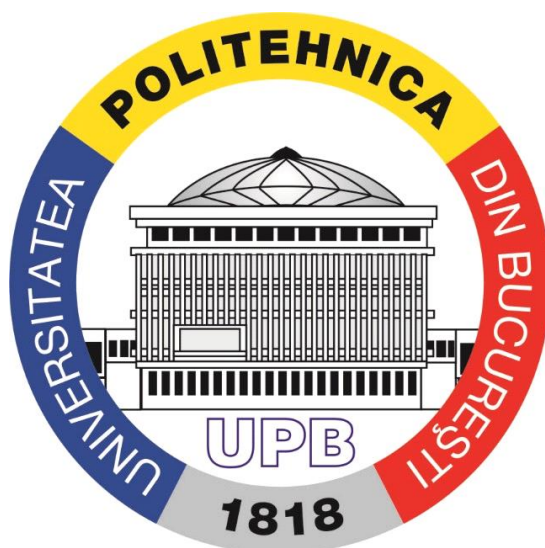


UNIVERSITATEA POLITEHNICA BUCUREȘTI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

ABD 2017 - 2019



PROIECT DE DIZERTAȚIE
Detectarea similaritatilor intre documente

Coordonator:

Prof.dr.ing.Florin Radulescu

Student:

Rotarescu Ciprian Marian

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 LEVENSHTAIN ALGORITHM.....	18
4.4 TRIGRAM COMPARISON.....	19
4.5 COSINE SILIMARITY.....	19
4.6 JARO-WINKLER ALGORITHM	20
5. ALEGEREA METODEI DE IMPLEMENTARE.....	22
6. CONFIGURARE BACKEND.....	24
7. APLICATIE UI/UX.....	29
8. TESTE.....	32
9. CONCLUZII.....	37
10. BIBLIOGRAFIE.....	38

LISTA FIGURILOR, TABELELOR ȘI A PLANȘELOR

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 aleasă implică implementarea unei aplicații 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 aplicații web ce permite utilizatorului să compare fișiere de tip pdf prin care să se deducă modificările / similaritățile 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 principala 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 (atat pentru dezvoltatorul initial cat si pentru alti participanti).

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 similiaritatea 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 evidentiat in aceasta aplicatie iar restul textului este identic. Deci prin concluzie se poate descoperi nu numai ce modificari a 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 redua cu ajutorul unor aplicatii speciale ce ajuta la detectarea textelor similare/identice iar aplicabilitatea acestor aplicatii pot fi diverse.

De asemenea doresc sa evidenziez faptul ca aceasta aplicatie nu are ca obiectiv detectarea plagiatului. Exista 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 secundata 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 acesteia poate fi considerata si o solutie financiara daca se doreste acest aspect. Insa daca se doreste acest lucru trebuie sa existe o fundatie solida 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 evidentiata date tehnice legate de codul folosit (de exemplu backend-ul folosit, ce baze de date sunt folosite, backend routes pentru frontend), ce solutii au fost gasite pentru implementarea temei alese, rezultate si teste, solutia acceptata pentru implementare, implementarea solutiei acceptate in aplicatia de tip web si rezultatele obtinute.

Tot in acest document se pot gasi algoritmii si API (application program interface) folositi in cercetare pentru a gasi o solutie ideala pentru aceasta aplicatie. Cateva exemple pot fi:

Algoritmul Resign care foloseste of licenta MIT si permite diferentierea textelor din doua locatii si produce un text final cu modificarile evidentiata sub diverse culori / subliniate.

Algoritmul Trigam este folosit pentru a compara doua stringuri si reprezinta secvente de n-gram.

Algoritmul Levenstein ce este folosita in acest algoritm masoara diferenta dintre doua secventa iar in cazul nostru texte. Distanța dintre doua cuvinte/texte reprezinta numarul minim de editari ce cu un singur caracter ce poate schimba semnificatia cuvintului in altul. Denumirea algoritmului vine de la matematicianul Vladimir Levenshtein.

Algoritmul Jaro Winkler se bazeaza pe masurarea distantei de editare dintre doua frecvente. Distanța Jar-Winkler foloseste o scala de prefix care ofera evaluari mai favorabile pentru siruri de caractere care se potrivesc de la inceput pentru o lungime prestabilita.

Algoritmul Cosine Similarity ce poate fi aplicat intre doua stringuri si este reprezentata 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 in aceasta aplicatie si sunt explicate toate functionalitatile pe care acestea le au (Login page, Dashboard page, Library page, Create page). Sunt expuse si screenshots pentru fiecare pagina si sectiuni din pagina de interes in parte pentru o mai buna intelegere a acestora.

Rezultatele sunt evidentiata atat in cadrul solutiei finale de implementare (Draftable Compare API) pentru si a altor algoritmi folositi pe parcursul implementarii acestei aplicatii. A fost aleasa aceasta metoda deoarece a rezolvat o problema mare in timpul implementarii acestei aplicatii si aceea de a fi user friendly. O aplicatie web trebuie cat mai prietenoasa cu utilizatorii sau aceasta risca sa nu fie utilizata deloc indiferent cat de multa utilitatea genereaza aceasta.

Fiecare rezultat este explicat si sunt expuse screenshots din aplicatie pentru a oferi un punct de vedere solid pentru cel care citeste.

Tot in acest document sunt explicate tehnologiile folosite, o scurta descriere al acestora, motivul pentru care au fost folosite si in unele cazuri cateva exemple de cod pentru o mai buna intelegere a acestora.

In ultima parte a documentatiei exista de asemenea cateva concluzii la care s-au ajuns in urma implementarii acestei aplicatii si a deciziilor luate pe baza experientei acumulate in tot acest timp. De asemenea sunt prezente si bibliografiile si sursele de referinta luate pentru implementarea aplicatiei.

3. Caracteristicile tehnologiilor propuse pentru dezvoltare

In 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 interactiuni intre browser si user (butoane, evenimente la click, animatii, etc), conectarea unei aplicatii cu o baza de date, transmiterea de date catre o baza de date, extragerea de informatii dintr-o baza de date, calcule complexe si 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 ajutata dezvoltarea aplicatiilor web.

Limbajul de programare Javascript poate fi folosit si inafara browserului web. Un exemplu poate fi dat de catre cei de la Netscape ce poate fi folosit ca limba CGI ca si Perl sau ASP. Un alt exemplu poate fi dat de NodeJS ce foloseste sintaxa de javascript pentru a genera programe software avansate si sa foloseasca resursele calculatorului.

Acest limbaj de programare poate fi confundat cu Java datorita similaritatii denumirii insa trebuie specificat acestea sunt doua limbaje de programare diferite.

Cu toate ca denumirea acestor doua limbaje de programare sunt asemanatoare, Javascriptul este un limbaj de scripting ce este folosit pentru a reda functionalitate paginilor web in timp ce Java este un limbaj de programare real cu sintaxa diferita si functionalitate diferita.

Java și JavaScript au ca origine limbajele de programare C și C ++ dar directia luata de catre acestea sunt total diferita.

JavaScript nu este un limbaj de programare strict (exista insa Typescript ce poate ajuta in sensul acesta) deoarece nu trebuie sa declari tipul variabilelor sau re tip de date trebuie sa returneze o metoda. Exista insa Typescript care ajuta cu aceasta problema iar Typescriptul poate fi inteles ca un limbaj Javascript cu imbunatatiri pentru a scrie cod cat mai precis.

Toate browserele web moderne acceptă JavaScript cu ajutorul interpreților încorporați.

JavaScript este aproape în întregime bazat pe obiecte. Acest lucru poate fi evidentiat prin faptul ca orice string, numar, boolean au metode ce pot fi folosite pentru a obtine informatii suplimentare sau operatii complexe. Un Obiect de javascript poate fi create cu ajutorul unei sintaxe simple cum ar fi acoladele "let object = { }". Pentru a accesa o proprietate sau metoda se acceseaza obiectul vizat si se foloseste simbolul punct "." sau paranteze patrate "[]" cu denumirea metodei sau proprietatii. Acestea pot fi adaugate, suprascrise sau sterse daca se doresc.

In Javascript se utilizeaza notiunea de "prototype" pentru a transmite datele intre obiectele de aceeasi categorie. Daca un obiect nu gaseste o metoda sau proprietate acesta cauta in "prototype" cheia dorita.

