei.

2. Stratul/straturile hidden Straturile hidden se ocupă de procesarea inputului. Există diferite moduri de a organiza straturile ascunse ale rețelei, în funcție de tipul rețelei: în cadrul unei rețele feedforward doar vor transmite datele prelucrate,iar în cadrul unei rețele convoluționale ce se ocupă de procesarea imaginilor vor exista straturi convoluționale și de pooling pentru a descoperi anumite trăsături în imagine, iar la rețelele de tip recurent, în straturile ascunse vor apărea conceptele de memorie și stări. În toate cazurile, ideea de bază este ca straturile hidden vor prelua datele de intrare neprelucrate de la stratul de input, apoi vor prelucra acele date conform ponderilor asignate și calculate în diverse moduri, pasând apoi datele procesate ultimului strat, cel de output. Numărul de straturi hidden, organizarea lor, precum și tipurile de straturi și neuroni folosite depind atât de structura rețelei cât și de scopul în care este folosită aceasta.

3. Stratul de ieșire Neuronii din stratul de ieșire pot fi proiectați diferit față de straturile anterioare, reprezentând actorii finali ai rețelei și având rolul de a produce outputul programului. Astfel, stratul de ieșire se coalizează și produce în mod concret rezultatul final, cu rol în eficientizarea și îmbunătățirea acestuia.

   Referitor la cele menționate anterior, pentru a înțelege mai bine rețeaua neurală, cele trei nivele de straturi(stratul de intrare, straturile ascunse și stratul de ieșire)vor fi privite împreună ca întreg.



Pe lângă straturile descrise, o rețea neuronală mai are o funcție de cost folosită pentru a aproxima diferența dintre valoarea prezisă de rețea și valoarea reală. Antrenarea unei rețele neurale are ca scop principal reducerea valorii funcției de cost, existând mai multe tehnici pentru a îmbunătați acest procedeu.

Rețeaua folosită pentru antrenare este bazată pe un model secvențial cu un strat hidden cu 500 de neuroni de tip relu. Stratul de output este de unul de tip softmax cu un optimaizer adam. Antrenamentul se face în 7 epoci. Diferența dintre cele doua tipuri de date primite la input constă în ajustarea stratului de input, deoarece pentru DCT se primesc mai putine caracteristici în comparație cu metoda Haar.

Pentru antrenare există un fișier numit labelsset în care sunt încărcate mulțimea de litere și cifre ce rețeaua le antrenează, în cazu rețelei aplicației, dimensiuea mulțimii este de 37 și este compusa din cifrele 0-9 și literele A-Z. Nu s-au luat în calcul diacriticele deoarece asemănarea acestora cu literele din care derivă este ridicată și astfel procentajul de recunoaștere scade drastic.

Acestă acțiune de antrenare se realizează separat, iar fișierul rezultat este folosit în cadrul serverului în recunoașterea caracterelor împreună cu fisierul labelsset ce este folosit și în cadrul antrenării.

Google cloud vision API

Google cloud vision este un API este un model bazat pe învățare automată ce se centrează în recunoașterea de imagini. Acesta are o interfață REST api simplă.

În cazul aplicației folosim acest tool în cazul în care antrenarea cu una din cele doua metode de recunoștere nu este suficientă pentru a servi scopului nostru.

Ca primă acțiune a tool-ului este să analizeze imaginea și să segmenteze locația textului, acest lucru nu mai fiind nevoie din caza că la input API-ul deja primește fiecare literă segmentată. După ce primește inputul, api-ul își folosește componenta de recunoaștere pentru a genera textul. Posibile greșeli sunt corectate de către acesta prin căutarea în dicționar a similitudinilor.

Toate acestea sunt realizate într-o rețea convuluțională în care fiecare neuron este conectat la un subset de neuroni ai fiecărui strat.

