

UNIVERSITATEA POLITEHNICA BUCUREŞTI  
FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE  
DEPARTAMENTUL CALCULATOARE

ABD 2017 - 2019



## PROIECT DE DIZERTAȚIE

Detectarea similaritatilor intre documente

**Coordonator stiintific:**

Prof.dr.ing.Florin Radulescu

**Absolvent:**

Rotarescu Ciprian Marian

**BUCUREŞTI**

Iunie 2019

## CUPRINS

1. INTRODUCERE.....	5
2. CONTINUTUL DOCUMENTATIEI.....	6
3. CARACTERISTICILE TEHNOLOGIILOR PROPUSE PENTRU DEZVOLTARE.....	6
3.1 JAVASCRIPT.....	7
3.2 ANGULAR.JS.....	7
3.3 NODE.JS.....	8
3.4 HTML.....	9
3.5 CSS .....	10
4. IMPLEMENTARI & CONCLUZII.....	11
4.1 ALGORITMUL CREATE DE JOHN RESIG.....	11
4.2 DRAFTABLE COMPARE API.....	16
4.3 LEVENSHTEIN ALGORITM.....	18
4.4 TRIGRAM COMPARISON.....	19
4.5 COSINE SIMILARITY.....	19
4.6 JARO-WINKLER ALGORITM .....	20
5. ALEGAREA METODEI DE IMPLEMENTARE.....	22
6. CONFIGURARE BACKEND.....	24
7. APlicatie UI/UX.....	29
8. TESTE.....	32
9. CONCLUZII.....	37
10. BIBLIOGRAFIE.....	38

## LISTA FIGURIILOR, TABELELOR ȘI A PLANŞELOR

Figura 1 - NodeJs create server.....	9
Figura 2- HTML sample.....	10
Figura 3 - Algoritmul John Resig (1).....	11
Figura 4 - Algoritmul John Resig (2).....	12
Figura 5 - Algoritmul John Resig - read/upload.....	12
Figura 6 - Algoritmul John Resig - interfata.....	13
Figura 7 - Algoritmul John Resig - lista pdf.....	13
Figura 8 - Algoritmul John Resig - rezultat 1.....	14
Figura 9 - Algoritmul John Resig - rezultat 2.....	15
Figura 10 Draftable - Modificare titlu.....	16
Figura 11 Draftable - Modificare cuprins.....	17
Figura 12 Draftable - Modificare continut.....	17
Figura 13 Draftable - Modificare titlu si header.....	18
Figura 14 Algoritmul Levenshtein - formula.....	18
Figura 15 Cosine Similarity - formula.....	19
Figura 16 Cosine Similarity - exemplu.....	19
Figura 17 Cosine Similarity - cuvinte gasite.....	20
Figura 18 Cosine Similarity - numararea cuvintelor.....	20
Figura 19 Cosine Similarity - vectori.....	20
Figura 20 Algoritmul Jaro Winker- formula.....	20
Figura 21 Algoritmul Jaro Winker- exemplu.....	21
Figura 22 Algoritmul Jaro Winker- distanta Jaro.....	21
Figura 23 Algoritmul Jaro Winker- formula distantei Jaro-Winkler.....	21
Figura 24 Algoritmul Jaro Winker- distanta Jaro-Winkler.....	22
Figura 25 Draftable - prezentare.....	23
Figura 26 Draftable - credentiale api.....	23
Figura 27 Draftable - cont.....	24
Figura 28 Backend filles.....	24
Figura 29 backend implementation (1) .....	24
Figura 30 backend implementation (2) .....	25
Figura 31 backend implementation (3) .....	25
Figura 32 backend implementation (4) .....	26
Figura 33 backend implementation (5) .....	26
Figura 34 backend implementation (6) .....	27
Figura 35 backend implementation (7) .....	27

Figura 36 backend implementation (8) .....	27
Figura 37 backend implementation (9) .....	28
Figura 38 Mongodb DB .....	28
Figura 39 Mongodb collections.....	28
Figura 40 Mongodb UI - Login.....	29
Figura 41 Mongodb UI - Dashboard.....	30
Figura 42 Mongodb UI - Dashboard pdf files.....	30
Figura 43 Mongodb UI - Library.....	31
Figura 44 Mongodb UI - Create.....	31
Figura 45 Teste - "Measuring the health of open source software ecosystems Beyond".	32
Figura 46 Teste - "Measuring the health of open source software ecosystems Beyond".	32
Figura 47 Teste - " Prezentare ISBD SQL vs NoSQL " .....	33
Figura 48 Teste - " Prezentare ISBD SQL vs NoSQL " .....	34
Figura 49 Teste - " Prezentare ISBD SQL vs NoSQL " .....	35
Figura 50 Teste - "Diacritice".....	36

## **Detectarea similaritatilor intre documente**

Tema isi propune sa se finalizeze cu o dizertatie in cadrul careia sa existe si o implementare a unui program de testare a documentelor pentru a descoperi eventuale diferente intre acestea.

### **1. Introducere**

Tema de cercetare aleasa implica implementarea unei aplicatii de testare a documentelor, pentru a ajuta un user in a descoperi diferentele intre diverse doua documente. Acest lucru poate ajuta o persoana sa vada ce modificari au intervenit intr-un document intr-o perioada mare de timp (lucrare stiintifica, carti, etc).

Pentru implementarea acestei teme propunem implementarea unei aplicatii web ce permite utilizatorului sa compare fisiere de tip pdf prin care sa se deducă modificarile / similaritatile dintre acestea. Este aleasa varianta web a aplicatiei datorita accesului usor al acestuia de catre orice utilizator si datorita faptului ca nu necesita nici o instalare pe un anumit device.

Pentru aceasta aplicatie am ales sa folosesc AngularJS 1.6 ca si tehnologie principală si NodeJS ca si backend. Aceste tehnologii mentionate pot fi structurate intr-o arhitectura de model MVC (Model – View – Controller), ceea ce permite organizarea codului, separarea logicii, o optimizarea imbunatatita si de asemenea permite modificarea codului mult mai usor (atât pentru dezvoltator initial cat si pentru alți participanți).

Aceasta aplicatie poate fi folositoare pentru persoanele care doresc sa vada evolutia documentelor printr-o interfata simpla si usor de folosit iar aplicabilitatea ei poate varia. Cateva exemple pot fi scanarea a doua documente pentru a verifica similaritatea acestora, verificarea documentelor pentru a vedea diferenta intre versiunea veche si cea noua, detectarea semnelor ortografice si multe altele.

Aceasta aplicatie poate fi folosita si pentru a detecta plagiatul intr-o masura limita. Ne putem da seama de acest lucru datorita faptului ca doar textul lipsa sau modificat este evidențiat in aceasta aplicatie iar restul textului este identic. Deci prin concluzie se poate descoperi nu numai ce modificari au suferit documentul in cauza ci putem stabili si daca documentul a fost plagiat.

Plagiatul poate fi reprezentat de idei, texte, implementari sau metode create de catre o alta persoana pentru care sunt insusite de o alta persoana decat autorul original. Aceasta problema poate fi redusa cu ajutorul unor aplicatii speciale ce ajuta la detectarea textelor similare/identice iar aplicabilitatea acestor aplicatii pot fi diverse.

De asemenea doresc sa evidențiez faptul ca aceasta aplicatie nu are ca obiectiv detectarea plagiului. Există tooluri si aplicatii mult mai avansate pentru acest lucru si de asemenea mult mai avansate din punct de vedere al implementarii iar aceasta functionalitate nu reprezinta obiectul principal al acestui web app ci poate fi considerata una secundara daca chiar se doreste acest lucru.

Merita mentionat si faptul ca nu exista multe aplicatii de acest gen disponibil in mediul web iar implementarea si posibil dezvoltarea acestora poate fi considerata si o solutie financiara daca se doreste acest aspect. Insa daca se doreste acest lucru trebuie sa existe o fundatie solidă din punct de vedere al securitatii deoarece poate fi vorba de informatii confidentiale.

In paginile urmatoare sunt descrise mai multe informatii despre aceasta aplicatie web, cercetarea amanuntita folosita pentru implementare, detalii tehnice folosite in aplicatie cum ar fi backend-ul folosit, tehnologiile folosite, interfetele web disponibile precum si rezultatele obtinute in urma testelor implicate in cadrul dezvoltarii.

## **2. Continutul documentatiei**

In cadrul acestui document sunt evidențiate date tehnice legate de codul folosit (de exemplu backend-ul folosit, ce baze de date sunt folosite, backend routes pentru frontend), ce soluții au fost găsite pentru implementarea temei alese, rezultate și teste, soluția acceptată pentru implementare, implementarea soluției acceptate în aplicația de tip web și rezultatele obținute.

Tot în acest document se pot găsi algoritmii și API (application program interface) folositi în cărțire pentru a găsi o soluție ideală pentru această aplicație. Câteva exemple pot fi:

Algoritmul Resign care folosește o licență MIT și permite diferențierea textelor din două locații și produce un text final cu modificările evidențiate sub diverse culori / subliniate.

Algoritmul Trigram este folosit pentru a compara două stringuri și reprezintă secvențe de n-gram.

Algoritmul Levenshtein ce este folosită în acest algoritm mai multă diferență dintre două secvențe și în cazul nostru texte. Distanța dintre două cuvinte/texte reprezintă numărul minim de editări ce cu un singur caracter poate schimba semnificația cuvantului în altul. Denumirea algoritmului vine de la matematicianul Vladimir Levenshtein.

Algoritmul Jaro Winkler se bazează pe măsurarea distanței de editare dintre două frecvențe. Distanța Jaro-Winkler folosește o scală de prefix care oferă evaluări mai favorabile pentru siruri de caractere care se potrivește de la început pentru o lungime prestabilită.

Algoritmul Cosine Similarity ce poate fi aplicat între două stringuri și este reprezentată ca punctul de reprezentare vectorial al acestora.

Draftable Compare API ce este folosit pentru detectarea textelor dintr-un document (<https://github.com/draftable/compare-api-node-client>).

De asemenea sunt prezente toate paginile web disponibile în această aplicație și sunt explicate toate funcționalitățile pe care acestea le au (Login page, Dashboard page, Library page, Create page). Sunt expuse și screenshoots pentru fiecare pagina și secțiuni din pagina de interes în parte pentru o mai bună înțelegere a acestora.

Rezultatele sunt evidențiate atât în cadrul soluției finale de implementare (Draftable Compare API) pentru și altor algoritmi folosiți pe parcursul implementării acestei aplicații. A fost aleasă aceasta metodă deoarece a rezolvat o problemă mare în timpul implementării acestei aplicații și aceea de a fi user friendly. O aplicație web trebuie căt mai prietenoasă cu utilizatorii sau aceasta riscă să nu fie utilizată deloc indiferent căt de multă utilitatea generează aceasta.

Fiecare rezultat este explicit și sunt expuse screenshoots din aplicație pentru a oferi un punct de vedere solid pentru cel care citește.

Tot în acest document sunt explicate tehnologiile folosite, o scurtă descriere a acestora, motivul pentru care au fost folosite și în unele cazuri câteva exemple de cod pentru o mai bună înțelegere a acestora.

În ultima parte a documentației există de asemenea câteva concluzii la care s-au ajuns în urma implementării acestei aplicații și a decizilor luate pe baza experienței acumulate în tot acest timp. De asemenea sunt prezente și bibliografiile și sursele de referință luate pentru implementarea aplicației.

## **3. Caracteristicile tehnologiilor propuse pentru dezvoltare**

În acest capitol sunt prezentate caracteristicile tehnologiilor propuse pentru dezvoltarea temei propuse. Acestea sunt principalele tehnologii esențiale pentru implementarea temei de detectare a textelor plagiate.

### **3.1 Javascript**

JavaScript este folosit cel mai frecvent limbaj de programare folosit pe partea de frontend a unei aplicatii web. Cu ajutorul acestuia se pot implementa diverse interacțiuni între browser și user (butoane, evenimente la click, animații, etc), conectarea unei aplicatii cu o baza de date, transmiterea de date către o baza de date, extragerea de informații dintr-o baza de date, calcule complexe și procesari avansate ce pot crea un user experience foarte bun.

Datorita simplitatii acestui limbaj de programare s-au produs un numar mare de librarii utile, foarte multe fiind open source iar comunitatea din spatele acestora fiind una semnificativa ajuta dezvoltatea aplicatiilor web.

Limbajul de programare Javascript poate fi folosit și în afara browserului web. Un exemplu poate fi dat de către cei de la Netscape ce poate fi folosit ca limba CGI ca și Perl sau ASP. Un alt exemplu poate fi dat de NodeJS ce folosește sintaxa de javascript pentru a genera programe software avansate și să folosească resursele calculatorului.

Acest limbaj de programare poate fi confundat cu Java datorită similarității denumirii însă trebuie specificat acestea sunt două limbi de programare diferite.

Cu toate că denumirea acestor două limbi de programare sunt asemănătoare, Javascriptul este un limbaj de scripting ce este folosit pentru a reda funcționalitate paginilor web în timp ce Java este un limbaj de programare real cu sintaxă diferită și funcționalitate diferită.

Java și JavaScript au ca origine limbi de programare C și C ++ dar direcția luată de către acestea sunt total diferite.

JavaScript nu este un limbaj de programare strict (există însă TypeScript care poate ajuta în sensul acesta) deoarece nu trebuie să declari tipul variabilelor sau re tip de date trebuie să returneze o metodă. Există însă TypeScript care ajută cu această problemă și TypeScriptul poate fi înțeleasă ca un limbaj JavaScript cu îmbunătățiri pentru a scrie cod mai precis.