3.2 Angular.JS

Datorita popularitatii pe care o are Javascript-ul si a dezvoltarii acestuia de catre intregi comunitati s-au dezvoltat diverse framework-uri care are la baza acestuia Javascript.

AngularJS este un astfel de framework ce poate fi folosit pentru aplicațiile web dinamice. Vă permite să folosit codul HTML pentru a insera logica de javascript in acesta prin intermediul anumite simboluri ({{ }}) sau comenzi speciale pentru a performa loops sau alte taskuri generale intr-o aplicatie (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 simplifică dezvoltarea aplicațiilor 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 aplicație este potrivită 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 aplicațiilor 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, afișarea, 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 pot accesa fisiere de pe device-ul instalat (laptop/calculator), se pot 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 de transmite "hello world". Prin ajutorul acestui server se pot face mai multe conexiuni ce pot fi 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 mentionat faptul ca te poti conecta la diverse baze de date (MySQL, SQL, NoSQL) cu ajutorul pachetelor NPM si de asemenea sa transmits date catre orice aplicatie se doreste. Aceste pachete NPM sunt dezvoltate de catre dezvoltatori sau echipe de dezvoltatori ce rezolva diverse probleme din viata de zi cu zi a unui developer (editare de fisiere pdf, algoritmi complexi, etc).

De asemenea trebuie mentionat ca este foarte usor sa instalezi NodeJS si exista mult support in cazul in care exista probleme de instalare sau altele.

3.4 HTML

Acronimul HTML vine de la Hypertext Markup Language si este folosit pentru a crea pagini web, orice aplicatie web are cel putin o pagina 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.

Limba HTML consta dintr-o serie de coduri scurte introduce într-un fisier de tip html pentru ca un browser sa il poata citi si interpreta codul inserat in acesta.

HTML-ul poate fi afectat de diverse pluggin-uri sau framework-uri ce poate rezulta in manipularea acestuia si afisarea unui continut nou/editat.

HTML poate încorpora programe scrise într-o limbă de scripting, cum ar fi JavaScript, care afectează comportamentul și conținutul paginilor web. Includerea CSS definește aspectul și aspectul conținutului.

Acest browser citește fișierul și traduce textul într-o formă vizibilă. Pentru editarea codului pentru această aplicație am folosit Sublime Text 3 iar un cod HTML arată ca și în figura următoare:

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

</body>
</html>
```

Figura 2 - HTML sample

3.5 CSS

CSS-ul poate fi folosit în a reda conținutul paginilor web un aspect vizual frumos și de asemenea pentru a îmbunătăți user experience-ul utilizatorilor. CSS-ul poate crea animații vizuale semnificative însă pentru a putea permite utilizatorului să experimenteze aceste animații browserul utilizatorului trebuie să fie actualizat (suportul pentru browserele vechi cum ar fi Internet Explore nu suportă toate funcționalitățile 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 asemănătoare cu obiectele de Javascript și anume că trebuie specificat elementul / elementele HTML ce face obiectivul CSS-ului iar proprietățile ce se aplică acestui element sunt scrise sub formă de cheie - valoare. Un exemplu poate fi următorul "div { color: "grey" }" (în acest exemplu accesăm toate elementele html de tip div și schimbăm culoarea acestora în gri).

CSS-ul are un set de reguli și de selectori disponibil pentru dezvoltatori pentru a ținti elementele HTML.

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

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

id : ce reprezintă un identificator unic în document

clasă : ce reprezintă o adnotare a mai multor elemente într-un document

element: ce reprezintă 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 :

`:hovera: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 făcut 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 o licenta MIT. Acest algoritm permite diferentierea textelor din doua locatii si produce un text final cu modificarile evidentiata sub diverse culori / subliniate.

În urmatoarea figura se poate observa rezultatul diferențelor detectate dintre două texte aproape 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 introduse 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 doua 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 evidentiata in culoarea mov si au textul "inserted text" ce au fost introduse special pentru a genera o comparare cu "First text".

Rezultatul produs a fost de a evidentia cuvintele introduse "inserted text" din al doilea document cu o culoare albastra si 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 evidentia 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 taiate si evidentiata 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 constatat ca acest algoritm poate avea o marja de eroare 5%-10% aceasta fiind destul de mare (lucru evidentiat 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 butoate tip file ce indeplinesc acest obiectiv. Butoanele sunt evidentiata in imaginile de mai jos.

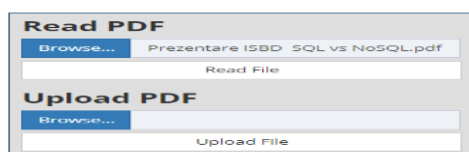


Figura 5 - Algoritmul John Resig - read/upload

De asemenea trebuie evidentiat ca atunci cand se apasa butonul "Read File", se populeaza 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

Read PDF

Browse...

Prezentare ISBD SQL vs NoSQL.pdf

Read File

Upload PDF

Browse...

Upload File

Documents stored in DB

Test 1.pdf

Test 3.pdf

Prezentare ISBD SQL vs NoSQL.pdf

Text extract

Uploaded PDF sample

Un iversitatea Politehnica, Bucuresti Facultatea de Automatica si Calculatoare IMPLEMENT AREA SISTEMELOR DE BAZE DE DATE Baze de date NoSQL vs MySQL Specializarea ABD 2017 Profesor M asterand Alexandru Boicea Rotarescu Ciprian Marian Introducere Expertii spun ca datele lumii se dubleaza la fiecare doi ani. Aceasta crestere ep ica a datelor mari din ultima vreme a evidentiat limitările dependentei de formele traditionale de stocare si gestionare a datelor si a acordat o atentie speciala noilor metode de abordare a volumului, varietatii si veridicitatii datelor structurate si nes structurate. Nu cu mult timp in urma, datele au fost stocate in fisiere fizice care au fost arhiuate in rafturi de dosare care umpleau incaperi intregi in birourile marilor corporatii. Apoi au venit computerele, iar tehnica go - to pentru stocare sa schimba t in baze de date plate. Incepand cu anii 1970, bazele de date SQL au fost parte integranta a infrastructurii IT a organizatiilor. MySQL este cea mai populara baza de date din lume si ramane atat din cauza naturii sale sursa deschisa, Tehnologia se echimba rapid si acum noua cuvant cheie din lumea bazei de date este NoSQL. Piata este una formidabila, prognoza estimata de crestere fiind de 3,4 miliarde USD in 2020, reprezentand o rata anuala de crestere compusa (CAGR) de 21% pentru perioada 2015 - 2020. NoSQL este o tehnologie bazata pe baze de date diferita de MySQL in primul rand pentru ca nu implica limba structurata a interogariilor. NoSQL : Punctele de mai jos evidentiaza unele dintre cele mai mari avantaje si dezavantaje ale NoSQL. A avantaje Non - Relational inseamna fara tabel : Bazele de date NoSQL sunt non - relationale, deci foarte diferite de bazele de date SQL. Aceasta inseamna ca sunt mai usor de gestionat si ofera un nivel mai ridicat de flexibilitate cu modelele de date mai noi. Majoritatea surselor deschise si a celor cu costuri reduse Natura sursa a bazelor de date NoSQL le face o solutie atragatoare pentru organizatiile mai mici, cu bugete limitate. Bazele de date NoSQL de top de pe piata de astazi (MongoDB, MarkLogic, Couchbase, CloudDB si Amazon's Dynamo DB) permit procesarea rapida a aplicatiilor Big Data in timp real in moduri accesibile. Scalabilitatea Expertii bazei de date NoSQL folosesc deseori scalabilitatea elastica ca punct de vanzare important al NoSQL. Bazele de date NoSQL sunt proiectate sa functioneze pe acceleratie completa, chiar si cu hardware low - cost. O baza de date NoSQL, pe de alta parte, are o schema dinamica pentru date nestructurate, iar datele sunt stocate in mai multe moduri: pot fi orientate pe coloane, orientate pe documente, bazate pe grafic sau organizate ca un magazin KeyValue. Aceasta flexibilitate inseamna ca : •

Result
Identic text is %

Figura 6 - Algoritmul John Resig - interfata

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

Documents stored in DB

Test 1.pdf

Test 3.pdf

Prezentare ISBD SQL vs NoSQL.pdf

Figura 7 - Algoritmul John Resig - lista pdf

De asemenea trebuie evidential 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.

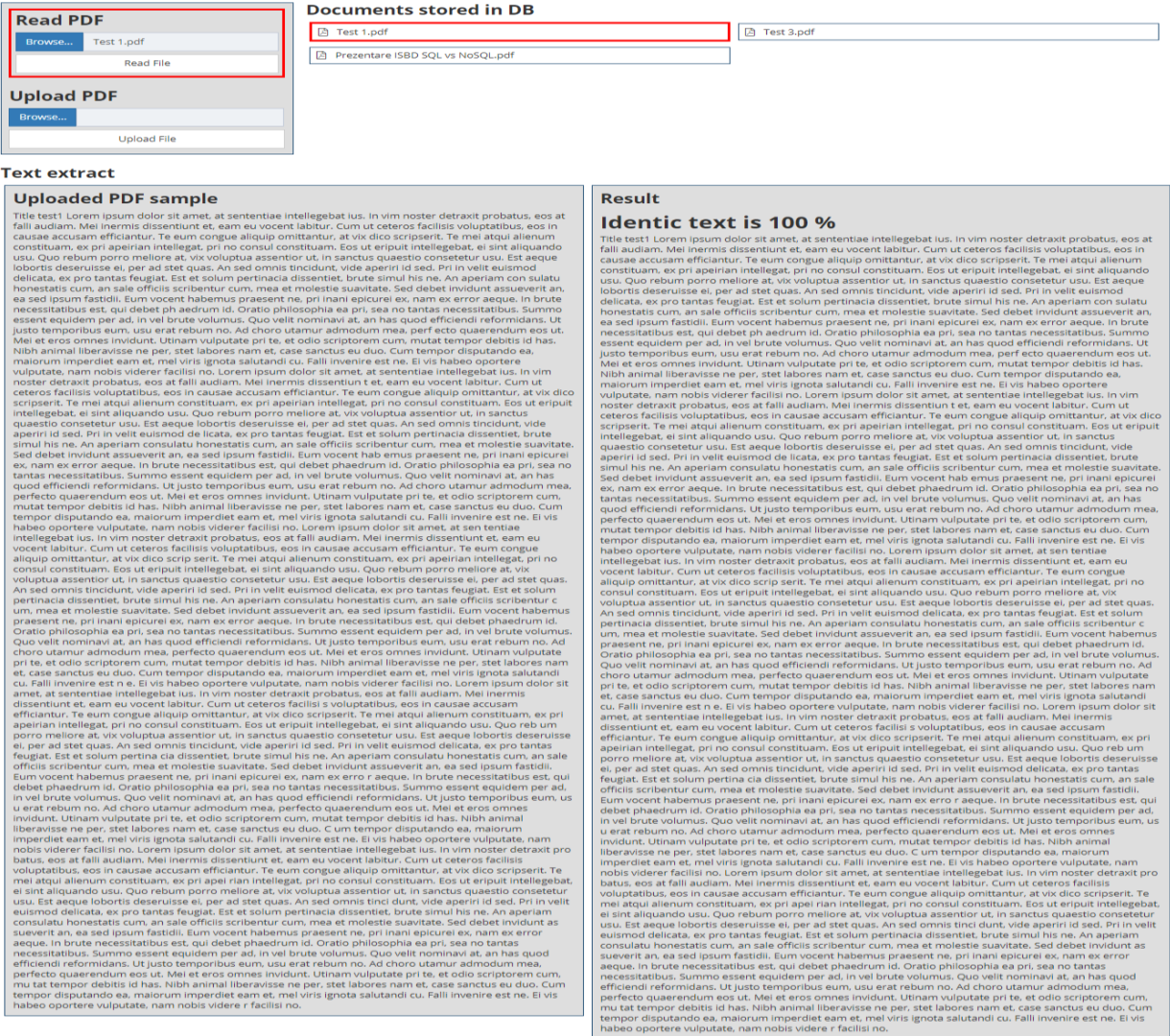
Din punct de vedere functional cand se compara cele doua fisiere se apeleaza algoritmul John Resig din backend si genereaza un rezultatul sub forma unui string. Acest string este poate fi evidential prin culoare rosie, albastra sau neagra (default). Fiecare culoare reprezinta faptul ca exista diferente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

Rosu - reprezinta text lipsa;

Albastru - inseamna text extra;

Negru - default (exista in ambele documente);

Text detector v0.1



Result

Identic text is 100 %