Pentru a putea folosi acest API, este nevoie de o cheie API pentru dezvoltatori. Dupa obținerea acestei chei unice, totul se rezumă la trimiterea de cereri post cu url-ul imaginei.

```
let body = {
  "requests": [
    {
      "image": {
        "source": {
          "imageUri": "https://i.imgur.com/Nlot5mR.jpg" //image URL
        }
      },
      "features": [
        {
          "type": "TEXT_DETECTION",
          "maxResults": 1
        }
      ]
    }
  ]
}
```

Baza de date

Componenta aplicației care are rolul de a ține informațiile referitoare la studenți, profesori și cursuri. Deoarece aplicația este concepută în special pentru facultatea de informatică, dar nu numai, se presupune că se folosește baza de date a facultății dar unele informații din ea sunt triate pentru simplitate.

Deoarece serverului este în cea mai mare parte conceput în limbajul Java, s-a ales integrarea unei baze de dade dedicate pentru acest limbaj pentru a spori productivitatea și pentru a diminua din timpul de execuție.

ObjectDB este o puternică bază de date orientă obiect (ODBMS) concepută pentru Java. Este o bază compactă, ușor de folosit și foarte rapidă. Este folosită în modul client-server. Spre deosebire de celelalte baze de date, ObjectDB nu furnizează propriul API. Pentru a lucra cu o astfel de baze de date, se folosește unul din standardele Java: JPA sau JDO. Ambele API-uri fiind construite în ObjectDB, astfel un ORM intermediar nu este necesar. Folosirea unei baze de date relaționare (RDBMS) pentru a salva obiecte Java necesită un timp de conversie dintre obiecte și tabelele din acea bază de date. Este adevărat că unele ORM-uri pot reduce din lucrul manual, dar nu pot elimina timpul în plus de procesare.

ObjectDB furnizează o bogată mulțime de proprietăți. Multe dintre aceste proprietăți sunt implementate ca și parte ObjectDB-ului pentru JPA. Proprietățile fundamentale ale bazelor de date relaționale (chei primare, interogări grupate și agregate) care de obicei lipsesc din bazele de date orientat obiect sunt de asemenea suportate de către ObjectDB, combinând astfel ambele tipuri de baze de date.

Baza de date comunică direct cu backendul chiar de la începutul aplicației unde userul este nevoit pentru a se înregistra, iar mai apoi pentru a oferi statistici studenților, sau pentru a a adăuga noi date (punctaje/prezențe).

În cadrul aplicației baza de date este formată din 4 entități care sunt văzute ca și tabele într-o bază de date de tip RDBMS. Cele 4 entități sunt:

1. Student : ce servește în a reține date despre student.

2. Profesor : ce servește în a reține date despre profesor.

3. Cours : ce conține o referință către profesorul titular cursului.

4. Record : entitate ce reține pentru un anumit student și un anumit curs, date despre activitatea studentului la acel curs.

Pentru ca baza de date să fie accesibilă, trebuie să înregistrăm un Listener pentru a inițializa baza de date. Aceasta este deschisă (sau creată daca nu există deja) cand serverul aplicației web pornește prin instanțierea unui *EntityManagerFactory*. Iar baza de date se va închide când serverul aplicației este oprit sau prin închiderea *EntityManagerFactory*.

Pentru a stoca unele obiecte în ObjectDB folosind JPA-ul, trebuie să ne definim clasa ca fiind una de tip entitate, acest lucru se realizează adnotând clasa respectivă cu *Entity*. Alte tipuri de adnotări folosite sunt în cadrul entității. *Id* cu un interval specificat și optând pentru opțiunea de *SequenceGenerator* marchează faptul că variabila destinată este cheie unică în entitatea respectivă. Astfel de chei auto-generatoare sunt folosite în clasele Student, Profesor și Course. Clasa Record are ca și cheie primară, cheia dintre id-ul cursului și id-ul studentului la care se face referire.

```
@Entity
@SequenceGenerator(name="stud", initialValue=1, allocationSize=100)

public class Student {

    String nume;
    String prenume;
    int an;
    String grupa;
    String parola;
    @Id @GeneratedValue(strategy= GenerationType.SEQUENCE, generator="stud") int id_student;


    Student(String nume, String prenume, int an, String grupa, String parola ){
        this.nume = nume;
        this.prenume = prenume;
        this.an = an;
        this.grupa = grupa;
        this.parola = parola;

    }
```