Toate browserele moderne acceptă JavaScript cu ajutorul interprétilor încorporați.

JavaScript este aproape în întregime bazat pe obiecte. Acest lucru poate fi evidențiat prin faptul că orice string, număr, boolean au metode ce pot fi folosite pentru a obține informații suplimentare sau operații complexe. Un obiect de JavaScript poate fi creat cu ajutorul unei sintaxe simple cum ar fi acoladele "let object = {}". Pentru a accesa o proprietate sau metodă se accesează obiectul vizat și se folosește simbolul punct "." sau paranteze patrate "[ ]" cu denumirea metodei sau proprietății. Acestea pot fi adăugate, suprascrisă sau sterse dacă se doresc.

În JavaScript se utilizează notiunea de "prototype" pentru a transmite datele între obiectele de aceeași categorie. Dacă un obiect nu găsește o metodă sau proprietate acesta căuta în "prototype" cheia dorită.

### **3.2 Angular.JS**

Datorită popularității pe care o are JavaScript-ul și a dezvoltării acestuia de către intregi comunități s-au dezvoltat diverse framework-uri care sunt la baza acestuia JavaScript.

AngularJS este un astfel de framework care poate fi folosit pentru aplicațiile web dinamice. Vă permite să folosiți codul HTML pentru a insera logica de JavaScript în acesta prin intermediul anumite simboluri ( {{ }} ) sau comenzi speciale pentru a performa loops sau alte taskuri generale într-o aplicație (ng-repeat, ng-show, ng-hide, etc).

AngularJS "data binding" si "dependency injection" ajuta dezvoltatorul sa economiseaza timp in a scrie cod pentru tascuri generale cum ar fi actualizarea datelor atat in view cat si in controller.

Asa cum am mentionat anterior AngularJS simplifica dezvoltarea aplicatiilor prin stabilirea unor seturi de reguli si de comenzi pentru a face viata unui dezvoltator mai usoara dar trebuie stiut de la bun inceput ca acest framework este doar o solutie pentru un anumit tip de problema iar de asemenea se poate stabili ca nu orice tip de aplicatie este potrivita pentru AngularJS. AngularJS a fost construit avand ca scop construirea de aplicatii web. Din fericire, o mare parte din aplicatiile utilizate din viata de zi cu zi sunt aplicatiilor web.

Cateva din elementele cheie ale lui AngularJS sunt:

Data bindings, acestea sunt simbolizate prin {{}} iar acestea ajuta in transmiterea de variabile/functii din controller in HTML. De exemplu daca doresti sa populezi textul unui div cu datele dintr-o variabila din angular transmiterea acestora se poate face prin :

```
$scope.name = "John Doe";  
<div>{{ name }}</div>;
```

Structuri de control DOM cum ar fi repetarea de date dintr-un array, afisarea, ascunderea segmentelor dintr-un DOM:

```
$scope.hideTemplate = false;  
<div ng-if="hideTemplate"></div>
```

In cazul de mai sus comanda ng-if poate afisa sau ascunde un continut daca continutul din "" este translatat intr-un boolean de true.

In cazul in care un ng-if este false continutul cat si elementul HTML este scos complet din DOM si acest lucru poate fi aratat cu ajutorul unui inspect dat din browser.

```
<div ng-show="hideTemplate"></div>  
<div ng-hide="hideTemplate"></div>
```

Comenzile de mai sus sunt folosite pentru a afisa sau ascunde anumite portiuni din DOM (document object model).

```
$scope.listArray = [1,2,3,4,5];  
<ul>  
    <li ng-repeat="list in listArray">{{ list }}</li>  
</ul>
```

Ng-repeat este folosit pentru a itera prin liste si de asemenea pentru afisarea datelor in DOM. Pe langa afisarea acestora se pot adauga comenzi aditionale pentru a extra indexul acestora sau diverse informatii in functie de continutul listei (list of objects).

Angular JS permite dezvoltatorului sa isi compuna singuri propriile instructiuni de Dom cu ajutorul directivelor si a componentelor.

### 3.3 NodeJS

Node.js este o platforma construita cu ajutorul limbajului de programare C++ si cu ajutorul acestuia se poate accesa fisiere de pe device-ul instalat (laptop/calculator), se poate crea fisiere/edita fisiere, se poate crea un server de backend pentru diverse aplicatii si multe altele.

Node este proiectat pentru a construi aplicații scalabile și se pot implementa aplicații de tip real-time chat cum ar fi "whatsapp". În exemplul de mai jos figurează crearea unui server care transmite "Hello World". Prin ajutorul acestui server se pot face mai multe conexiuni care sunt gestionate simultan.

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

Figura 1 - NodeJS create server

Merita menționat faptul că te poți conecta la diverse baze de date (MySQL, SQL, NoSQL) cu ajutorul pachetelor NPM și de asemenea să transmiti date către orice aplicație dacă dorești. Aceste pachete NPM sunt dezvoltate de către dezvoltatori sau echipe de dezvoltatori care rezolvă diverse probleme din viața de zi cu zi a unui developer (editare de fișiere PDF, algoritmi complexi, etc.).

De asemenea trebuie menționat că este foarte ușor să instalezi NodeJS și există multă suport în cazul în care există probleme de instalare sau altele.

### 3.4 HTML

Acronimul HTML vine de la Hypertext Markup Language și este folosit pentru a crea pagini web, orice aplicație web are cel puțin o pagină de tip HTML. HTML a fost creat în 1991 de Tim Berners-Lee, creatorul oficial și fondator al ceea ce acum știm ca World Wide Web.

Limbajul HTML constă dintr-o serie de coduri scurte care introduce într-un fișier de tip HTML pentru ca un browser să îl poată citi și interpreta codul inserat în acesta.

HTML-ul poate fi afectat de diverse plugin-uri sau framework-uri care pot rezulta în manipularea acestuia și afisarea unui conținut nou/editat.

HTML poate încorpora limbi de programare, cum ar fi JavaScript prin câteva taguri speciale (<script src="" type=""></script>), care afectează logica, comportamentul și conținutul paginilor web / aplicațiilor web. De asemenea se poate include CSS tot prin taguri speciale numite <link>.

Acet browser citeste fisierul si traduce textul intr-o forma vizibila. Pentru editarea codului pentru aceasta aplicatie am folosit Sublime Text 3 iar un cod HTML arata ca si in figura urmatoare:

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
</head>
<body>

</body>
</html>
```

Figura 2 - HTML sample

### 3.5 CSS

CSS-ul poate fi folosit in a reda continutul paginilor web un aspect vizual frumos si de asemenea pentru a imbunatati user experience-ul utilizatorilor. CSS-ul poate crea animatii vizuale semnificative insa pentru a putea permite utilizatorului sa experimenteze aceste animatii browserul utilizatorului trebuie sa fie actualizat (suportul pentru browserele vechi cum ar fi Internet Explore nu suporta toate functionalitatatile CSS-ului din ziua de azi de ex).

CSS-ul este creat pentru a genera aspect general sau specific elementelor HTML cu ar fi culoarea, fontul, font-family, etc.

Această separare poate îmbunătăți accesibilitatea conținutului, poate oferi mai multă flexibilitate și control în specificarea caracteristicilor de prezentare, permite mai multor pagini HTML să împărtășească formatarea specificând CSS relevant într-un fișier .css separat și reducând complexitatea și repetarea conținutului structural .

Sintaxa CSS-ului este asemanatoare cu obiectele de Javascript si anume ca trebuie specificat elementul / elementele HTML ce face obiectivul CSS-ului iar proprietatile ce se aplica acestui element sunt scrise sub forma de cheie - valoare. Un exemplu poate fi urmatorul "div { color: "grey" } (in acest exemplu accesam toate elementele html de tip div si schimbam culoarea acestora in gri).

CSS-ul are un set de reguli si de selectori disponibil pentru developeri pentru a targeta elementele HTML.

Selectorii se pot aplica la toate elementele de acelasi tipul html, de exemplu, toate divurile (div) sau headerele de tipul (h2).

Se pot selecta atribute speciale pe elemente html pentru a avea acces mult mai specific asupra unui elemnt html prin :

*id* : ce reprezinta un identificator unic în document

*clasă* : ce reprezinta o adnota a mai multe elemente într-un document

*element*: ce reprezinta elementul html propriu zis

Clasele și ID-urile sunt sensibile la litere mici, încep cu litere și pot include caractere alfanumerice și subliniere. O clasă se poate aplica oricărui număr de elemente de elemente. Un ID poate fi aplicat numai unui singur element.

Pseudo-clasele sunt folosite în selectorii CSS pentru a permite formatarea pe baza informațiilor care nu sunt conținute în arborele de documente. Un exemplu de clasă pseudo-utilizată :