Title test1 Lorem ipsum dolor sit amet, at sententiae intellegbat ius. In vim noster detraxit probatus, eos at falli audiam. Mei inermis dissentiunt et, eam eu vocent labitur. Cum ut ceteros facilis voluptatibus, eos in causae accusam efficiantur. Te eum congue aliquip omittantur, at vix dico scripsit. Te mei atqui alienum constiuam, ex pri aperiam intellegat, pri no consul constiuam. Eos ut eripuit intellegbat, ei sint aliquando usu. Quo rebum porro meliore at, vix voluptua assentior ut, in sanctus quaestio consuetur usu. Est aeque lobortis deseruisse ei, per ad stet quas. An sed omnis tinctiunt, vide aperiri id sed. Pri in veilit euismod delicata, ex pro tantas feugiat. Est et solum pertinacia dissentiet, brute simul his ne. An aperiam consulu honestatis cum, an sale officis scribentur cum, mea et molestie suauitate. Sed debet inuidunt assueverit an, ea sed ipsum fastidii. Eum vocent habemus praesent ne, pri inani epicurei ex, nam ex error aeque. In brute necessitatibus est, qui debet phaedrum id. Oratio philosophia ea pri, sea no tantas necessitatibus. Summo essent equidem per ad, in vel brute volumus. Quo veilit nominavi at, an has quod efficiendi reformidans. Ut justo temporibus eum, usu erat rebum no. Ad choro utamur admodum mea, perfect o quaerendum eos ut. Mei et eros omnes inuidunt. Utinam vulpulate pri te, et odio scriptorem cum, mutat tempor debitis id has. Nibh animal liberavisse ne per, stet labores nam et, case sanctus eu duo. Cum tempor disputando ea, maiorum imperdiet eam et, mel viris ignota salutandi cu, Falli invenire est ne. El vis habeo oportere vulpulate, nam nobis viderer facilis no. Lorem ipsum dolor sit amet, at sententiae intellegbat ius. In vim noster detraxit probatus, eos at falli audiam. Mei inermis dissentiunt et, eam eu vocent labitur. Cum ut ceteros facilis voluptatibus, eos in causae accusam efficiantur. Te eum congue aliquip omittantur, at vix dico scrip serit. Te mei atqui alienum constiuam, ex pri aperiam intellegat, pri no consul constiuam. Eos ut eripuit intellegbat, ei sint aliquando usu. Quo rebum porro meliore at, vix voluptua assentior ut, in sanctus quaestio consuetur usu. Est aeque lobortis deseruisse ei, per ad stet quas. An sed omnis tinctiunt, vide aperiri id sed. Pri in veilit euismod delicata, ex pro tantas feugiat. Est et solum pertinacia dissentiet, brute simul his ne. An aperiam consulu honestatis cum, an sale officis scribentur c um, mea et molestie suauitate. Sed debet inuidunt assueverit an, ea sed ipsum fastidii. Eum vocent habemus praesent ne, pri inani epicurei ex, nam ex error aeque. In brute necessitatibus est, qui debet phaedrum id. Oratio philosophia ea pri, sea no tantas necessitatibus. Summo essent equidem per ad, in vel brute volumus. Quo veilit nominavi at, an has quod efficiendi reformidans. Ut justo temporibus eum, usu erat rebum no. Ad choro utamur admodum mea, perfect o quaerendum eos ut. Mei et eros omnes inuidunt. Utinam vulpulate pri te, et odio scriptorem cum, mutat tempor debitis id has. Nibh animal liberavisse ne per, stet labores nam et, case sanctus eu duo. Cum tempor disputando ea, maiorum imperdiet eam et, mel viris ignota salutandi cu, Falli invenire est ne. El vis habeo oportere vulpulate, nam nobis viderer facilis no. Lorem ipsum dolor sit amet, at sententiae intellegbat ius. In vim noster detraxit probatus, eos at falli audiam. Mei inermis dissentiunt et, eam eu vocent labitur. Cum ut ceteros facilis voluptatibus, eos in causae accusam efficiantur. Te eum congue aliquip omittantur, at vix dico scripsit. Te mei atqui alienum constiuam, ex pri aperiam intellegat, pri no consul constiuam. Eos ut eripuit intellegbat, ei sint aliquando usu. Quo reb uam porro meliore at, vix voluptua assentior ut, in sanctus quaestio consuetur usu. Est aeque lobortis deseruisse ei, per ad stet quas. An sed omnis tinctiunt, vide aperiri id sed. Pri in veilit euismod delicata, ex pro tantas feugiat. Est et solum pertinacia dissentiet, brute simul his ne. An aperiam consulu honestatis cum, an sale officis scribentur c um, mea et molestie suauitate. Sed debet inuidunt assueverit an, ea sed ipsum fastidii. Eum vocent habemus praesent ne, pri inani epicurei ex, nam ex error aeque. In brute necessitatibus est, qui debet phaedrum id. Oratio philosophia ea pri, sea no tantas necessitatibus. Summo essent equidem per ad, in vel brute volumus. Quo veilit nominavi at, an has quod efficiendi reformidans. Ut justo temporibus eum, usu erat rebum no. Ad choro utamur admodum mea, perfect o quaerendum eos ut. Mei et eros omnes inuidunt. Utinam vulpulate pri te, et odio scriptorem cum, mutat tempor debitis id has. Nibh animal liberavisse ne per, stet labores nam et, case sanctus eu duo. Cum tempor disputando ea, maiorum imperdiet eam et, mel viris ignota salutandi cu, Falli invenire est n e. El vis habeo oportere vulpulate, nam nobis viderer facilis no. Lorem ipsum dolor sit amet, at sententiae intellegbat ius. In vim noster detraxit probatus, eos at falli audiam. Mei inermis dissentiunt et, eam eu vocent labitur. Cum ut ceteros facilis voluptatibus, eos in causae accusam efficiantur. Te eum congue aliquip omittantur, at vix dico scripsit. Te mei atqui alienum constiuam, ex pri aperiam intellegat, pri no consul constiuam. Eos ut eripuit intellegbat, ei sint aliquando usu. Quo reb um porro meliore at, vix voluptua assentior ut, in sanctus quaestio consuetur usu. Est aeque lobortis deseruisse ei, per ad stet quas. An sed omnis tinctiunt, vide aperiri id sed. Pri in veilit euismod delicata, ex pro tantas feugiat. Est et solum pertinacia dissentiet, brute simul his ne. An aperiam consulu honestatis cum, an sale officis scribentur c um, mea et molestie suauitate. Sed debet inuidunt assueverit an, ea sed ipsum fastidii. Eum vocent habemus praesent ne, pri inani epicurei ex, nam ex error aeque. In brute necessitatibus est, qui debet phaedrum id. Oratio philosophia ea pri, sea no tantas necessitatibus. Summo essent equidem per ad, in vel brute volumus. Quo veilit nominavi at, an has quod efficiendi reformidans. Ut justo temporibus eum, usu erat rebum no. Ad choro utamur admodum mea, perfect o quaerendum eos ut. Mei et eros omnes inuidunt. Utinam vulpulate pri te, et odio scriptorem cum, mu tat tempor debitis id has. Nibh animal liberavisse ne per, stet labores nam et, case sanctus eu duo. Cum tempor disputando ea, maiorum imperdiet eam et, mel viris ignota salutandi cu, Falli invenire est ne. El vis habeo oportere vulpulate, nam nobis videre r facilis no.

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 evidentiati cu text rosu/albastru in functie 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.

Acest lucru a fost posibil datorita analizei zonei evidentiata in partea dreapta. Acest 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>).