Pentru a realiza una din acțiunile CRUD, trebuie să preluăm atributul setat în Listener pentru a începe tranzacția. Mai apoi se definește un obiect de tip Query ce conține interogarea dorită. Operațiie CRUD sunt cele obișnuite din orice baza de date RDBMS. Interogările bazei de date sunt făcute la cererea utilizatorului atunci când

acesta dorește realizarea unei anumite acțiuni, ex: trimiterea tuturor informațiilor din tabel pentru a se stoca.

Frontend

În zilele noastre se dorește ca orice componentă a IT-ului să fie una rapidă, portabilă și durabilă. Acest modul se ocupă cu experiența vizuală a utilizatorului aplicației cât și practică. Deoarece aplicația este de natură web, soluția noastră va fi implementată cu ajutorul tehnologiei React.

React este o librărie JavaScript folosită în construcția de interfere. Este construit de către Facebook. React este folosit ca și o bază în dezvoltarea unei paginilor web sau a aplicațiilor mobile.

React are o structură bazată pe componente. O componentă poate să fie privită ca și o piesă de lego. Aceasta este creată din componente mai mici. Fiecare decide cum să fie redată, iar fiecare componentă are propria ei logică. Reutilizarea codului se poate realiza ușor prin reutilizarea componentelor. Acestea primesc anumite proprietăți de la componentele părinte, iar componentele de tip copil pot trimite proprietăți celor de tip părinte. Deoarece fiecare componentă are propria logică, unele valori se pot salva în state-ul propriu. Construcția unei componente în react se face invocând metoda React.createClass() care activează la randul ei și o serie de cicluri de viață.

Construirea unei aplicații web necesită o puternică intercațiune cu utilizatorul. Din acest motiv se produc și multe actualizări în DOM, iar manipularea eficinetă a DOM-ului încă este o problemă în dezvoltarea web. React dorește să imbunătățească performanța web din acest motiv, și din acest motiv folosește un DOM virtual, iar mai apoi un algoritm eficient ce găsește diferența dintre DOM-ul original și cel virtual. Acele diferențe sunt adăugate în DOM-ul inițial.

În cadrul modulului de frontend sunt construite 3 pagini care sunt furnizate la 3 adrese diferite. Fiecare din acestă pagină este construită folosind cel puțin 2 componente. Ca primă pagină cu care utilizatorul interacționează este cea de login. Aceasta este construită folosind un formular simplu în care se introduce adresa de email a facultății și parola.

Pagina profesorului este alcătuită din componenta Navigationbar, care se regăsește și în pagina studentului si primește ca și proprietate numele utilizatorului, si de componenta DropDown care are la randul ei mai multe componente copil:

1. DropZone componentă ce se ocupă cu încarcarea imaginei.

2. TableResult componentă ce se ocupă cu afișsarea automată în funcție de tipul de imagine aleasă a unui tabel editabil.

```
<div className="teacher-page">

    <Navigationbar name= {this.state.teacher_name}  ></Navigationbar>
    <Dragdrop id={this.props.location.state["id"]}></Dragdrop>
</div>
```

Fig.6 folosirea componentelor

Pagina studentului este alcătuită din componenta Navigationbar împreună cu un număr de componente de tip StudentStatus care primește ca și proprietăți numele cursului și toate informațiile unui student în legătură cu acel curs.