```
:hover+hover#elementid:hover:link:visited::first-line::first-letter
```

Selectorii pot fi combinați în mai multe moduri pentru a obține o mare specificitate și flexibilitate.

Selectorii multipli pot fi uniți într-o listă distanțată pentru a specifica elementele după locație, tip de element, id, clasă sau orice combinație a acestora și ordinea selectorilor este importantă

#### 4. Implementari & concluzii

Pentru implementarea acestei teme s-a facut o cercetare mai amanuntita asupra posibilitatilor de implementare a temei si in urma acestelor exista o serie de pros/cons. In urmatoarele pagini sunt redate implementarile respective si concluziile de pe urma acestora.

##### 4.1 Algoritmul creat de John Resig

Pentru detectarea textelor dintr-un document am ales sa folosesc un algoritm pentru textelor creat de John Resig, care foloseste of licenta MIT. Acest algoritm permite diferențierea textelor din două locații și produce un text final cu moficările evidențiate sub diverse culori / subliniate.

În urmatoarea figura se poate observa rezultatul diferențelor detectate dintre două texte aproape similare similare.

###### First Text original text

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

###### Second Text Textele de culoare diferita sunt introduce random

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, "Inserted text" when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was "Inserted text" popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing "Inserted text" software like Aldus PageMaker including versions of Lorem Ipsum.

###### Result

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, "Inserted text".when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was "Inserted text".popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing "Inserted text" software like Aldus PageMaker including versions of Lorem Ipsum.

Figura 3 - Algoritmul John Resig (1)

In figura de mai sus sunt afisate trei texte "First Text", "Second Text" si "Result". "First Text", "Second Text" sunt interpretate ca si două documente diferite iar al treilea text reprezinta rezultatul dintre cele anterioare.

In acest test am dorit sa compar continutul din "Second text" cu "First Text". Al doilea document contine cuvinte evidențiate în culoarea mov și au textul "inserted text" ce au fost introduse special pentru a genera o comparare cu "First text".

Rezultatul produs a fost de a evidenția cuvintele introduse "inserted text" din al doilea document cu o culoare albastră și de asemenea subliniate.

De asemenea s-a facut si un test in cazul incare situatia este inversata ("First Text" contine cuvinte in plus) iar rezultatul se poate evidenția imaginea de mai jos.

#### First Text original text

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s. "Inserted text" when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was "Inserted text" popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing "Inserted text" software like Aldus PageMaker including versions of Lorem Ipsum.

#### Second Text

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

#### Result

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, "Inserted text" when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was "Inserted text"-popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing "Inserted text" software like Aldus PageMaker including versions of Lorem Ipsum.

Figura 4 - Algoritmul John Resig (2)

In acest caz rezultatul compara al doilea document cu primul iar cuvintele ce nu se regasesc in documentul al doilea sunt tăiate si evidențiate cu culoarea rosie.

Pentru continut de dimensiuni reduse acest algoritm face exact ce a fost propus dar pentru continut de dimensiuni medii/mari rezultatul nu este chiar acela dorit. In urma unor teste mai amanuntite s-a constat ca acest algoritm poate avea o marja de eroare 5%-10% aceasta fiind destul de mare (lucru evidențiat in imaginile de mai jos).

Pentru a parcurge cateva teste mai detaliate s-a creat o interfata foarte simpla ce are ca scop urmatoarele obiective:

1. Citirea unui pdf.
2. Inserarea continutului unui pdf in baza de date.
3. Vizualizarea documentelor din db sub forma de lista.
4. Folosirea algoritmului pe aceste documente.

Pentru citirea unui pdf si de inserarea continutului s-au creat doua butoane tip file ce indeplinesc acest obiectiv. Butoanele sunt evidențiate in imaginile de mai jos.



Figura 5 - Algoritmul John Resig - read/upload

De asemenea trebuie evidențiat ca atunci cand se apasă butonul "Read File", se populează elementul html de tip div cu continutul acestuia (UPLOADED PDF SAMPLE).

Din punct de vedere al functionalitatii butonul "Read File" apeleaza un api customizat pentru a citi datele din fisierul selectat. Acesta la randul lui functioneaza cu o librarie de NodeJs.

Functionalitatea butonului de "Upload File" genereaza un document similar cu cel selectat cu file upload intr-o locatie aleasa de developer. De obicei aceste fisiere sunt stocate pe un server pentru a putea fi accesate de mai multi utilizatori.

### Text detector v0.1

The screenshot shows a user interface for a PDF text extraction tool. On the left, there's a sidebar with 'Read PDF' and 'Upload PDF' sections. In the main area, under 'Text extract', there's a box titled 'Uploaded PDF sample' containing a large amount of text about NoSQL databases. To the right, under 'Documents stored in DB', there's a list of three files: 'Test 1.pdf', 'Test 3.pdf', and 'Prezentare ISBD SQL vs NoSQL.pdf'. Below this, a 'Result' section displays the message 'Identic text is %'.

Figura 6 - Algoritmul John Resig - interfata

Pentru vizualizarea documentelor din baza de date am dorit sa le evidențiez sub forma unei liste în care fiecare element reprezinta un document.

### Documents stored in DB

This screenshot shows a list of documents stored in the database. It includes three items: 'Test 1.pdf', 'Test 3.pdf', and 'Prezentare ISBD SQL vs NoSQL.pdf'. Each item has a small thumbnail icon to its left.

Figura 7 - Algoritmul John Resig - lista pdf

De asemenea trebuie evidentiat ca s-a introdus un eveniment la click pe aceste elemente. Atunci cand se face un click pe acest element el, frontend-ul compara textul din sectiunea "Uploaded PDF sample" cu continutul ascuns din elementul din lista selectat.

Din punct de vedere functional cand se compara cele doua fisiere se apeleaza algoritmul John Resig din backend si genereaza un rezultatul sub forma unui string. Acest string este poate fi evidentiat prin culoare rosie, albastra sau neagra (default). Fiecare culoare reprezinta faptul ca exista diferente intre cele doue documente:

Rosu - reprezinta text lipsa;

Albastru - inseamna text extra;

Negru - default ( exista in ambele documente );

Text detector v0.1

Figura 8 - Algoritmul John Resig - rezultat 1

In imaginea de mai sus este selectat un pdf din baza de date ce contine acelasi continut cu un fisier pdf selectat din sistemul de operare. In urma analizei s-a constatat ca textul/continutul este 100% identic. Daca ar fi fost inserat intr-unul din documente un text ce nu se regaseste in celalalt fisier acesta ar fi fost evidențiat cu text roșu/albastru în funcție de ce fisier se compara primu.

In experimentul urmator folosind aceeasi pasi dar un alt obiect din baza de date avem urmatorul rezultat.

In acest experiment am inserat un obiect cu text initial identic cu cel vizibil in partea stanga a aplicatie si modificat ulterior cu text random.

In urma analizei se poate vedea ca doar 61% din text este identic cu fisierul propus pentru testare.

Textul modificat/inserat este evideniat in partea dreapta a aplicatie cu o culoare rosie / albastra in functie situatia textului (daca acesta a fost taiat dintr-un document sau acesta a fost completa in celalalt).



Figura 9 - Algoritmul John Resig - rezultat 2

Dupa o analiza mai detaliata al continutului am constatat ca exista o marja de eroare intre procentajul dat de algoritm 61% si realitate.

Acum lucrul este posibil datorita analizei zonei evidente in partea dreapta. Acum lucru indica faptul ca acest algoritm functioneaza mai bine pe documentele cu text scurte decat cele indicate pentru tema aleasa.

#### 4.2 Draftable Compare API

Pentru detectarea textelor dintr-un document am ales sa folosesc un API popular numit "Draftable Compare API" (<https://github.com/draftable/compare-api-node-client>).

Acet API vine si cu un demo pentru a vizualiza rezultatul dat de catre acesta librarie.

Rezultatul dat de catre dezvoltatori sta la baza compararii a doua fisiere(unul pdf si altul rtf). numit "left.rtf" si "right.pdf". Intre aceste doua fisiere sunt evidenitate diferențele de cuvintelor dintre cele doua fisiere prin nuantare de background al cuvintelor lipsa sau adaugate (rosu/verde).

Urlurile pentru aceste doua fisiere sunt urmatoarele: 1.<https://api.draftable.com/static/test-documents/code-of-conduct/left.rtf>;

2.<https://api.draftable.com/static/test-documents/code-of-conduct/right.pdf>

In Imaginile de mai jos sunt evidenitate modificarile pe care acestea le au:

1.Modificari la baza titlului ("STANDARDS/STATEMENT, ETHICS/STANDARDS");

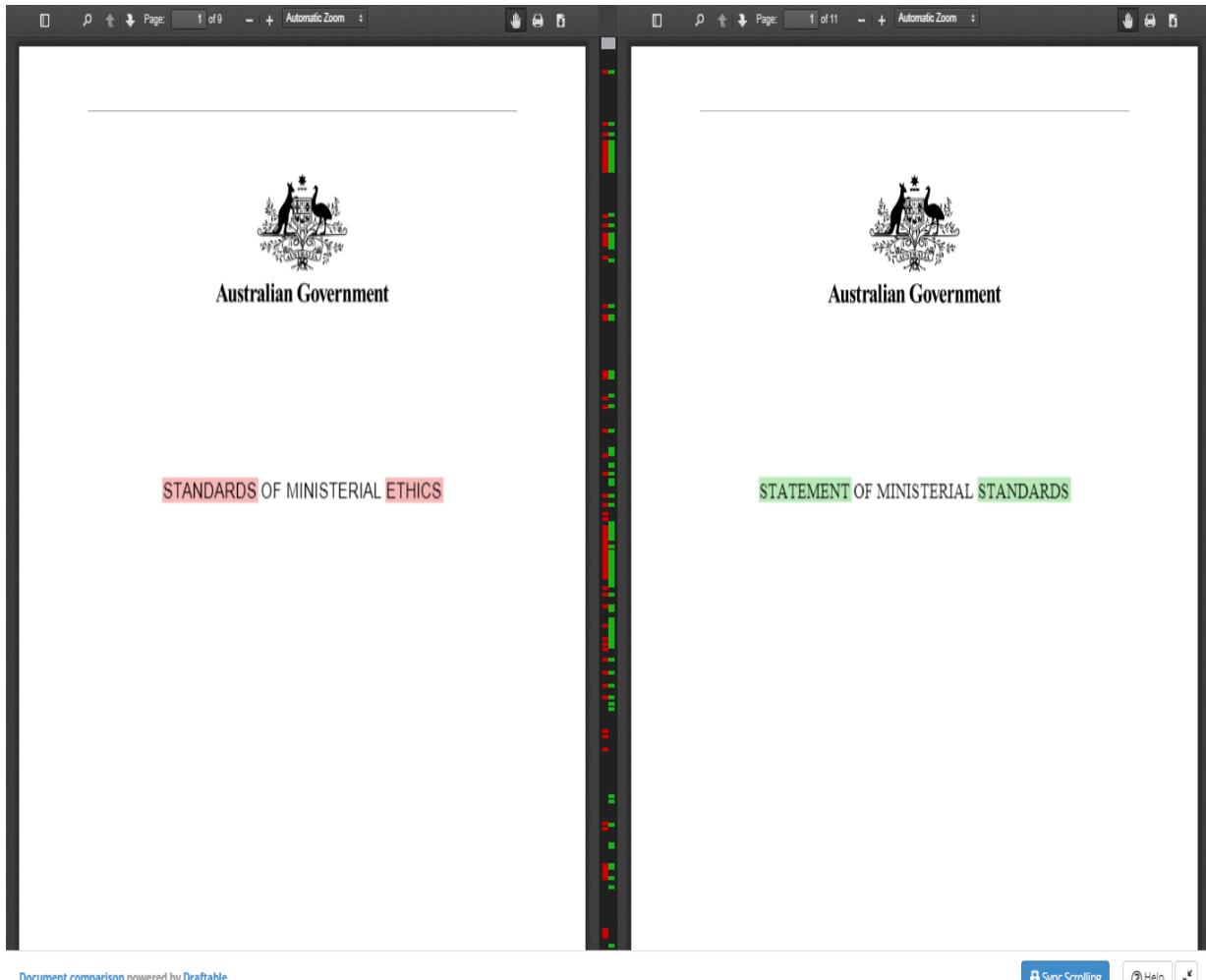


Figura 10 Draftable - Modificare titlu

## 2. Modificari in baza cuprinsului;

Contents	Page No	Page No	
FOREWORD	1	FOREWORD	1
<b>STANDARDS OF MINISTERIAL ETHICS</b>	<b>2</b>	<b>STATEMENT OF MINISTERIAL STANDARDS</b>	
1. Principles	2	1. Principles	2
2. Integrity	3	2. Integrity	3
Directorships etc	3	Directorships etc	3
Shareholdings	4	Shareholdings	4
Family members	4	Family members	4
Other forms of employment	4	Other forms of employment	4
Gifts	5	Gifts	5
Employment of family members	5	Employment of family members	5
Post-ministerial employment	5	Post-ministerial employment	5
3. Fairness	5	3. Fairness	5
4. Accountability	6	4. Accountability	6
5. Responsibility	6	5. Responsibility	6
6. The Public Interest	6	6. The Public Interest	6
7. Implementation	6	7. Implementation	6
8. Contact with Lobbyists	7	8. Contact with Lobbyists	7

Figura 11 Draftable - Modificare cuprins

## 3.Modificari in baza continutului.

**FOREWORD**

Ministers and Parliamentary Secretaries hold high public office and are entrusted with considerable privilege and power. The people of Australia are entitled to expect that, in the discharge of their duties, they will act in a manner that is consistent with the highest standards of integrity and propriety.

In 2007, the Labor Government introduced new Standards of Ministerial Ethics, requiring our Ministers to conduct themselves to a higher standard of conduct than had been the case in the past. The Standards are underpinned by the principle that Ministers and Parliamentary Secretaries must act with due regard for integrity, fairness, accountability, responsibility and the public interest.

As Prime Minister, I will do all I can to ensure that Ministers and Parliamentary Secretaries continue to live up to these high standards of conduct, and in doing so, continue to live up to the expectations of the Australian public.

Julia Gillard

**FOREWORD**

Ministers and Parliamentary Secretaries are entrusted with the conduct of public business and must act in a manner that is consistent with the highest standards of integrity and propriety. They are required to act in accordance with the law, their oath of office and their obligations to the Parliament.

In addition to those requirements, it is vital that Ministers and Parliamentary Secretaries conduct themselves in a manner that will ensure public confidence in them and in the government.

This document sets out the expected standards for that conduct.

TONY ABBOTT

Figura 12 Draftable - Modificare continut

#### 4.Modificar in baza titlurilor si header-ului

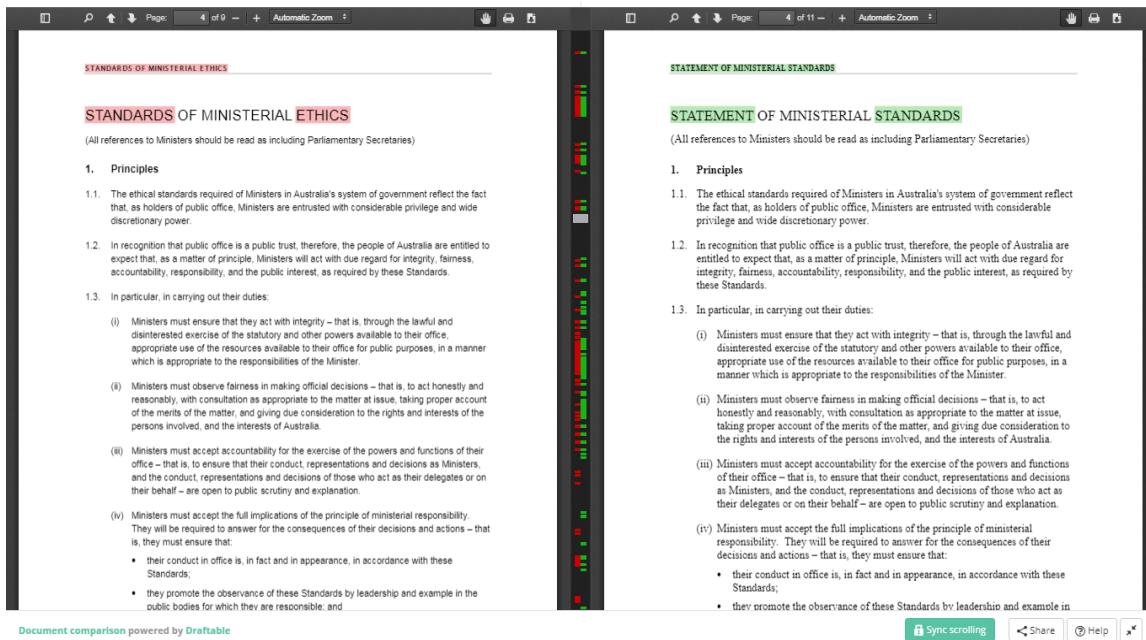


Figura 13 Draftable - Modificare titlu si header

Acest exemplu dat de catre dezvoltatori constituie exact ce ce trebuie pentru tema aleasa. De asemenea ne si arata vizual toate diferentele intre cele doua documente prin compararea pdf-urilor in paralel si sunt evidențiate toate modificările in ambele parti prin backgrounduri diferite.

Fiind spuse partile pozitive despre acesta librerie exista de asemenea si cons. Una dintre ele fiind faptul ca exista o limita de utilizari free/luna iar depasirea acestei limite necesita un abonament lunar platit pentru dezvoltatori.

#### 4.3 Algoritmul Levenshtein

Distanta Levenshtein ce este folosita in acest algoritm masoara diferenta dintre doua sevenete sau texte in cazul de fata. Distanta dintre doua cuvinte/texte reprezinta numarul minim de editari pentru un singur caracter ce poate schimba semnificatia cuvantului in altul. Denumirea algoritmului vine de la matematicianul Vladimir Levenshtein.

Formula matematica pentru distanta intre doua siruri de caractere folosind acest algoritm (a,b) este:

$$\text{lev}_{a,b}(|a|, |b|) \text{ Unde } \text{lev}_{a,b}(i, j) = \begin{cases} \max(i, j) & \text{if } \min(i, j) = 0, \\ \min \begin{cases} \text{lev}_{a,b}(i - 1, j) + 1 \\ \text{lev}_{a,b}(i, j - 1) + 1 \\ \text{lev}_{a,b}(i - 1, j - 1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

Figura 14 Algoritmul Levenshtein - formula

De exemplu, distanța dintre "pisoi/kitten" și "așezat/sitten" (in engleză) distanta Levenshtein este de 3, deoarece următoarele trei modificări se schimbă una în cealaltă și nu există nicio modalitate de a face acest lucru cu mai puțin de trei modificări:

k itten → s itten (substituirea lui "s" pentru "k")  
sitt e n → sitt i n (înlocuirea lui "i" cu "e")  
sittin → sittin g (introducerea "g" la sfârșit).

Acest algoritm este ideal pentru seturi de texte de mici dimensiuni sau medii și poate fi utilizat în aplicații de verificare a ortografiei, aplicații de corectie a gramaticii sau alte aplicații ce au ca scop traducerea textelor.

#### 4.4 Compararea Trigram

Aceasta metoda de comparare poate fi folosită pentru a compara două stringuri prin anumite sevenete numite n-gram. Pentru a demonstra acesta metoda avem cuvântul 'martha' și al doilea cuvânt 'marhta' (acest cuvânt este similar cu cel anterior doar că sunt inversate caracterele 'h' și 't') iar rezultatul este urmatorul :

Sevenetele n-gram pentru cuvântul "martha" sunt următoarele : { mar art rth tha }

Sevenetele n-gram pentru cuvântul "marhta" sunt următoarele: { mar arh rht hta }

Pentru a detecta similaritatea între cele două cuvinte/texte împartim numărul de n-grams identificate între cele două cuvinte iar în cazul de sus este 1 { mar } cu numărul de n-grams unice identificate dintre cele două cuvinte 7 { mar art rth tha arh rht hta }.

Astfel pentru cuvintele "martha" și "marhta" similaritatea între cele două stringuri este de 14% (1/7).

#### 4.5 Cosine Similarity

Cosine Similarity între două stringuri este reprezentat ca fiind punctul de reprezentare vectorial al acestora acelor stringuri.