Acest API vine si cu un demo pentru a vizualiza rezultatul dat de catre aceasta librerie.

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 evidentiata diferentele de cuvintelor dintre cele documente prin nuanzare de background al cuvintelor lipsa sau adaugate (rosu/verde).

Url-urile 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 evidentiata modificarile pe care acestea le au:

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

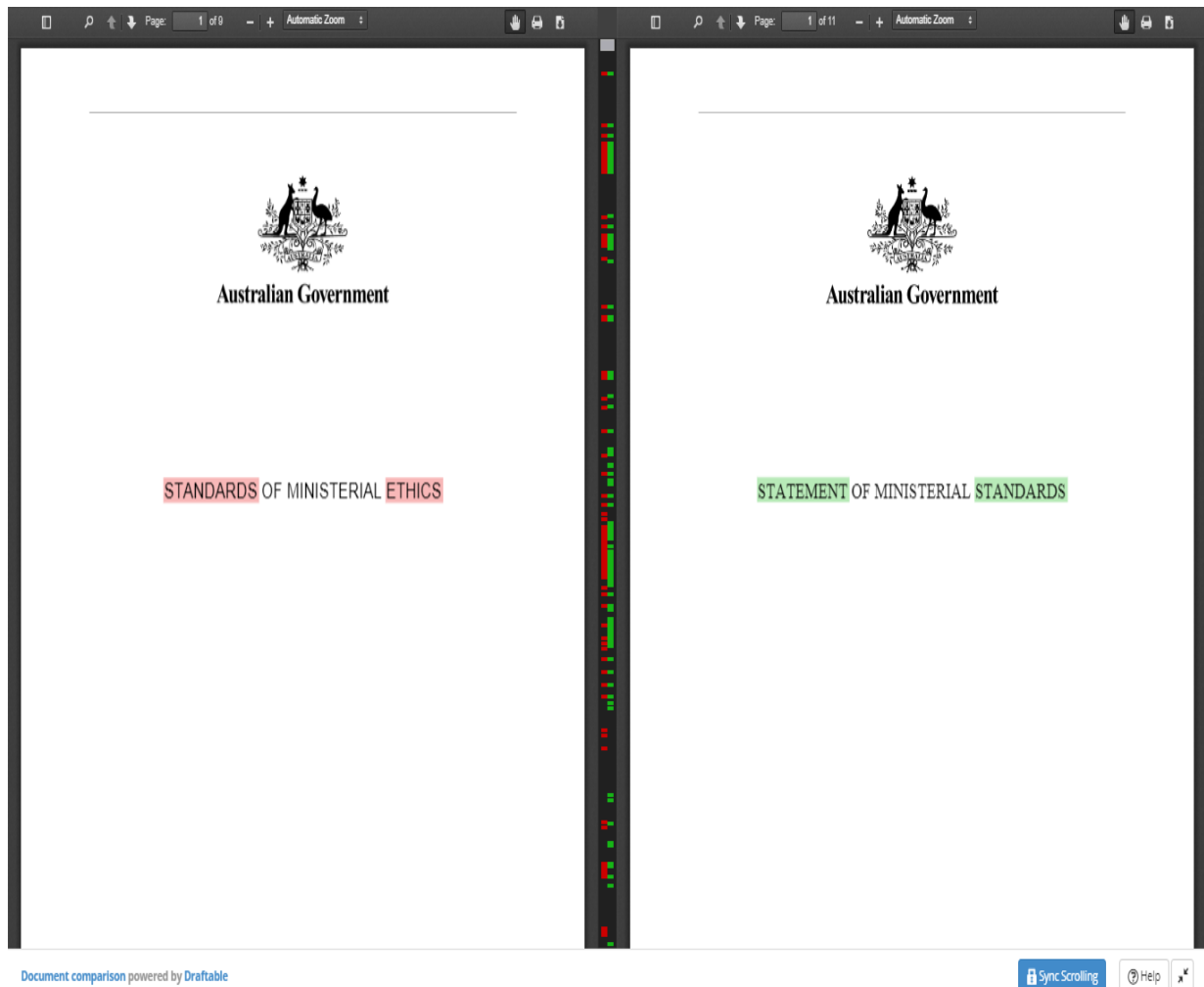


Figura 10 Draftable - Modificare titlu

2. Modificari in baza cuprinsului;

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

Document comparison powered by Draftable

Sync scrolling Share Help

Figura 11 Draftable - Modificare cuprins

3.Modificari in baza continutului.

Document comparison powered by Draftable

Sync scrolling Share Help

Figura 12 Draftable - Modificare continut

4.Modificar in baza titlurilor si header-ului

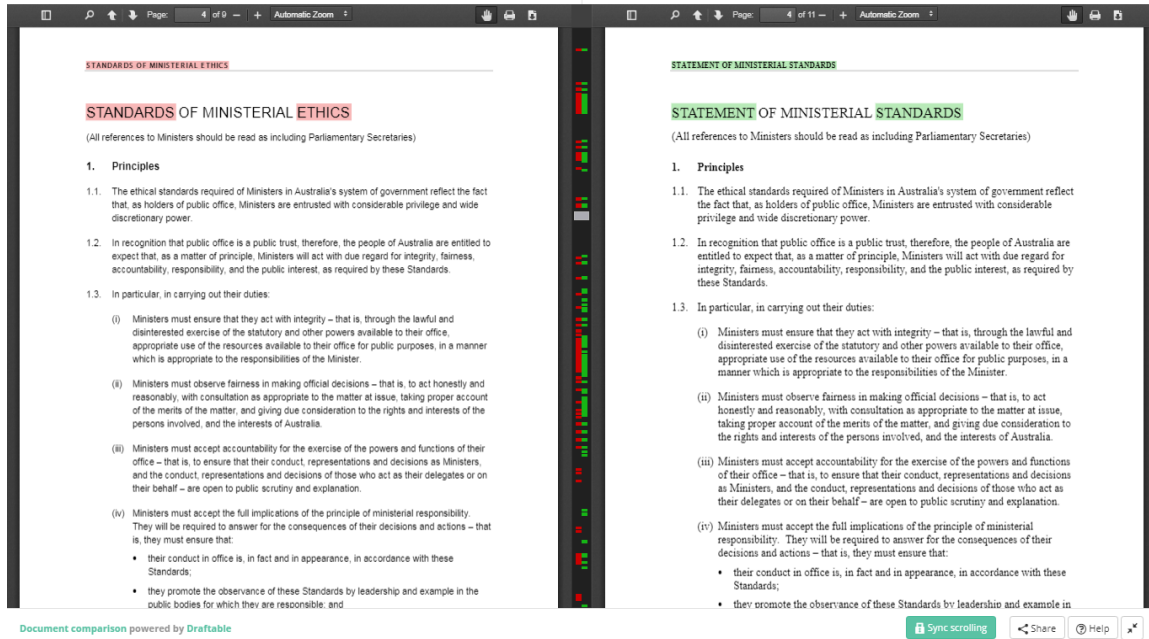


Figura 13 Draftable - Modificare titlu si header

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 evidentiata toate modificarile 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

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

Din punct de vedere matematic distanta intre doua siruri de caractere folosind acest algoritm (a,b) este:

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

Figura 14 Algoritmul Levenshtein - formula

De exemplu, distanța dintre "pisoi/kitten" și "așezat/sitten" (in engleza) 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 corectare a gramaticii sau alte aplicații ce au ca scop traducerea textelor.

4.4 Compararea Trigram

Această metodă de comparare poate fi folosită pentru a compara două stringuri prin anumite secvențe numite secvențe n-gram. Pentru a demonstra această metodă 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 următorul :

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

Secvențele n-gram pentru cuvântul "marhta" sunt: { 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 $1 / 7$ 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 reprezentată 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 cât de similare sunt cele două texte (ordinea acestora prea nu contează) iar din aceste texte se formează o listă cu cuvinte folosite pentru numărare.

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

Figura 17 Cosine Similarity - cuvinte gasite

Dupa crearea acestui array se numara aparitia fiecarui cuvant in 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. Suntem interesați doar de acei doi vectori verticali de numărare. De exemplu, există două exemple de "eu" în fiecare text. Vom decide cât de apropiate sunt aceste două texte între ele prin calcularea unei funcții a celor doi vectori, și anume cosinusul unghiului dintre ele.

Cei doi vectori formați în urma listei 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

Aproximativitatea dintre cele două texte este de 0.822 (1 reprezintă 100%).

4.6 Jaro-Winkler Algorithm

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

Formula pentru această distanță este sub formă 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 Algoritm Jaro Winkler - formula

d_j - reprezintă distanța Jaro;

m - reprezintă numărul de caractere care se potrivesc (între s_1 și s_2)

t - reprezinta jumatatea 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 = (%) (6/6 + 6/6 + (6-1) / 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 șirurile care se potrivesc de la început pentru o lungime prestabilită 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 șirului (până la maxim 4 caractere).

$$d_w = d_j + (lp(1 - d_j))$$

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

În urma celor menționate putem stabili că similaritatea dintre cele două cuvinte "martha" și "marhta" este de aproximativ 96.1%.

În urma experimentelor anterioare am optat să folosesc biblioteca Draftable Compare API datorită faptului că poate crea o experiență mai bună datorită view-ului creat de API și datorită faptului că se poate integra cu tehnologiile optate pentru crearea aplicației.

5 Alegerea metodei de implementare

În urma experimentelor anterioare am optat să folosesc biblioteca Draftable Compare API datorită faptului că poate crea o experiență mai bună datorită view-ului creat de API și datorită faptului că se poate integra cu tehnologiile optate pentru crearea aplicației.

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

De asemenea există suport online pentru Draftable Compare API pentru cazul în care există vreo problemă cu această bibliotecă. Oricine dorește să implementeze algoritmul lui John Resign nu beneficiază nici de un suport sau vreo garanție.

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 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/>". După ce este creat un cont pentru development se poate accesa secțiunea "account" în care sunt stocate tokenurile pentru a folosi acest API (aceste token-uri sunt folosite în NodeJS pentru autentificare).

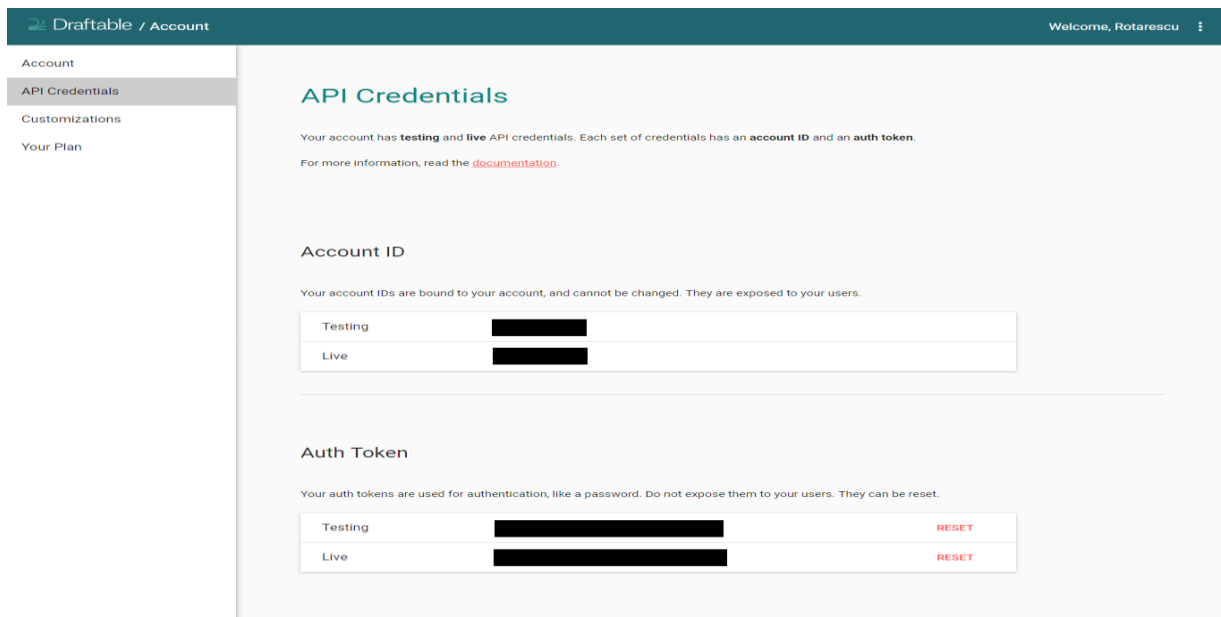


Figura 26 Draftable - credentiale api

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)

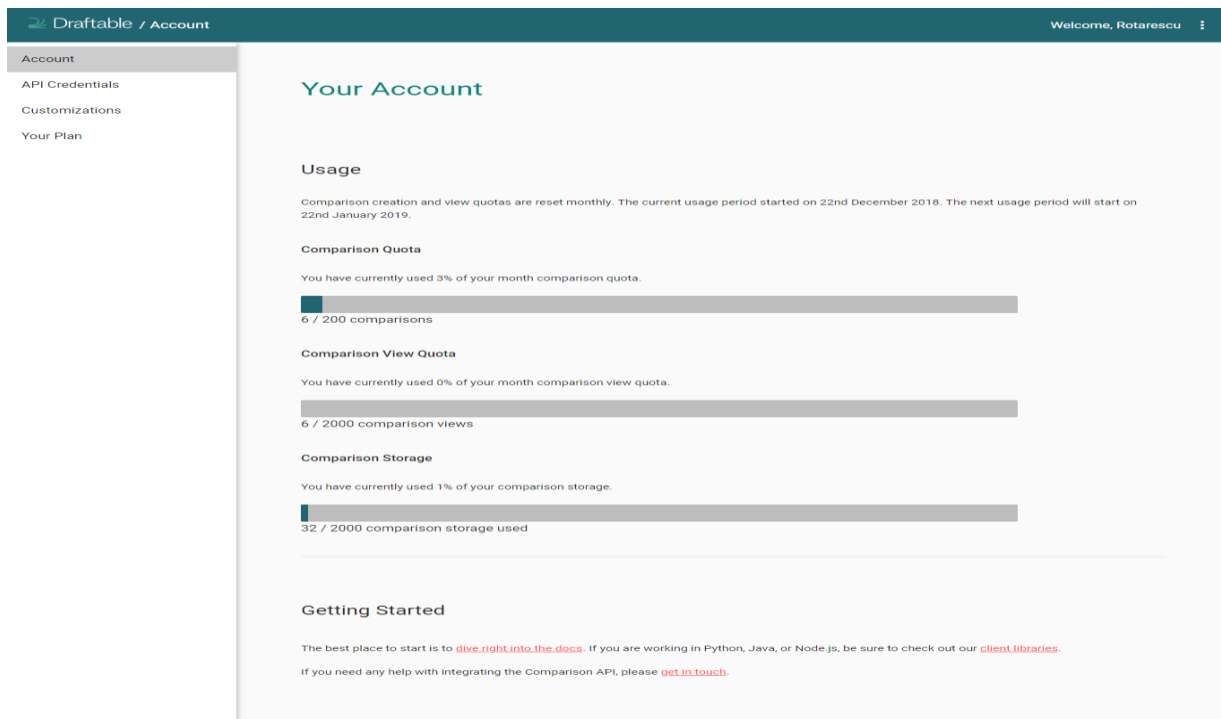


Figura 27 Draftable - cont

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	7 KB
 package.json	09-Dec-18 11:08 AM	JSON File	1 KB
 package-lock.json	09-Dec-18 11:08 AM	JSON File	26 KB

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", "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx");
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 biblioteca 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 conecteze 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 acele 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 identita 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. Acest 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 parametri 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 primeste ca si parametrii denumirea a doua fisiere ce se doresc a fi comparate.

Aceste fisiere se regasesc in baza de date nosql si de asemenea intr-un server pentru a fi folosite de catre utilizator oricand doreste. Parametrii sunt luate din obiectul "req" ("request") iar in urma computarii datelor sunt trimise catre frontend un obiect JSON cu url-ul generat de catre Draftable Compare API si contine printre alte date un url ce poate fi folosit intr-un "iframe" HTML.

Iframe-ul genereaza un view cu cele doua documente si se poate observa diferentele intre cele doua documente salvate pe server intr-un mod user friendly. Aceste URL-uri sunt salvate de catre

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. Acest 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)
  })
});
```