Metoda prin care comunicarea și transmiterea datelor se realizează cu ajutorul funcției asincrone *fetch()*, care provine din API-ul fetch al JavaScriptului ce realizează manipularea răspunsurilor și al cererilor din cadrul pipeline-ului de tip HTTP. Ca și argument, fetch primește adresa la care se face cererea respectivă și un dicționar ce specifică modul de trimitere a datelor (POST, GET etc.), dar și conținutul pe care dorim să îl trimitem către server.

```
await fetch('http://localhost:7777/Licenta/db',{
    mode: 'cors',
    method: 'post',
    headers:  {'Content-Type':'application/json'},
    body: JSON.stringify({scan:this.state.scanResult,id:this.props.id})
}).then(response => console.log(response)).then()
```

Fig.7 exemplu de funcție fetch

Deoarece transmiterea datelor prin metoda fetch necesită o anumită reprezentare a datelor trimise, aceste sunt stocate într-un format JSON, care necesită existența unei chei pentru un anumit câmp. Această parsare către un format JSON se realizează cu ajutorul metodei *stringify* a clasei JSON din Javascript. În cadrul aplcației toate cererile sunt de tipul POST, acest tip de cerere trimite datele către server, dar așteaptă și un răspuns din partea acestuia.

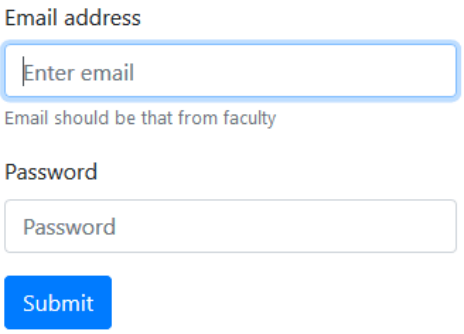
Manual de utilizare

Pentru utilizarea aplicației local, trebuie instalate următoarele dependențe:

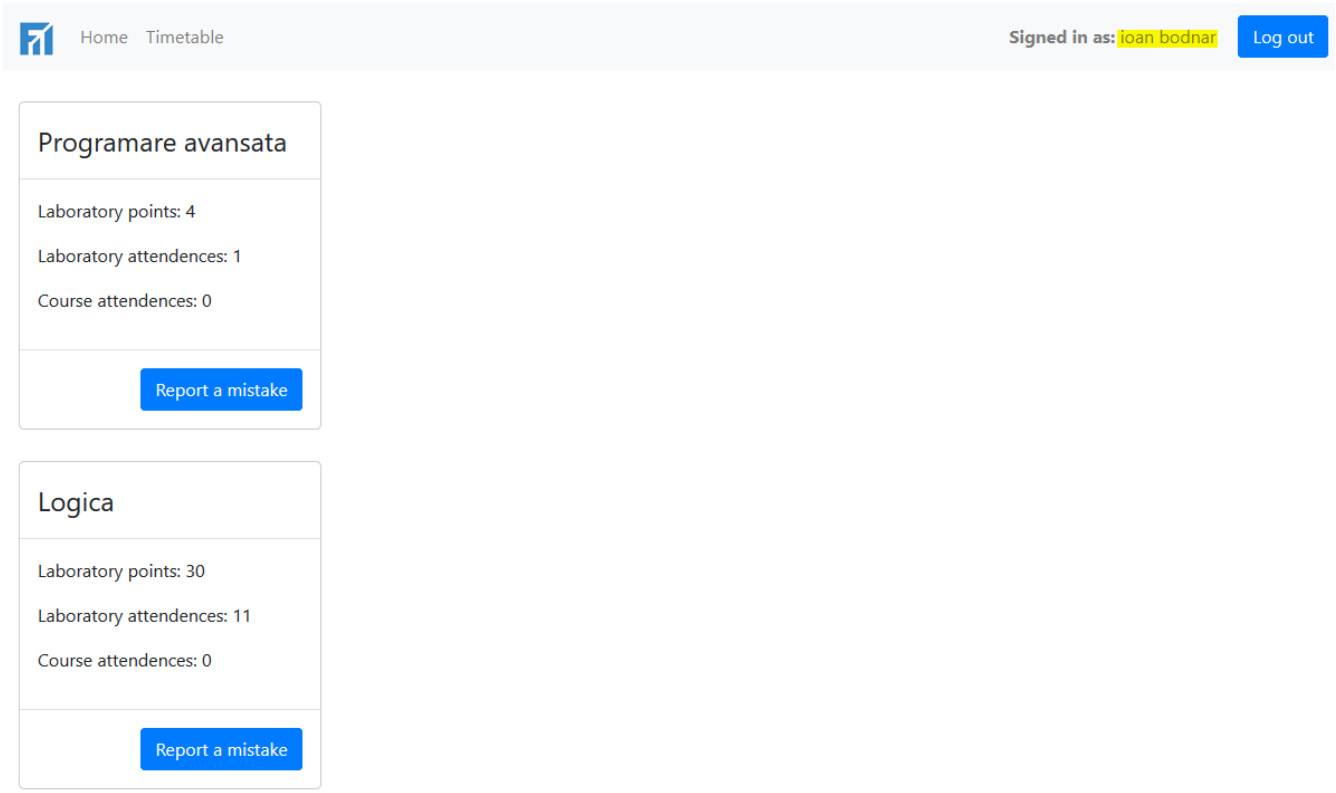
1. Node js pentru client
2. Java pentru server
3. Python pentru server