```
similitudine  
= cos (a, b)  
= produsul_punctual (a, b) / ( norma (a) * norma (b))  
= ab / || a || * || b ||
```

Figura 15 Cosine Similarity - formula

Un exemplu pentru acest algoritm poate fi redat în forma următoare:

Julie loves me more than Linda loves me

Jane likes me more than Julie loves me

Figura 16 Cosine Similarity - exemplu

Scopul final al algoritmului este acela de a observa că de similar sunt cele două texte (ordinea acestora prea nu contează) iar din aceste texte se formează o listă cu cuvinte folosite pentru numarare.

```
me Julie loves Linda than more likes Jane
```

Figura 17 Cosine Similarity - cuvinte gasite

Dupa crearea acestui array se numara aparitia fiecarui cuvant din ambele texte.

me	2	2
Jane	0	1
Julie	1	1
Linda	1	0
likes	0	1
loves	2	1
more	1	1
than	1	1

Figura 18 Cosine Similarity - numararea cuvintelor

Cu toate acestea, nu ne interesează cuvintele dar suntem interesați de cei doi vectori verticali de numarare. De exemplu, există două exemple de "eu/me" în fiecare text. Vom decide cat de apropriate sunt aceste două texte intre ele prin calcularea unei functii a celor doi vectori, și anume cosinusul unghiului dintre ele.

Cei doi vectori formati in urma liste de mai sus sunt:

```
a: [2, 1, 0, 2, 0, 1, 1, 1]  
b: [2, 1, 1, 1, 1, 0, 1, 1]
```

Figura 19 Cosine Similarity - vectori

Aproximitatea dintre cele două texte este de 0.822 (1 reprezinta 100%).

#### 4.6 Jaro-Winkler Algoritm

Acest algoritm se baseaza pe masurarea distantei de editare dintre doua frecvente. Distanța Jaro-Winkler folosește o scăla de prefix care oferă evaluări mai favorabile pentru siruri de caractere care se potrivesc de la început pentru o lungime prestabilită.

Formula pentru aceasta distanță este sub forma următoare:

$$d_j = \frac{1}{3} \left( \frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m} \right)$$

Formula de distanță Jaro

Figura 20 Algoritmul Jaro Winker- formula

d<sub>j</sub> - reprezinta distanța Jaro;

m - reprezinta numarul de caractere care se potrivesc (intre s<sub>1</sub> si s<sub>2</sub>)

$t$  - reprezinta jumatarea din numarul de transpozitii

$s1$  - reprezinta lungimea primului string

$s2$  - reprezinta lungimea de-al doilea string

Pentru a da un exemplu o sa folosim tot cuvintele "martha" si "marhta" iar rezultatele sunt urmatoarele:

```
m = 6
t = 2/2 = 1 (2 cupluri de caractere necorespunzătoare, a 4-a și a
5-a) {t / h, h / t}
| s1 | = 6
| s2 | = 6
```

Figura 21 Algoritmul Jaro Winker - exemplu

$$dj = \frac{1}{6} (6/6 + 6/6 + (6-1) / 6) = \frac{1}{6} 17/6 = 0,944$$

Distanța de distanță = 94,4%

Figura 22 Algoritmul Jaro Winker- distanța Jaro

Formula de mai sus ne da distanța Jaro iar cu aceasta putem calcula distanța Jaro-Winkler. Similaritatea Jaro-Winker folosește o scară prefixă p care dă un rating mai favorabil la sirurile care se potrivesc de la început pentru o lungime prestatibila l.

p este un factor de scalare constant pentru cât de mult scorul este ajustat în sus pentru a avea prefixe comune. Valoarea standard a acestei constante în lucrarea lui Winkler este p = 0,1.

l este lungimea prefixului comun la începutul sirului (până la maxim 4 caractere).

$$d_w = dj + (\ell p(1 - dj))$$

Formula de distanță Jaro-Winkler

Figura 23 Algoritmul Jaro Winker- formula distanței Jaro-Winkler

Deci, înapoi la exemplul "martha" / "marhta" , să luăm o lungime de prefix de l = 3 (care se referă la "mar" ). Ajungem la:

$$dw = 0,944 + ((0,1 * 3) (1-0,944)) = 0,944 + 0,3 * 0,056 = 0,961$$

Distanța Jaro-Winkler = 96,1%

Figura 24 Algoritmul Jaro Winker- distanța Jaro-Winkler

In urma celor mentionate putem stabili ca similaritatea dintre cele doua cuvinte "martha" si "marhta" este de aproximativ 96.1%.

In urma experimentelor anterioare am optat sa folosesc libraria Draftable Compare API datorita faptului ca poate crea un user experience mai bun datorita view-ului create de API si datorita faptului ca se poate integra cu tehnologiile optate pentru crearea aplicatiei.

## 5 Alegerea metodei de implementare

In urma experimentelor anterioare am optat sa folosesc libraria Draftable Compare API datorita faptului ca poate crea un user experience mai bun datorita view-ului create de API si datorita faptului ca se poate integra cu tehnologiile optate pentru crearea aplicatiei.

Algoritmul creat de John Resign nu este recomandat pentru documente de dimensiuni medii/mari si lasa de asemenea problema aspectului vizual deoarece este foarte greu de modificat un fisier de tip pdf pentru a face un highlight la text sau de a modifica continutul acestuia.

De asemenea exista suport online pentru Draftable Compair API pentru cazul in care exista vreo problema cu acesta librerie. Oricine doreste sa implementeze lgoritmul lui John Resign nu beneficieaza nici un support sau vreo garantie.

## Draftable Compare API - Node.js Client Library

This is a thin Javascript client for Draftable's [document comparison API](#). It wraps the available endpoints, and handles authentication and signing for you. The library is available on npm as `@draftable/compare-api`.

See the [full API documentation](#) for an introduction to the API, usage notes, and other references.

### Getting started

- Sign up for free at [api.draftable.com](https://api.draftable.com) to get your credentials.
- `npm install @draftable/compare-api`
- Instantiate the client:

```
const client = require('@draftable/compare-api').client(<yourAccountId>, <yourAuthToken>);  
const comparisons = client.comparisons;
```

- Start creating comparisons:

```
comparisons.create({  
    left: {  
        source: 'https://api.draftable.com/static/test-documents/code-of-conduct/left.rtf',  
        fileType: 'rtf',  
    },  
    right: {  
        source: 'https://api.draftable.com/static/test-documents/code-of-conduct/right.pdf',  
        fileType: 'pdf',  
    },  
}).then(function(comparison) {  
    console.log("Comparison created:", comparison);  
    // This generates a signed viewer URL that can be used to access the private comparison.  
    // By default, the URL will expire in 30 minutes. See the documentation for 'signedViewerURL(...)'.  
    console.log("Viewer URL (expires in 30 min):", comparisons.signedViewerURL(comparison.identifier));  
});
```

Figura 25 Draftable - prezentare

Pentru a folosi acest API este necesar un cont de pe site-ul " <https://api.draftable.com/>". Dupa ce este creat un cont pentru development se poate accesa sectiunea "account" in care sunt stocate tokenurile pentru a folosi acest api (aceste token-uri sunt folosite in NodeJs pentru autentificare).

The screenshot shows the 'API Credentials' section of the Draftable account settings. It displays two sets of credentials: 'Testing' and 'Live'. Each set includes an 'Account ID' and an 'Auth Token'. The 'Auth Token' fields are redacted with black bars. Below each set is a 'RESET' button.

	Account ID	Auth Token
Testing	[REDACTED]	[REDACTED]
Live	[REDACTED]	[REDACTED]

Figura 26 Draftable - credentiale api

Tot de pe acest site se poate accesa documentatia la acest API si anume metodele de autentificare, tehnologiile disponibile si de asemenea resursele disponibile. Fiecare cont are un anumit numar de utilizari disponibile pe luna iar aceasta difera in functie de tipul contului (contul free are in jur de 200 de comparari de text disponibile pe luna)

The screenshot shows the 'Your Account' section of the Draftable account settings. It displays three resource usage metrics: Comparison Quota, Comparison View Quota, and Comparison Storage. Each metric is shown with a progress bar indicating usage relative to the monthly quota.

Resource	Quota	Usage
Comparison Quota	6 / 200	3%
Comparison View Quota	6 / 2000	0%
Comparison Storage	32 / 2000	1%

Figura 27 Draftable - cont

## 6. Configurare backend

Pentru ca acest proiect sa foloseasca resursele mentionate anterior partea de backend este configurata in NodeJS. Cu ajutorul acestuia se poate configura un server de backend ce poate fi folosit pentru a manipula fisiere, autentificare si multe altele.

files	04-Jan-19 1:17 PM	File folder
node_modules	09-Dec-18 11:08 AM	File folder
main	09-Dec-18 11:52 AM	JS File
package.json	09-Dec-18 11:08 AM	JSON File
package-lock.json	09-Dec-18 11:08 AM	JSON File

Figura 28 Backend files

Fisierul ce este responsabil pentru backend este main.js din folder-ul backend iar in acesta avem create un server cu cateva configurari de baza pentru server, autentificarea pentru Draftable API (e nevoie de user si parola // line 9)

```
1 const express = require('express');
2 const app = express();
3 const mongoose = require('mongoose');
4 const fs = require('fs');
5 const removeAccents = require('remove-accents');
6 // const PDFParser = require("pdf2json");
7 const pdf = require('pdf-parse');
8
9 const client = require('@draftable/compare-api').client["xxxxxx-test"], "xxxxxxxxxxxxxxxxxxxxxxxxxxxxx"];
10 const comparisons = client.comparisons;
11
12 var busboy = require('connect-busboy');
13 // let pdfParser = new PDFParser(this,1);
14 app.use(busboy());
15
16 // use it before all route definitions
17 app.use(function (req, res, next) {
18
19     // Website you wish to allow to connect
20     res.setHeader('Access-Control-Allow-Origin', '*');
21
22     // Request methods you wish to allow
23     res.setHeader('Access-Control-Allow-Methods', 'GET, POST, OPTIONS, PUT, PATCH, DELETE');
24
25     // Request headers you wish to allow
26     res.setHeader('Access-Control-Allow-Headers', 'Cache-Control, Access-Control-Allow-Headers, Origin,Accept, X-Requested-With, Content-Type, Acc
27
28     // Set to true if you need the website to include cookies in the requests sent
29     // to the API (e.g. in case you use sessions)
30     res.setHeader('Access-Control-Allow-Credentials', true);
31
32     // Pass to next layer of middleware
33     next();
34 });
35
```

Figura 29 backend implementation (1)

De asemenea in acest fisier sunt prezente modulele de nodejs folosite pentru a folosi functionalitati cum ar fi citirea de fisiere, parsarea documentelor de tip PDF, conexiunea la libraria Draftable Compare API (aceasta trebuie inca user si client ce pot fi luate numai dupa ce iti creezi cont pe site-ul producatorului), mongodb (pentru crearea si conexiunea la baza de date nosql de pe "www/mlab.com").

Tot in acest fisier sunt setate cateva configurari de CORS pentru a permite frontendul sa se conexeze la rutele de backend create. Fara acesta orice http request este sortit unei erori de conectare.

Serverul este creat cu cateva routes/rute de backend de care se poate lega frontend-ul pentru a compara fisiere, inserare useri si login:

1. Prima ruta este "/compareApi". Aceasta foloseste doua fisiere stocate pe un server online si sunt special modificate pentru a se observa modificarile pe ace acestea le au. Acest exemplu este creat de catre dezvoltatorii API-ului "Draftable" si este dat pentru a oferi un exemplu rapid al API-ului.

Aceasta metoda accepta ca si parametrii doua url-uri catre fisierele pdf ce se doresc a fi comparate. De asemenea trebuie mentionat tipul acestor fisiere ("rtf", "pdf", etc) iar rezultatul acestor computari sunt trimise catre frontend prin metoda "res.send()".

Raspunsul generat reprezinta un obiect JSON ce contine printre alte informatii un url cu documentele comparate. Acest URL este folosit in frontend prin folosirea unui element "iframe" prin setarea parametrului "src" cu URL-ul primit din backend.