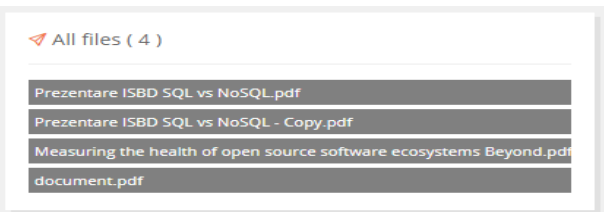


Figura 35 36 backend implementation (7)

6. O alta ruta de backend este '/uploadPDF cu ajutorul careia se insereaza datele unui document intr-un DB. Acesta ruta creata un fisier si il salveaza 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 proiecta 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.

Home

MongoDB Deployments

Create from backup Create new

Development and Utility Single-node deployments intended for environments that do not require high availability.

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

Environments

Create new

None exist at this time. Click "Create new" to create an [mLab Environment](#) (VPC).

Figura 38 Mongoddb 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.

Sandbox databases do not have redundancy and therefore are not suitable for production. Read our documentation on [how to upgrade](#).

Collections Users Stats Backups Tools

Collections

Delete all collections Add collection

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

Figura 39 Mongoddb collections

7. Aplicatie UI/UX

In acest capitol este prezentata aplicatia din punct de vedere al user interface (UI). Pentru aceasta aplicatie am folosit libraria 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 acest web app are o pagina dedicata autentificarii utilizatorului.

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

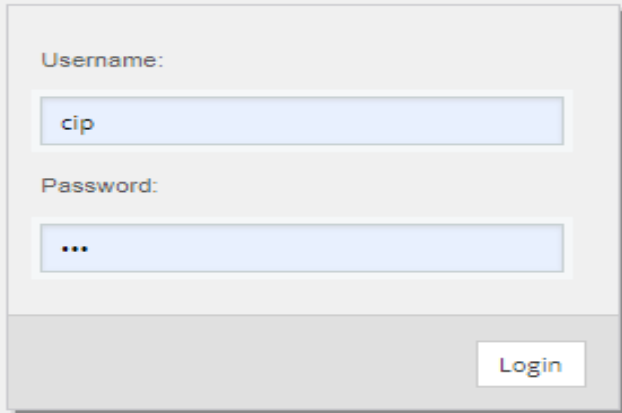
A screenshot of a login form for the MongoDB UI. The form is centered on a light gray background. It consists of a white box with a light gray border. Inside the box, there are two input fields. The first field is labeled 'Username:' and contains the text 'cip'. The second field is labeled 'Password:' and contains three dots '...'. Below the input fields, there is a gray rectangular area containing a white button labeled 'Login'.

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.