După ce toate dependențele au fost instalate, aplicația se poate rula local atât din punct de vedere al clientului, cât și a serverului. Pentru a pornirea clientului se folosește comanda npm start din linia de comandă. Pentru pornirea serverului în java din mediul de lucru Intellij, trebuie rulată aplicația având setată ca și configurație Tomcat. În ultimul rând, pornirea serverului Flask se realizează prin execuția scriptului predict.py prin comanda py din linia de comandă.

Prima pagină a aplicației este cea de înregistrare, unde utilizatorul introduce email-ul și parola pe care o deține în cadrul facultății.



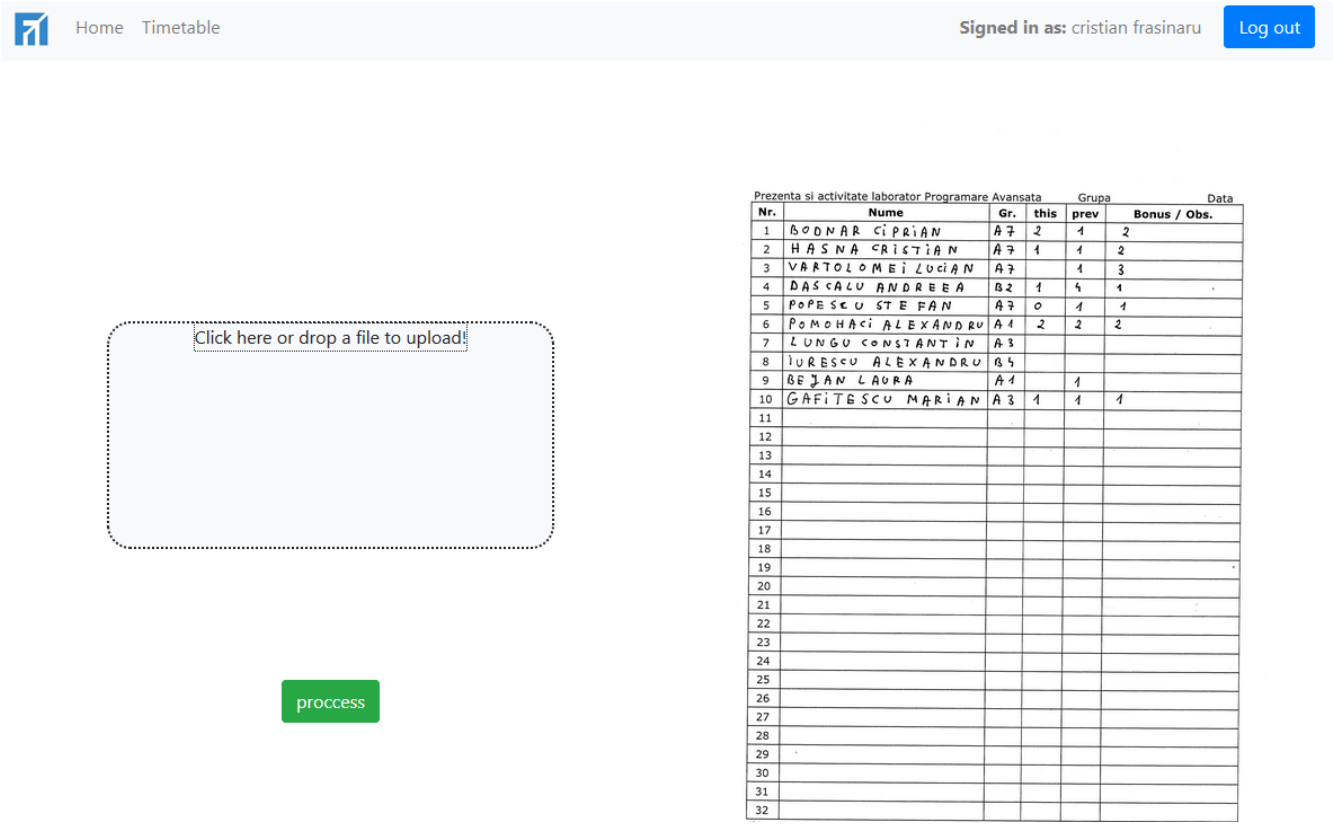
Diferențierea dintre utilizatori se face la înregistrare, considerăm cazul în care un student intră în aplicație. În partea unde se regăsețte bara de navigație este plasat un link către orarul facultății, numele utilizatorului ce a accesat aplicația cât și un buton ce deconectează utilizatorul și îl retrimite către pagina principală.