```
app.get('/compareApi', function (req, res) {
  comparisons.create({
    left: {
      source: 'https://api.draftable.com/static/test-documents/code-of-conduct/left.rtf',
      fileType: 'rtf',
    },
    right: {
      source: 'https://api.draftable.com/static/test-documents/code-of-conduct/right.pdf',
      fileType: 'pdf',
    },
  }).then(function(comparison) {
    console.log("Comparison created:", comparison);
    // # This generates a signed viewer URL that can be used to access the private comparison.
    // # By default, the URL will expire in 30 minutes. See the documentation for `signedViewerURL(...)` .
    console.log("Viewer URL (expires in 30 min):", comparisons.signedViewerURL(comparison.identifier));
    res.send( {
      URL: comparisons.signedViewerURL(comparison.identifier),
      comparison: comparison
    })
  });
});
```

Figura 30 backend implementation (2)

2. A doua ruta creata este "/compareTwoFilesApi" iar cu ajutorul acesteia se pot trimite doua fisiere tip PDF de pe calculator in backend si sa se extraga diferentele dintre cele doua documente folosind metoda "comparison.create()" (file inputs).

```
app.post('/compareTwoFilesApi', function (req, res) {
  var fstream;
  var paths = [];
  req.pipe(req.busboy);
  req.busboy.on('file', function (fieldname, file, filename) {
    fstream = fs.createWriteStream(__dirname + '/files/' + removeAccents(filename));
    paths.push(__dirname + '/files/' + removeAccents(filename));
    file.pipe(fstream);
  });
  req.busboy.on('finish', function () {
    console.log( fs.existsSync(paths[0]),fs.existsSync(paths[1]) )
    if(fs.existsSync(paths[0]) && fs.existsSync(paths[1])){
      comparisons.create({
        left: {
          source: fs.readFileSync(paths[0]),
          fileType: 'pdf',
        },
        right: {
          source: fs.readFileSync(paths[1]),
          fileType: 'pdf',
        },
      }).then(function(comparison) {
        console.log("Comparison created:", comparison);
        // # This generates a signed Viewer URL that can be used to access the private comparison.
        // # By default, the URL will expire in 30 minutes. See the documentation for `signedViewerURL(...)` .
        console.log("Viewer URL (expires in 30 min):", comparisons.signedViewerURL(comparison.identifier));
        res.send( {
          URL: comparisons.signedViewerURL(comparison.identifier),
          comparison: comparison
        })
      });
    }
  });
});
```

Figura 31 backend implementation (3)

Aceasta ruta este asemanatoare cu cea mentionata anterior dar prezinta diferente. Una dintre diferente consta in faptul ca url-ul sursa al documentelor scanate nu mai sunt statice (prima metoda este folosita pentru demo iar acele link-uri sunt date chiar de catre dezvoltatori pentru a arata functionalitatea acestui API).

O a doua diferenta a acestei rute de backend consta in faptul ca se genereaza o copie identica a fisierelor folosite in comparare. Acestea pot fi salvate pe un server de preferinta pentru a putea fi mai usor accesibil utilizatorilor si pentru a putea reveni la acele fisiere intr-o alta perioada.

3. A treia ruta creata este "/login" iar cu ajutorul careia un utilizator se poate autentifica in aplicatie. Aceast cod foloseste sintaxa de mongodb pentru pentru a cauta un utilizator ce prezinta acelasi nume si parola ca si parametrii trimisi de catre utilizator din formularul din pagina de "login". Acesti parametrii pot fi extrasi din parametrul req ("request").

```
app.post('/login', function (req, res) {
  let user = new users({ username: req.body.username, password: req.body.password});
  users.findOne({ username: req.body.username, password: req.body.password}, (err, resp) => {
    console.log(err, resp)
    if (err) return res.status(500).send({user, msg: "Error", status:500});
    if (!resp) return res.status(200).send({user, msg: "No user found!", status:200});
    return res.status(200).send({data: resp, msg: "User found!", status:200});
  });
});
```

Figura 32 backend implementation (4)

4. O alta ruta este cea de "/addUser" in care un utilizator admin poate crea un alt utilizator. Acest cod foloseste formatul de "User" definit in mongoDB pentru a genera un nou utilizator in baza de date nosql.

In cazul intampinarii unei situatii neprevazute cu baza de date acest cod genereaza un raspuns diferit frontendului (cod status diferit si un mesaj diferit pentru ca logica de frontend sa apeleze logica corespunzatoare si sa avertizeze utilizatorul cu un mesaj).

```
app.post('/addUser', function (req, res) {
  let user = new users({ username: req.body.username, password: req.body.password, type: req.body.type});
  user.save(err => {
    if (err) return res.status(500).send({user, msg: "Error", status:500});
    return res.status(200).send({user, msg: "User created", status:200});
  });
});
```

Figura 33 backend implementation (5)

5. O alta ruta de backend este '/compareLocalFiles' cu ajutorul careia un utilizator poate alege doua fisiere salvate intr-o baza de date. Aceasta ruta de backend primește ca și parametrii denumirea a două fisiere ce se doresc să fie comparate.

Aceste fisiere se regăsesc în baza de date nosql și de asemenea într-un server pentru a fi folosite de către utilizator oricând doreste. Parametrii sunt luate din obiectul "req" ("request") iar în urma computării datelor sunt trimise către frontend un obiect JSON cu url-ul generat de către Draftable Compare API și conține printre alte date un url care poate fi folosit într-un "iframe" HTML.

Iframe-ul generează un view cu cele două documente și se poate observa diferențele între cele două documente salvate pe server într-un mod user friendly. Aceste URL-uri sunt salvate de către

Draftable si pot fi folosite intr-un timp mai tarziu folosind o metoda speciala (mai multe detalii in documentatia dezvoltatorilor).

```
app.post('/compareLocalFiles', function (req, res) {
  if(req.body.one && req.body.two){
    comparisons.create({
      left: {
        source: fs.readFileSync(__dirname + '/files/' + removeAccents(req.body.one)) ,
        fileType: 'pdf',
      },
      right: {
        source: fs.readFileSync(__dirname + '/files/' + removeAccents(req.body.two)) ,
        fileType: 'pdf',
      },
    }).then(function(comparison) {
      console.log("Comparison created:", comparison);
      // # This generates a signed viewer URL that can be used to access the private comparison.
      // # By default, the URL will expire in 30 minutes. See the documentation for `signedViewerURL(...)`.
      console.log("Viewer URL (expires in 30 min):", comparisons.signedViewerURL(comparison.identifier));
      res.send( {
        URL: comparisons.signedViewerURL(comparison.identifier),
        comparison: comparison
      })
    });
  }
});
```

Figura 34 backend implementation (6)

6. O alta ruta de backend este '/getDocuments' cu ajutorul careia se incarca documentele salvate intr-un DB. Aceast cod de mongodb genereaza un array de obiecte ce pot fi folosite sa generam in frontend o lista de fisiere pdf. Fiecare fisier poate fi selectat cu ajutorul unui eveniment de "click" pe elementul preferat iar tot prin intermediul acestora se pot accesa metoda anterioara de comparare a documentelor. Primul si al doilea fisier selectat au culoarea diferita (galben si respectiv verde).

Sectiunea de mai jos se poate regasi in pagina de Dashboard in partea dreapta a paginii. Practic acesta reprezinta una dintre cele doua metode de comparare a documentelor (posibil cea mai utilizata si mai user friendly) iar cea de-a doua metoda de comparare este de file upload a doua fisiere.

```
app.get('/getAllDocuments', function (req, res) {
  filesaves.find({}, function(err, text) {
    res.send(text)
  })
});
```

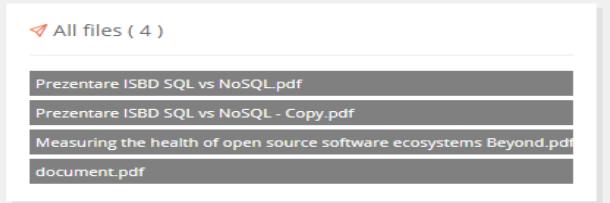


Figura 35 36 backend implementation (7)

6. O alta ruta de backend este '/uploadPDF' cu ajutorul careia se insereaza datele unui document intr-un DB. Aceasta ruta creata un fisier si il salvaza pe un server (sau local in cazul de testare) si poate fi accesibil tuturor utilizatorilor.

Tot in acest cod sunt salvate in colectia de fisiere un document cu descrierea documentului ales prin file upload.

In cazul in care exista vreo eroare in aceasta functionalitate acest cod o sa transmita catre frontend un status cod 500 si eroarea corespunzatoare. Pe partea de frontend daca exista erori utilizatorul este avertizat.