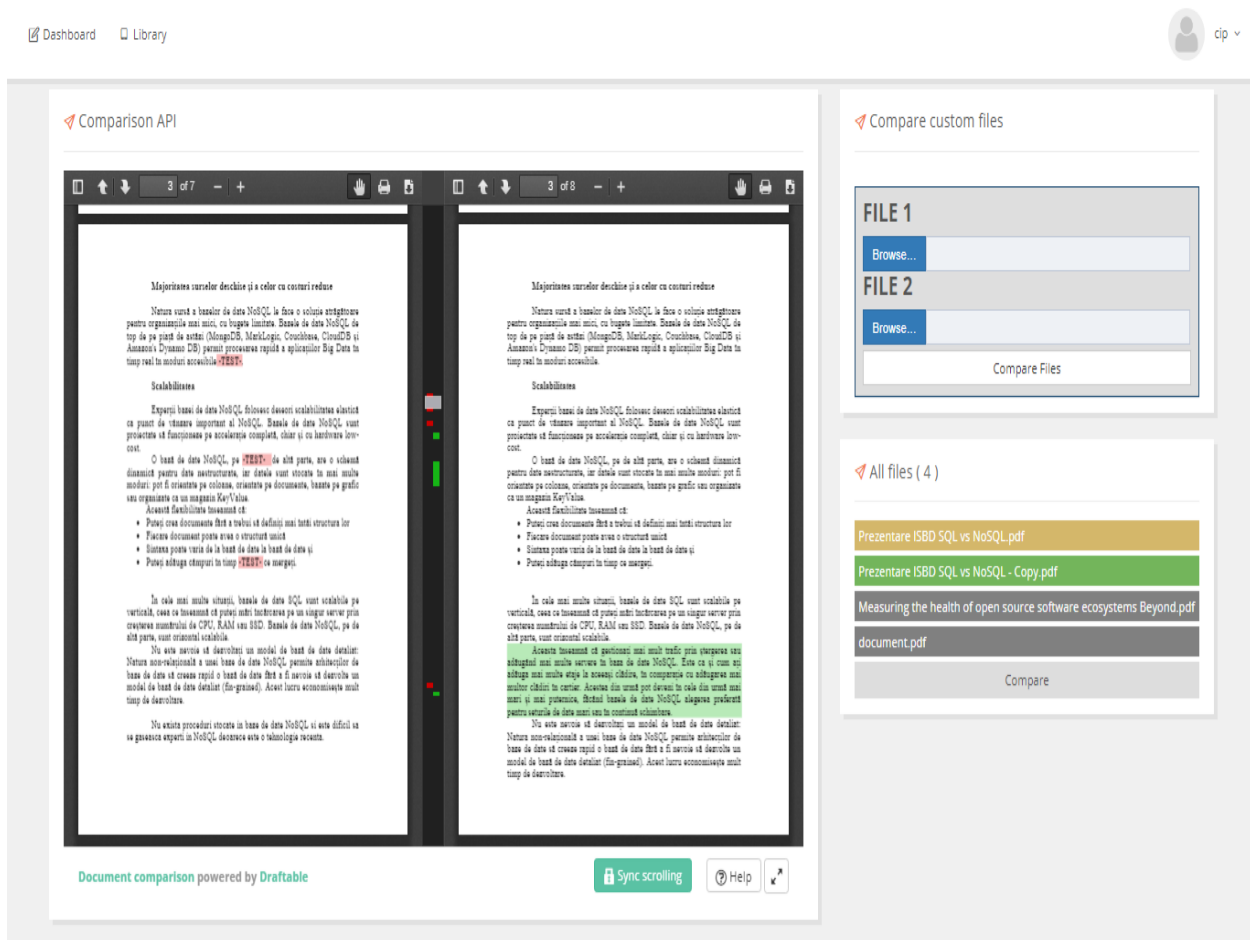


Figura 41 Mongodb UI - Dashboard

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

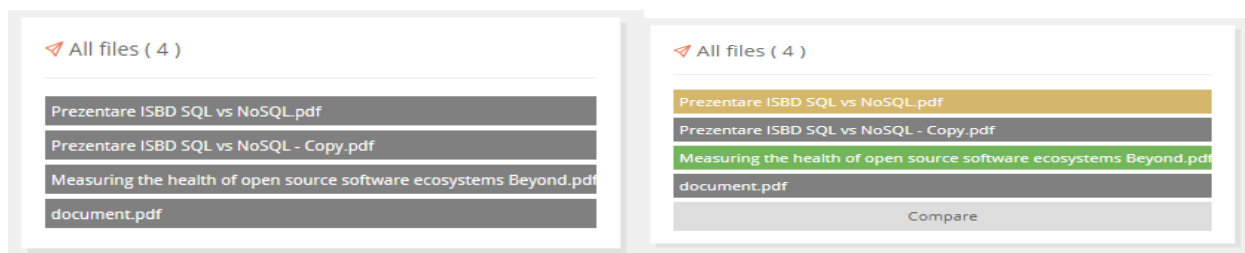


Figura 42 Mongodb UI - Dashboard pdf files

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.

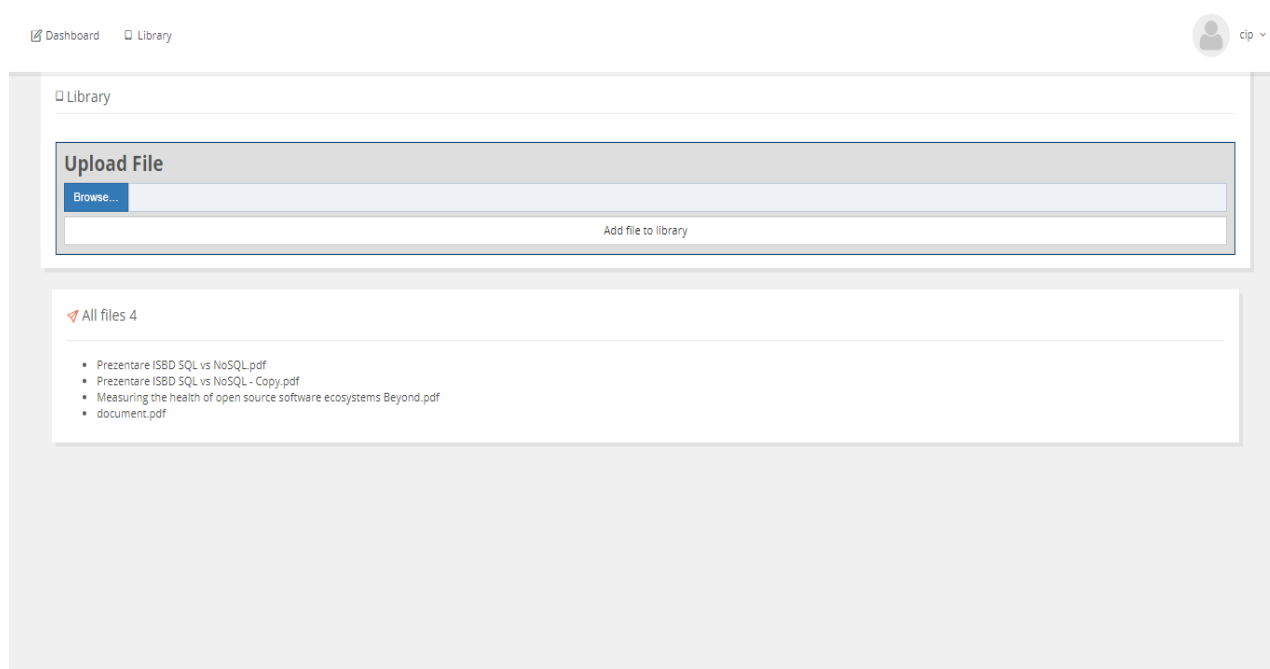


Figura 43 MongoDB UI - Library

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").

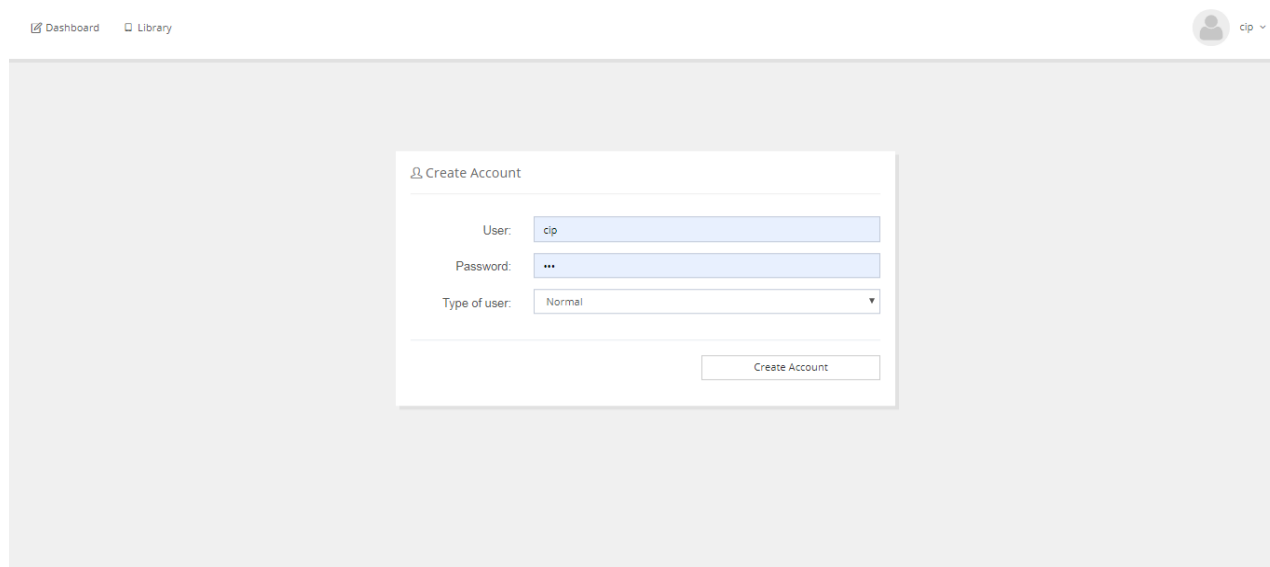


Figura 44 MongoDB UI - Create

8. Teste

În acest capitol sunt redată rezultatele aplicației implementând biblioteca Draftable Compare API menționată anterior. De asemenea este bine de menționat că sunt folosite câteva documente aproape identice, acestea având câteva modificări 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" și am creat o clonă a acestuia adăugând prefixul -copy la finalul documentului. Clona documentului are câteva cuvinte adăugate ("inserted text"), modificat anul de pe prima pagină (2019-2018) și sunt extrase câteva blocuri de text din document.