Statisticile oferite pentru studenți sunt selectate în funcție de anul și semestrul la care se află studentul respectiv. Acesta poate raporta o necorelație în punctaje și prezențe apăsând pe butonul Report a mistake care il redirectează către pagina de webmail a facultății.

Cazul în care utilizatorul aplicației este profesor, pe lângă bara de navigație ce conține aceleași proprietăți ca și în cazul studentului, diferența majoră se face în cadrul mecanismul de drag and drop, unde profesorul este nevoit să aleagă foaia de prezență scanată și să o încarce în aplicație.

În momentul în care imaginea este încărcată, aceasta va fii încărcată în partea dreaptă. Următoarea acțiune disponibilă este acea de a lansa în execuție parsarea și procesarea imaginei.



Nr.	Nume	Gr.	this	prev	Bonus / Obs.
1	BODNAR CIPRIAN	A7	2	1	2
2	HASNA CRISTIAN	A7	1	1	2
3	VARTOLOMEI LUCIAN	A7		1	3
4	DASCALU ANDREEA	B2	1	1	1
5	POPESCU STEFAN	A7	0	1	1
6	POMOHACI ALEXANDRU	A1	2	2	2
7	LUNGU CONSTANTIN	A3			
8	IURESCU ALEXANDRU	B4			
9	BEJAN LAURA	A1		1	
10	GAFITESCU MARIAN	A3	1	1	1
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					
22					
23					
24					
25					
26					
27					
28					
29					
30					
31					
32					

După ce imaginea a fost procesată, în josul paginei este afișat un tabel ce reprezintă exact tabelul din foaia de prezență. Profesorul poate edita posibile greșeli.

Nr	Name	Group	this	prev	bonus
1	DAMIAN ANDREI	A1			
2	REEMERITA MIHNEA	A1			
3	CERCEL IRINA - OLENAAA	24			
4	KioaT\u0100 MATEI	IAN			
5	HiPAN RADU - MATEI				
6	SALCEANU SERGIU	AA	2	A	
7	IURABIB ALIN				
8	IUREANU	COSMIN			
9	BLAJ MARIUS	A1			
10	ADAM Crist	;	AN		
11	CHEREI ANDREI	A1			
12	BELU C\u0102T\u0102LIN	A6	0	,	
13	GOTA STEFAN	A4	1	,	
14	STOICA DOAMA - DANA	A			

Salvarea în baza de date se realizează apasând butonul de salvare.