```

app.post('/uploadPDF', function (req, res) {
  var fstream;
  req.pipe(req.busboy);
  req.busboy.on('file', function (fieldname, file, filename) {
    fstream = fs.createWriteStream(__dirname + '/files/' + removeAccents(filename));
    file.pipe(fstream);
    fstream.on('close', function () {
      let dataBuffer = fs.readFileSync(fstream.path);
      pdf(dataBuffer).then(function(data) {
        let doc = new filesaves({ title: filename });
        doc.save(err => {
          if (err) return res.status(500).send(err);
          return res.status(200).send(doc);
        });
      });
    });
  });
});

```

Figura 37 backend implementation (8)

Pentru acest proiect am ales sa folosesc bazele de date NoSQL datorita faptului ca documentele pe care le dorim analizate nu pot avea un pattern concret sau o structura bine definita. Datorita acestui lucru consider ca baze de date non relationale sunt mai eficiente pentru stocarea datelor.

Am ales pentru acest proiect sa folosesc bazele de date de pe site-ul <https://mlab.com>. Cu ajutorul acestui site pot sa creez baze de date tip NoSQL pentru a stoca date si a rula cateva teste.

Pentru acest proiect am facut o baza de date numita "cheat". In aceast DB am creat o colectie pentru stocarea datelor numita "filesaves". Bazele de date NoSQL folosesc colectii in loc de tabele asa cum este traditional in SQL/MySQL.

The screenshot shows the mLab MongoDB Deployments interface. At the top, there are buttons for 'Create from backup' and 'Create new'. Below this, a table lists a single deployment:

DEPLOYMENT	PLAN TYPE	RAM	SIZE	SIZE ON DISK
ds235328/cheat	Sandbox	shared	56.92 KB	272.00 MB

Below the deployment section, there is an 'Environments' section with a 'Create new' button. A note states: 'None exist at this time. Click "Create new" to create an mLab Environment (VPC)'.

Figura 38 Mongodb DB

In aceasta baza de date avem doua colectii de date si anume filesaves si users. Filesaves contine datele despre fisierele comparate si users contine date despre conturile utilizatorilor.

The screenshot shows the mLab Collections interface. At the top, there are tabs for 'Collections', 'Users', 'Stats', 'Backups', and 'Tools'. The 'Collections' tab is selected. A warning message says: '⚠️ Sandbox databases do not have redundancy and therefore are not suitable for production. Read our documentation on [how to upgrade](#)'.

Below the tabs, there is a 'Collections' section with a 'Delete all collections' and 'Add collection' button. A table lists the collections:

NAME	DOCUMENTS	CAPPED?	SIZE
filesaves	4	false	8.42 KB
users	2	false	8.20 KB

Figura 39 Mongodb collections

## 7. Aplicatie UI/UX

In acest capitol este prezentata aplicatia din punct de vedere al user interface (UI). Pentru aceasta aplicatie am folosit libaria de css Bootstrap pentru reda un aspect vizual mai frumos si sa fie de asemenea user friendly. Paginile principale pe care aceasta aplicatie o are sunt:

### 7.1 Login

Orice aplicatie are o pagina de login iar in consecinta si aceast web app are o pagina dedicata authentificarii utilizatorului.

Este o authentificare simpla prin username/password iar aceste credentiale pot fi generate de utilizatori de tip admin.

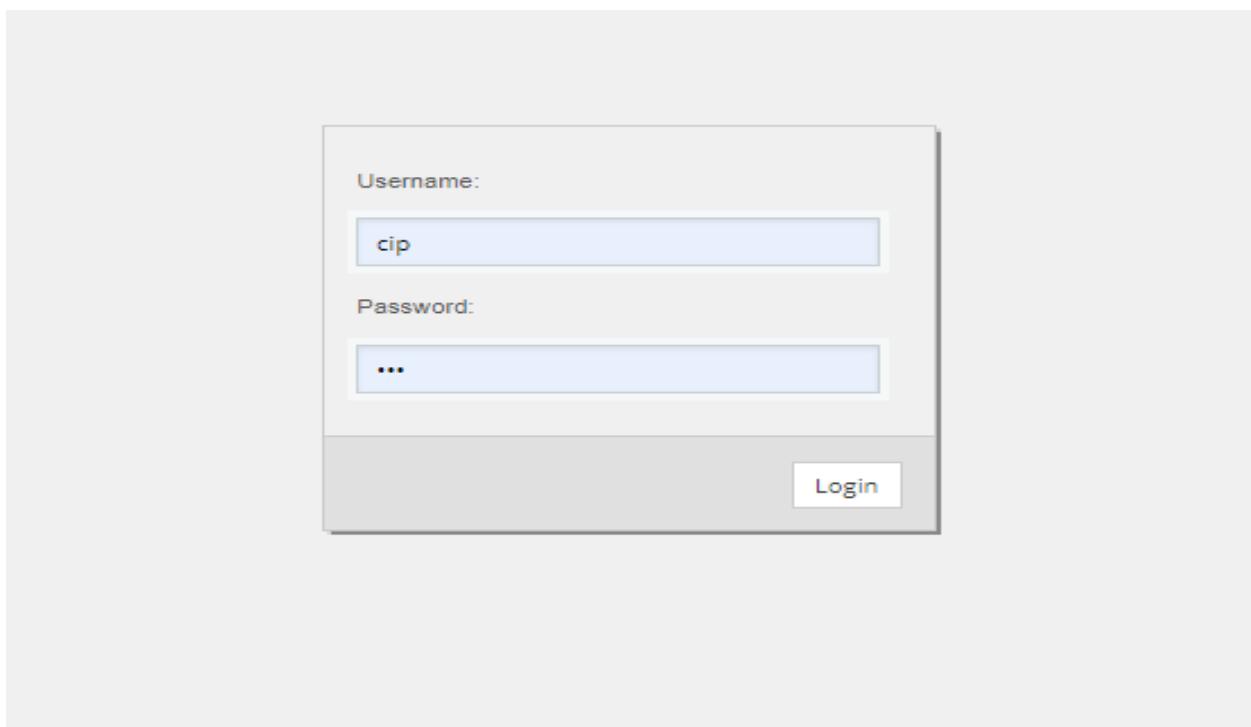


Figura 40 Mongodb UI - Login

### 7.2 Dashboard

Pagina principala a aplicatiei o reprezinta pagina "Dashboard" in care un utilizator poate sa compare documentele de tip pdf fie prin file input sau prin selectarea documentelor salvate in baza de date prin lista din stanga paginii. Ficare actiune are o sectiune separata in pagina iar din punct de vedere UX este foarte usor de interpretat chiar si fara un manual de utilizare.

In partea din stanga a paginii se afla rezultatul obtinut de catre Comparison API iar acesta dureaza cateva momente de initializare din momentul in care faci o comparare de fisiere. Pentru a adauga documente in sectiunea "All files" se acceseaza pagina "Library" din headerul aplicatiei.

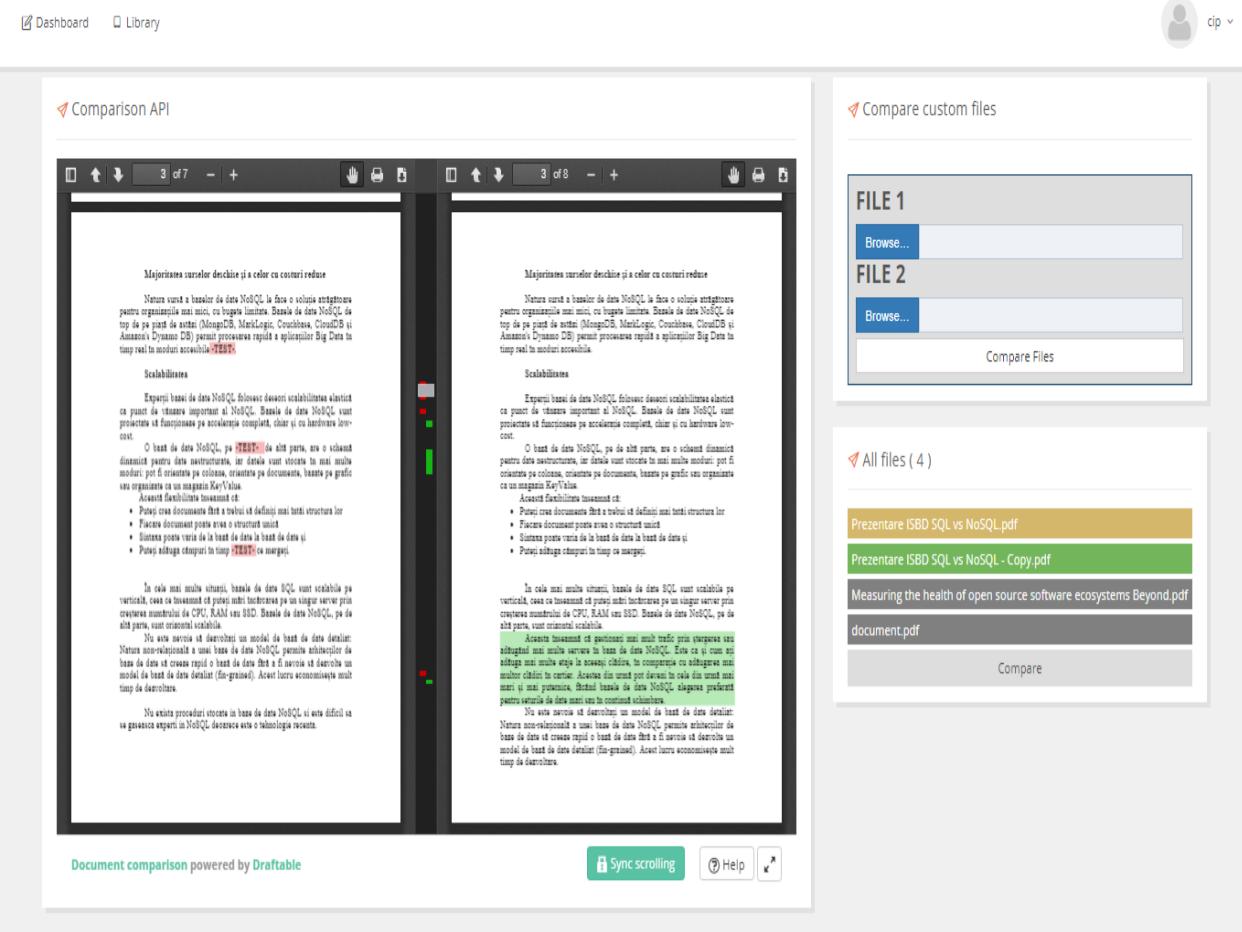


Figura 41 Mongodb UI - Dashboard

In partea de UI utilizatorul are in partea drapta din pagina de dashboard o lista in care poate selecta ce documente sa compare.

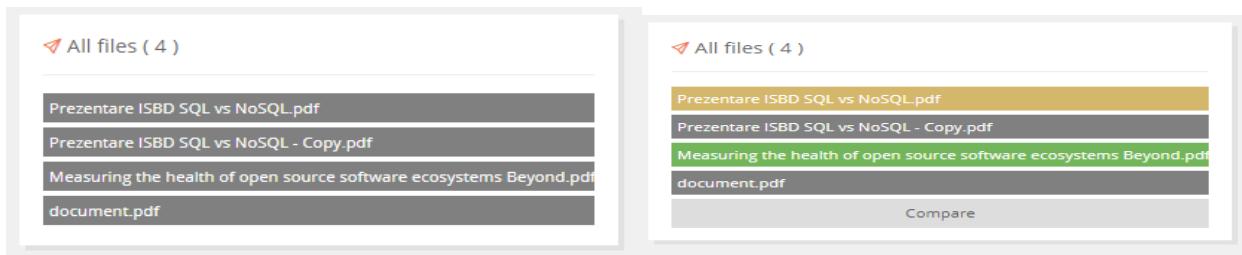


Figura 42 Mongodb UI - Dashboard pdf files

### 7.3 Library

Aceasta pagina este responsabila de a adauga datele unui fisier pdf (numele, tipul fisierului de ex) in baza de date NoSQL si de asemenea de a afisa ce fisiere sunt disponibile in DB. Pentru a adauga un fisier pdf se foloseste de formularul din aceasta pagina iar dupa ce utilizatorul a incarcat fisierul si a

apasat butonul de upload, aplicatia o sa se actualizeze si de asemenea numele fisierului trimis in baza de date este redat in lista de sub formular.

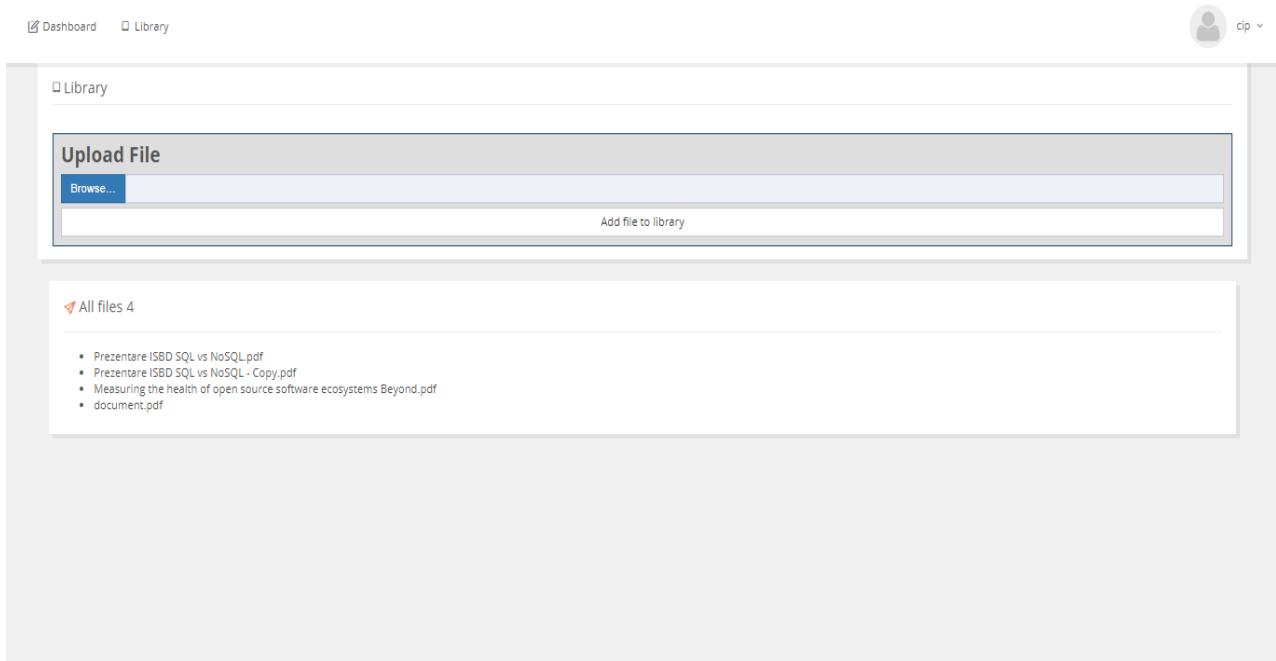


Figura 43 Mongodb UI - Library

#### 7.4 Create

Ultima setiune din aceasta aplicatie este cea de create utilizator. Aceasta sectiune este prezenta doar utilizatorilor de tip "admin". In acest formular sunt necesare numele userului nou creat, parola acestuia si de asemenea ce tip de utilizator o sa aiba contul nou creat ("normal" / "admin").

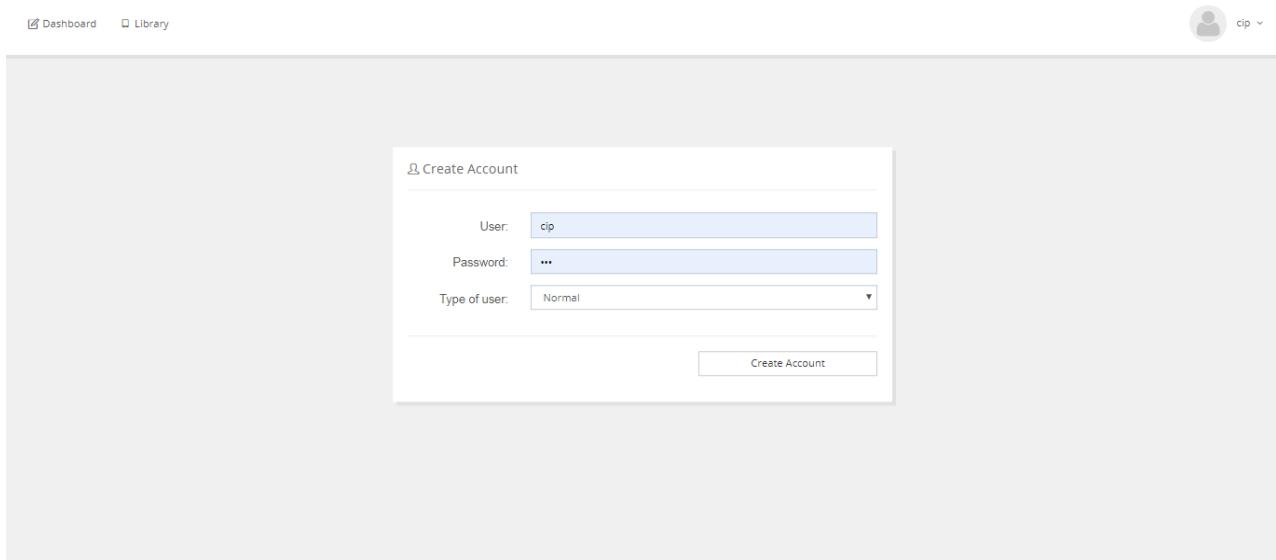


Figura 44 Mongodb UI - Create

## 8. Teste

In acest capitol sunt redate rezultatele aplicatiei implementand libraria Draftable Compare API mentionata anterior. De asemenea este bine de mentionat ca sunt folosite cateva documente aproape identice, acestea avand cateva modificarile cum ar fi cuvinte, numere sau blocuri de text modificate sau extrase.

Pentru primul test am folosit un document numit "Measuring the health of open source software ecosystems Beyond" si am create o clona al acestuia adaugand prefixul -copy la finalul documentului. Clona documentului are cateva cuvinte adăugate ("inserted text"), modificat anul de pe prima pagina (2019-2018) si sunt extrase cateva blocuri de text din document.

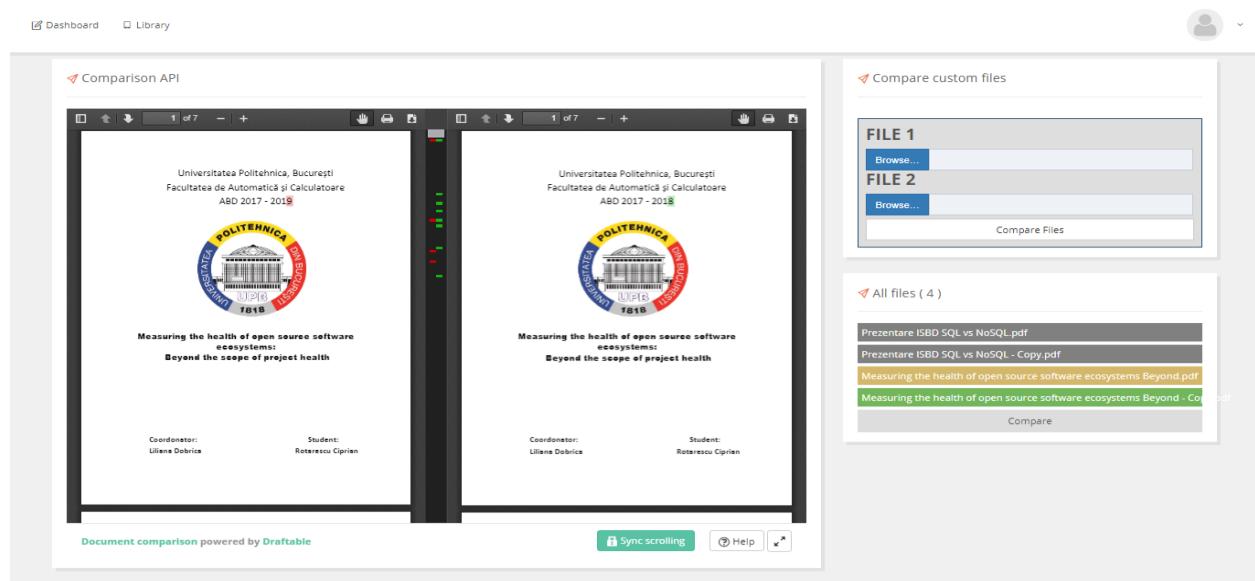


Figura 45 Teste - "Measuring the health of open source software ecosystems Beyond"

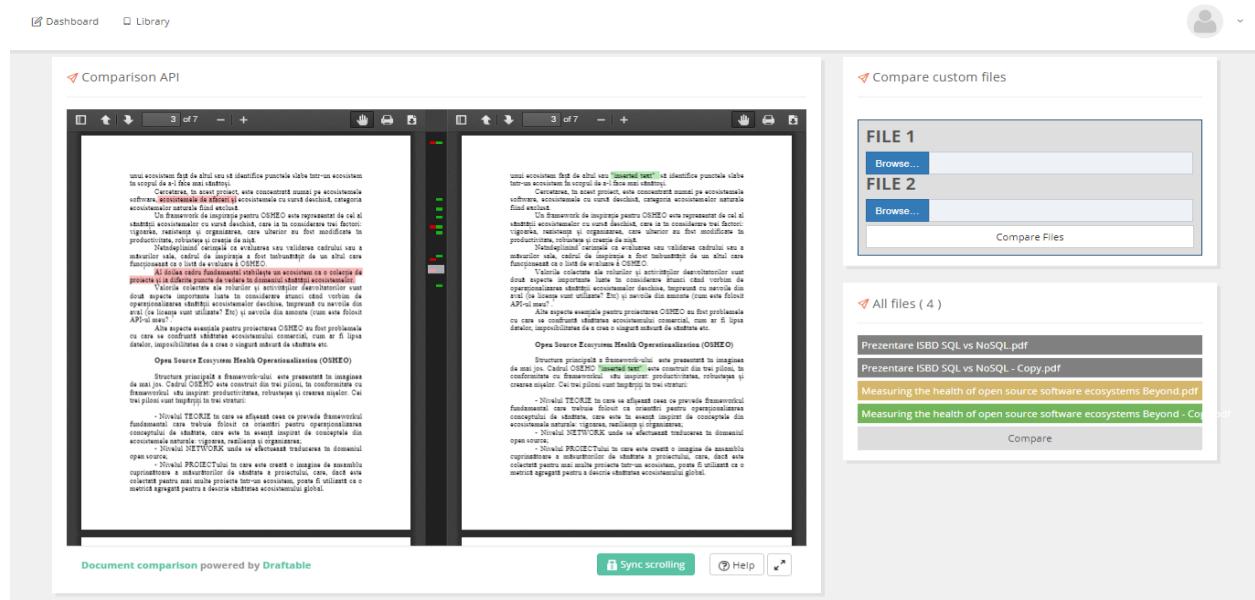


Figura 46 Teste - "Measuring the health of open source software ecosystems Beyond"

Se pot evidenția cuvintele adăugate/modificate în partea dreaptă ca au un fundal de culoare verde iar în partea stângă aceleasi cuvinte au un fundal roșu. E de menționat că și blocurile de text extrase în partea dreaptă au un fundal roșu în documentul din stânga.

Un alt exemplu pe care l-am testat cu ajutorul aplicației ce face subiectul acestui dizertatii, a fost de a compara alte două documente tip pdf numite "Prezentare ISBD SQL vs NoSQL" și "Prezentare ISBD SQL vs NoSQL - Copy".

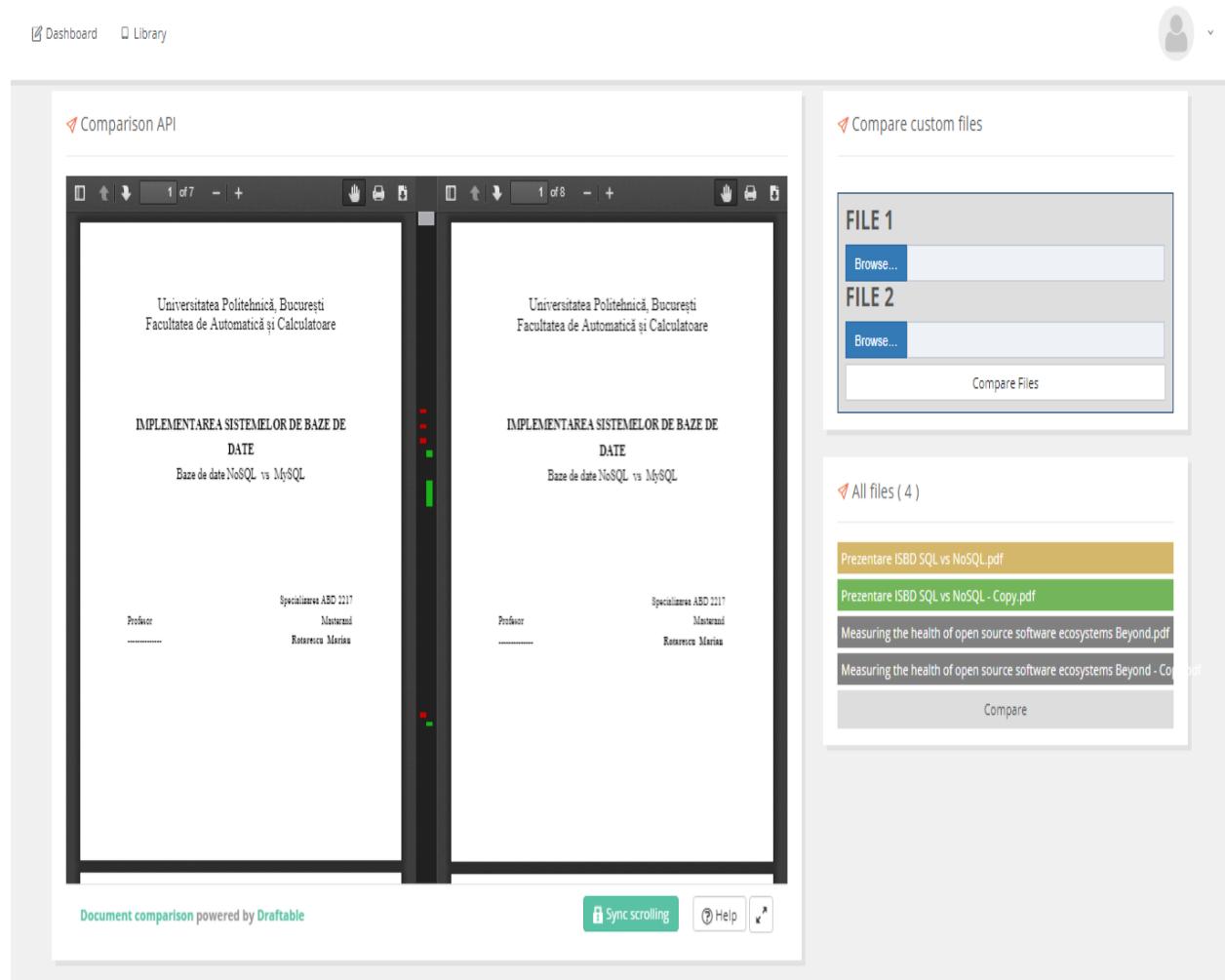


Figura 47 Teste - " Prezentare ISBD SQL vs NoSQL "

Modificările pe care le-am create pentru acest test sunt următoarele:

1. Am ales să inserez în documentul clona " Prezentare ISBD SQL vs NoSQL " texte random numite "-TEST-". Acestea sunt evidențiate cu un fundal roșu în cadrul imaginii din parte stânga.
2. Am ales să scoț portiuni mari de text din documentul clona pentru a putea fi evidențiate mai ușor. Acestea sunt evidențiate cu un fundal verde în cadrul imaginii din parte dreapta.

Cele mai multe modificări sunt evidențiate în cadrul pag 2-6. din documente. De asemenea se pot evidenția ce modificări au fost identificate dacă se observă linia despartitoare a celor două documente (sunt prezente forme verzi /rosii în funcție de ce s-a identificat ).

De asemenea exista si modul fullscreen ce poate ajuta un utilizator sa observe in mai mult detaliu modificarile observate.

Aceasta aplicatie poate sa functioneze ca un detector manual de plagiat intre doua documente.

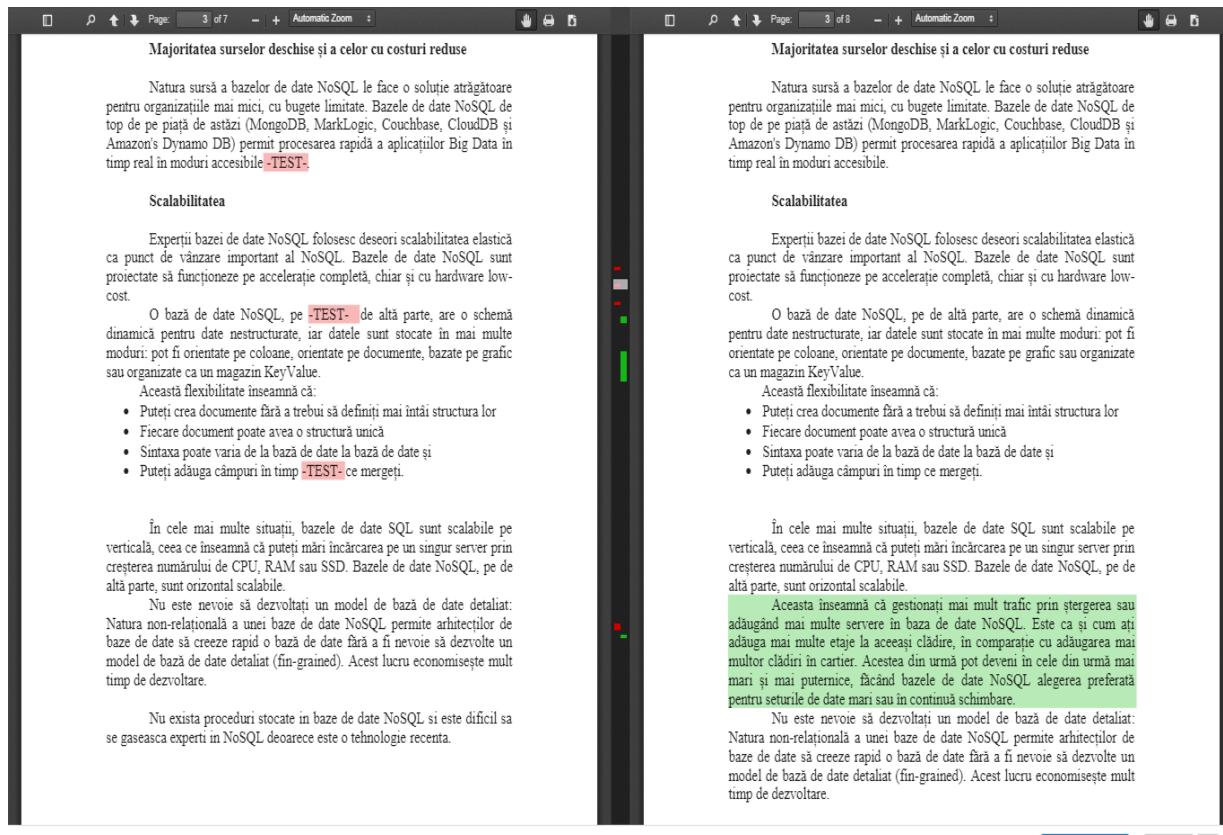


Figura 48 Teste - "Prezentare ISBD SQL vs NoSQL"

Iframe-ul generat de catre librerie doveste faptul ca genereaza o interfata foarte usoara si intuitiva pentru utilizator.

Este de mentionat faptul ca poate evidenția diferențe la diacritice pentru cuvinte (asta în cazul incare exista cuvinte aproape asemanatoare).

Se poate folosi de asemenea de butoanele generate de iframe pentru navigare, zoom si salvare daca utilizatorul doreste acest lucru. Bineînteleas ca acestea pot dispara daca se doreste acest lucru prin configurarea javascriptului.