Concluzii

Rezultate

Scopul apicației a fost gestionarea foilor de prezență printr-un algoritm automat ce extrage și recunoaște fiecare caracter din acea foaie.

Din lipsa numărului de date de antrenament pentru rețeau proprie construită, a rezultat un procentaj scăzut. Algoritmii folosiți pentru a extrage trăsături, dintr-o imagine ce reprezintă o literă, sunt Discrete cosine transform și Haar wavelet. Rețeaua neuronală este folosită ca și clasificator.

Pentru a păstra totuși scopul aplicației s-a folosit API-ul oferit de către Google, Google Vision, ce este folosit în prezent în cadrul aplicației.

În tabelul următor sunt prezentate procentele la antrenament pentru un set de date de 8000 și apoi de 18000 caractere extrase în urma algoritmului de segmentare a tabelului. Peste setul de date s-au aplicat cei doi algoritmi de extragere a trăsăturilor pentru fiecare caracter. Aceste trăsături fiind folosite la antrenarea rețelei neuronală ce clasifică fiecare caracter în funcție de trăsăturile acestuia. Etichetarea caracterelor s-a făcut manual. Etichetele fiind toate literele mari de la A-Z fără diacritice și cifrele 0-9.

Procentele de antrenare		
Număr de date antrenament	Discret cosine transform	Haar wavelet
8000	2.950%	4.89%
18000	3.339%	3.66%

Se observă o diferență în cea ce privește acuratețea rețelei pentru cele doua metode de extragere a trăsăturilor. Haar wavelet produce un procentaj mult mai mare odată ce numărul datelor de antrenament crește. Dezavantajul metodei Haar este acela că sunt trimise înspre antrenare un număr de 16000 de trăsături (40 *40), în comparație cu metoda DCT care este un compresator și trimite înspre antrenare un numar de 36 de trăsături pentru fiecare caracter.

Direcții viitoare

Din punct de vedere al aplicabilității în cadrul facultății, aplicația se poate dezvolta aplicând noi extensii ce vin în sprijinul studenților și al profesorilor. Aceste extensii se pot referi la comunicarea cu un repository pentru proiecte, detectarea plagiaturii în cadrul proiectelor, temelor etc.

Ca și un pas de pornire pentru dezvoltarea în continuare a unui recunoscător de texte automat este colectarea a mai multor date de antrenment și/sau clasificarea acestora folosind alte tipuri de învățare/clasificare cum ar fi:

1. algoritmului knn ce primește ca și trăsături întregul ansamblu de pixeli a imaginei

2. algoritmul SVM ce primește ca și trăsătri histograma (HOG)

Bibliografie

- Feature extraction methods for OCR a survey, Ovind Due Trier, Anil KJain and Torfinn Taxt, Pattern Recognition, Vol 29, No.4, 1996.
- J. Pradeepa, E. Srinivasana, S. Himavathib, "Neural Network Based Recognition System Integrating Feature Extraction and Classification for English Handwritten", International journal of Engineering,Vol.25, No. 2, pp. 99-106, May 2012
- S.G. Mallat,A Theory for Multiresolution Signal Decomposition: The Wavelet Repre-sentation, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 11, No.7, July 1989, pp.674-693.
- https://www.objectdb.com/database/overview
- https://www.cs.ucr.edu/~scharan/assets/Optical.Character.Recognitio pdf
- Discrete cosine transform, https://en.wikipedia.org/wiki/Discrete_ cosine_transform
- https://hackernoon.com/optical-character-recognition-with-google-cl
- https://en.wikipedia.org/wiki/Comparison_of_optical_character_ recognition_software
- http://nrl.northumbria.ac.uk/1908/1/Handwritten%20Arabic%20Characte 20Recognition.pdf