Experimentul este asemanator cu cel anterior avand doar o simpla observatie si anume ca descrierea cuvintelor sunt scrise sub forma "-TEST-".

Există și o secțiune în plus în documentul dreapta ce este evidențiată cu fundal verde al textului:

**"Aceasta înseamnă că gestionați mai mult trafic prin stergerea sau adăugând mai multe servere în baza de date NoSQL. Este ca și cum ati adăugați mai multe etaje la aceeași clădire, în comparație cu adăugarea mai multor clădiri în cartier. Acestea din urmă pot deveni în cele din urmă mai mari și mai puternice, făcând bazale de date NoSQL alegera preferată pentru seturile de date mari sau în continuă schimbare."**

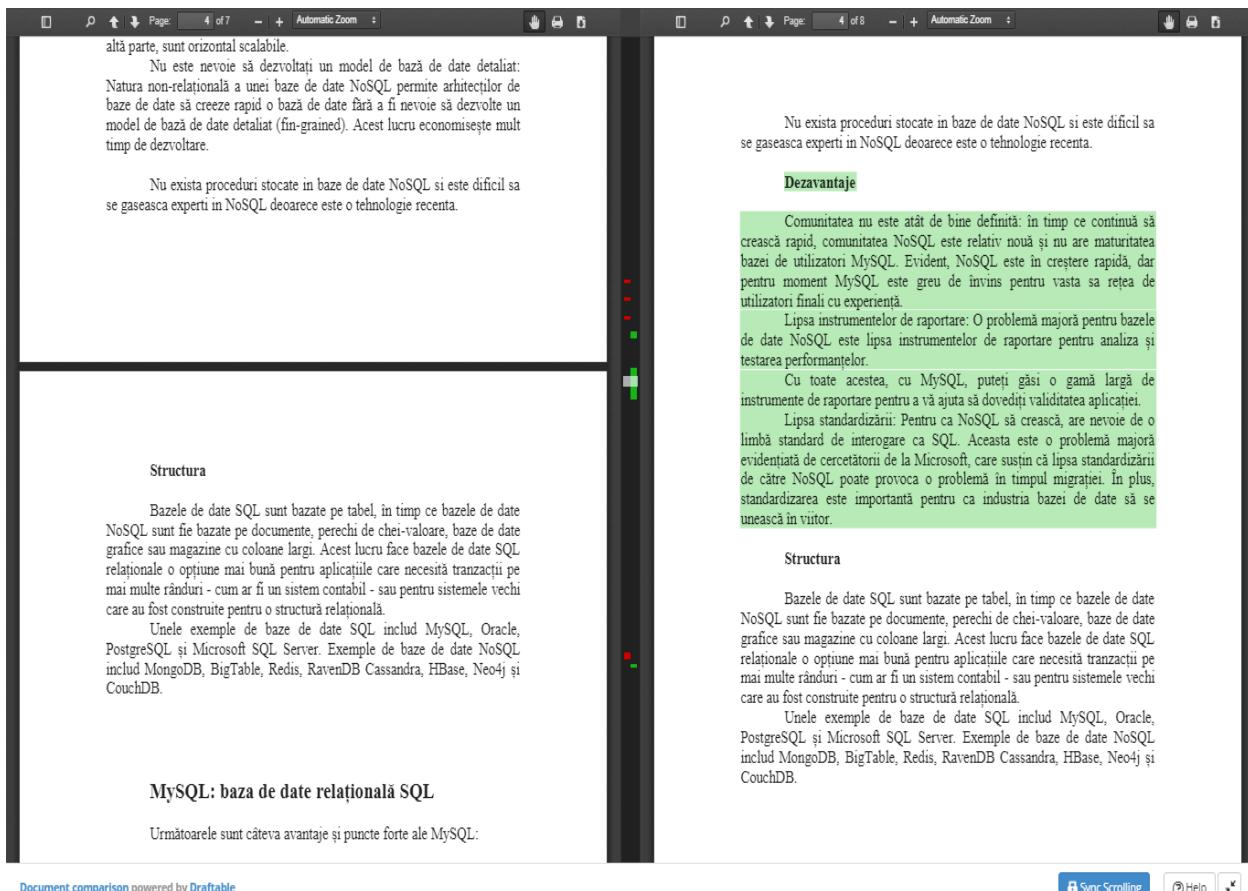


Figura 49 Teste - " Prezentare ISBD SQL vs NoSQL "

Sectiunea "Dezavantaje" a fost decupat dintr-un document si se poate observa fara probleme in imaginea de mai sus iar in urmatoarele randuri sunt redate textele introduse/decupate:

### "Dezavantaje"

**Comunitatea nu este atât de bine definită: în timp ce continuă să crească rapid, comunitatea NoSQL este relativ nouă și nu are maturitatea bazei de utilizatori MySQL. Evident, NoSQL este în creștere rapidă, dar pentru moment MySQL este greu de învins pentru vasta sa rețea de utilizatori finali cu experiență.**

**Lipsa instrumentelor de raportare: O problemă majoră pentru bazele de date NoSQL este lipsa instrumentelor de raportare pentru analiza și testarea performanțelor.**

**Cu toate acestea, cu MySQL, puteți găsi o gamă largă de instrumente de raportare pentru a vă ajuta să dovediți validitatea aplicației.**

**Lipsa standardizării: Pentru ca NoSQL să crească, are nevoie de o limbă standard de interogare ca SQL. Aceasta este o problemă majoră evidențiată de cercetătorii de la Microsoft, care susțin că lipsa standardizării de către NoSQL poate provoca o problemă în timpul migrației. În plus, standardizarea este importantă pentru ca industria bazei de date să se unească în viitor."**

De asemenea este important de mentionat ca acest API poate sa identifice textele cu diacritice pentru documentele in limba Romana.

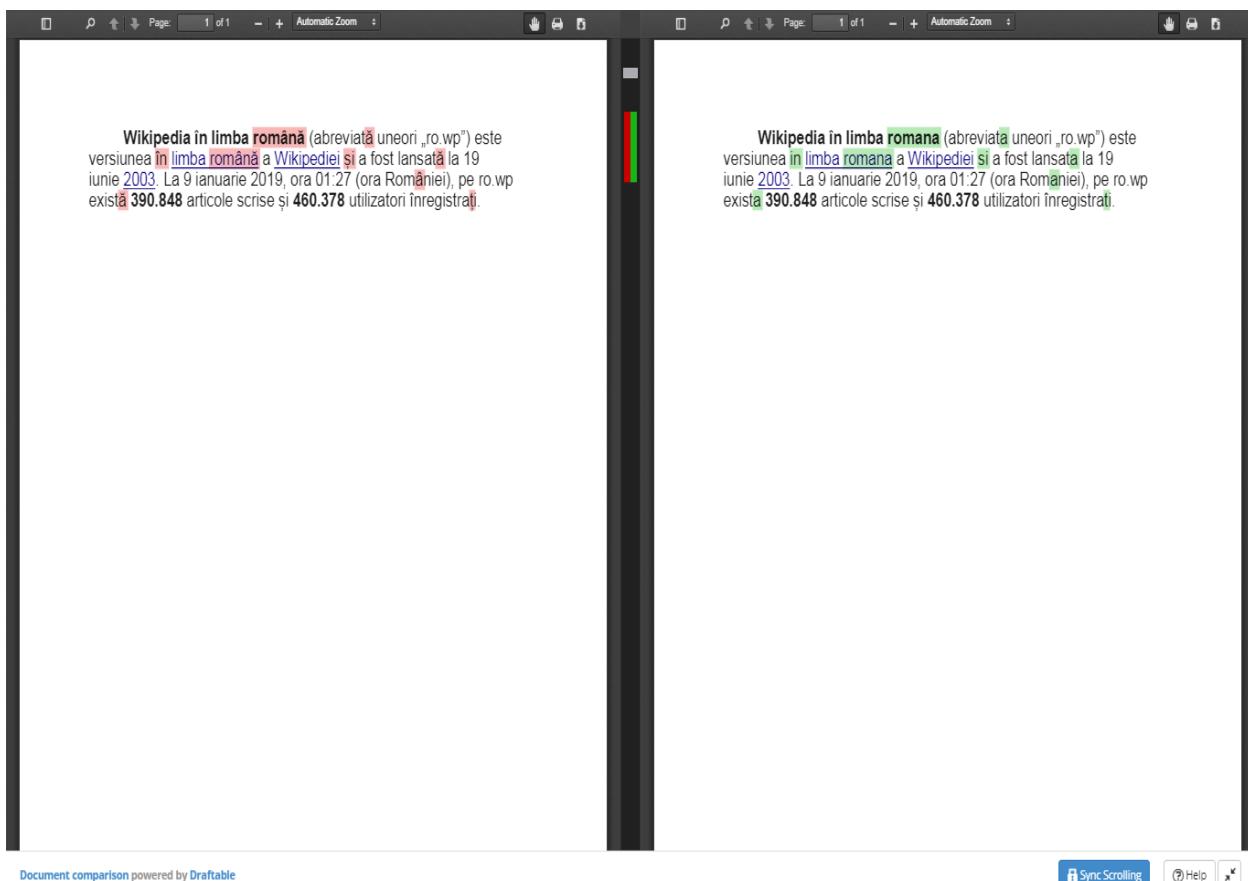


Figura 50 Teste - "Diacritice"

Modificările de cuvinte pot fi observate pentru literele cum ar fi "â", "ă", etc. Acest lucru poate fi benefic pentru lucrările în limba română. Lucrarea Draftable dovedește înțotdeauna că este alegerea ideală pentru implementarea actuală.

## **9. Concluzii**

Pentru detectarea plagiului si in urma testelor mentionate s-a dovedit faptul ca acest API - DRAFTABLE poate fi folosit pentru indeplinirea scopului propus . Există însă și un drawback legat de acest API și anume că trebuie platit pentru a avea acces la un număr ridicat de comparații și alte beneficii.

Algoritmii prezentati in capitolul 4 sunt folositori dar exista problema modului de afisare a rezultatelor. Datorita acestei probleme Draftable API este solutia cea mai rapida pentru rezolvarea atat acestui task de afisare cat si a corectitudinii datelor.

Aceasta aplicatie poate fi folositoare pentru persoanele care doresc sa vada evolutia documentelor printr-o interfata simpla si usor de folosit (de exemplu evolutia unei documentatii).

Aplicatia poate fi imbunatatita prin implementarea modului de autentificare prin email / facebook / gmail si de asemenea de inserare a altor functionalitati.

## **10. Bibliografie**

<https://api.draftable.com/>

<https://nodejs.org/en/about/>

<https://github.com/draftable/compare-api-node-client>

<https://www.quirksmode.org/js/intro.html>

[https://www.tutorialspoint.com/nodejs/nodejs\\_introduction.htm](https://www.tutorialspoint.com/nodejs/nodejs_introduction.htm)

<https://en.wikipedia.org/wiki/JavaScript>

<https://www.lifewire.com/what-is-html-3482374>

[https://en.wikipedia.org/wiki/Cascading\\_Style\\_Sheets](https://en.wikipedia.org/wiki/Cascading_Style_Sheets)

<https://docs.angularjs.org/guide/introduction>

<https://www.upwork.com/hiring/development/angularjs-basics/>

<https://medium.com/@sumn2u/string-similarity-comparision-in-js-with-examples-4bae35f13968>

[https://en.wikipedia.org/wiki/Levenshtein\\_distance](https://en.wikipedia.org/wiki/Levenshtein_distance)