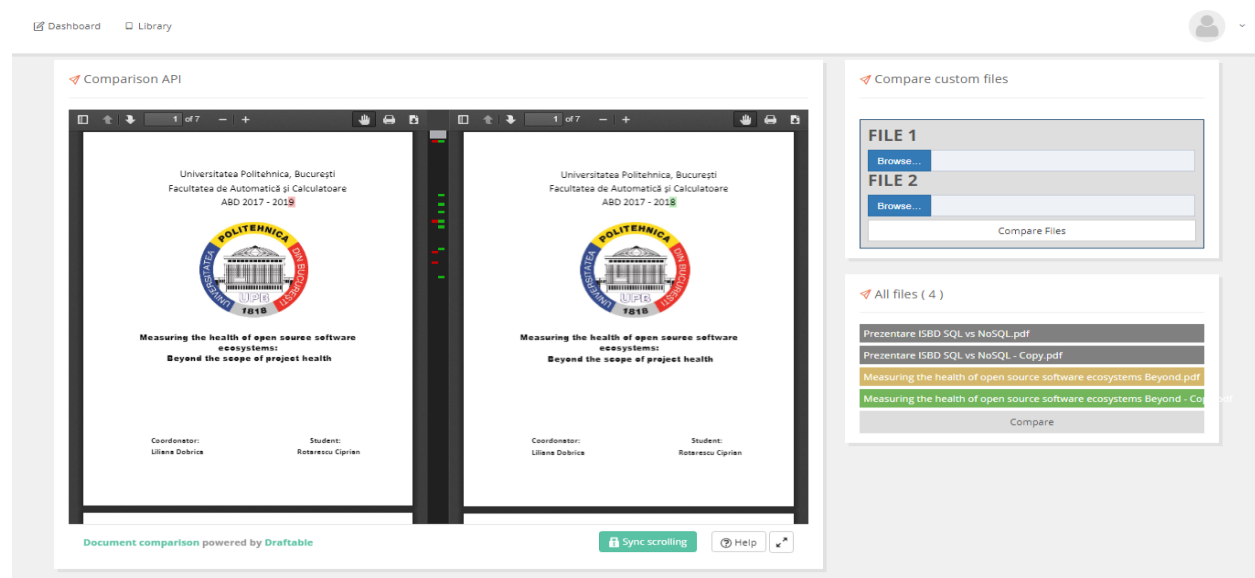


Figura 45 Teste - "Measuring the health of open source software ecosystems Beyond"



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

Se pot evidentia cuvintele adaugate/modificate in partea dreapta ca au un fundal de culoarea verde iar in partea stanga aceleasi cuvinte au un fundal rosu. E de mentionat ca si blocurile de text extrase in partea dreapta au un fundal rosu in documentul din stanga.

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

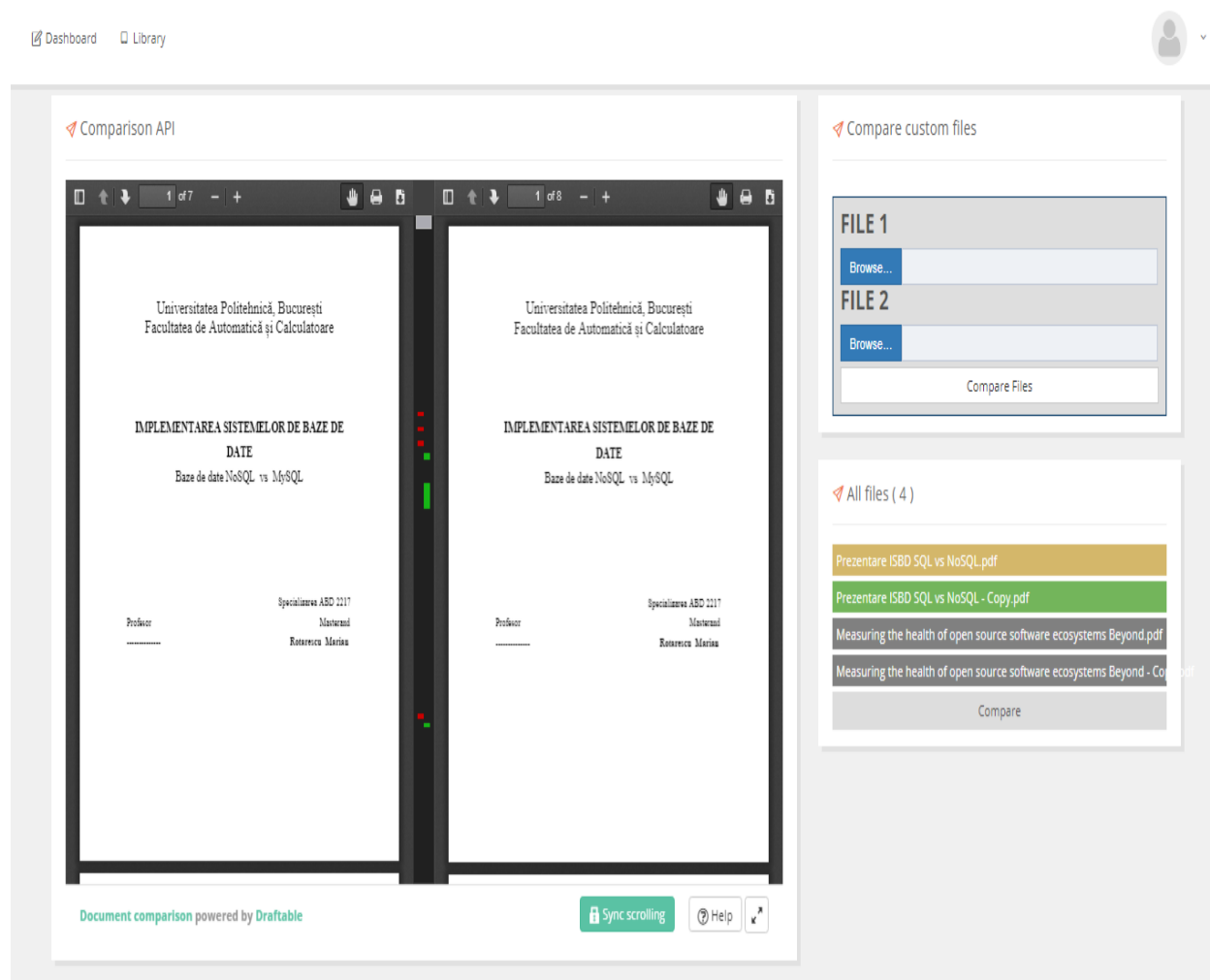


Figura 47 Teste - " Prezentare ISBD SQL vs NoSQL "

Modificarile pe care le-am create pentru acest test sunt urmatoarele:

1. Am ales sa inserez in documentul clona " Prezentare ISBD SQL vs NoSQL " texte random numite "-TEST-". Acestea sunt evidentiata cu un fundal rosu in cadrul imaginii din parte stanga.
2. Am ales sa scot portiuni mari de text din documentul clona pentru a putea fi evidentiata mai usor. Acestea sunt evidentiata cu un fundal verde in cadrul imaginii din parte dreapta.

Cele mai multe modificari sunt evidentiata in cadrul pag 2-6. din documente. De asemenea se pot evidentia ce modificari au fost identificate daca se observa linia despartitoare a celor doua documente (sunt prezente forme verzi /rosii in functie 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.

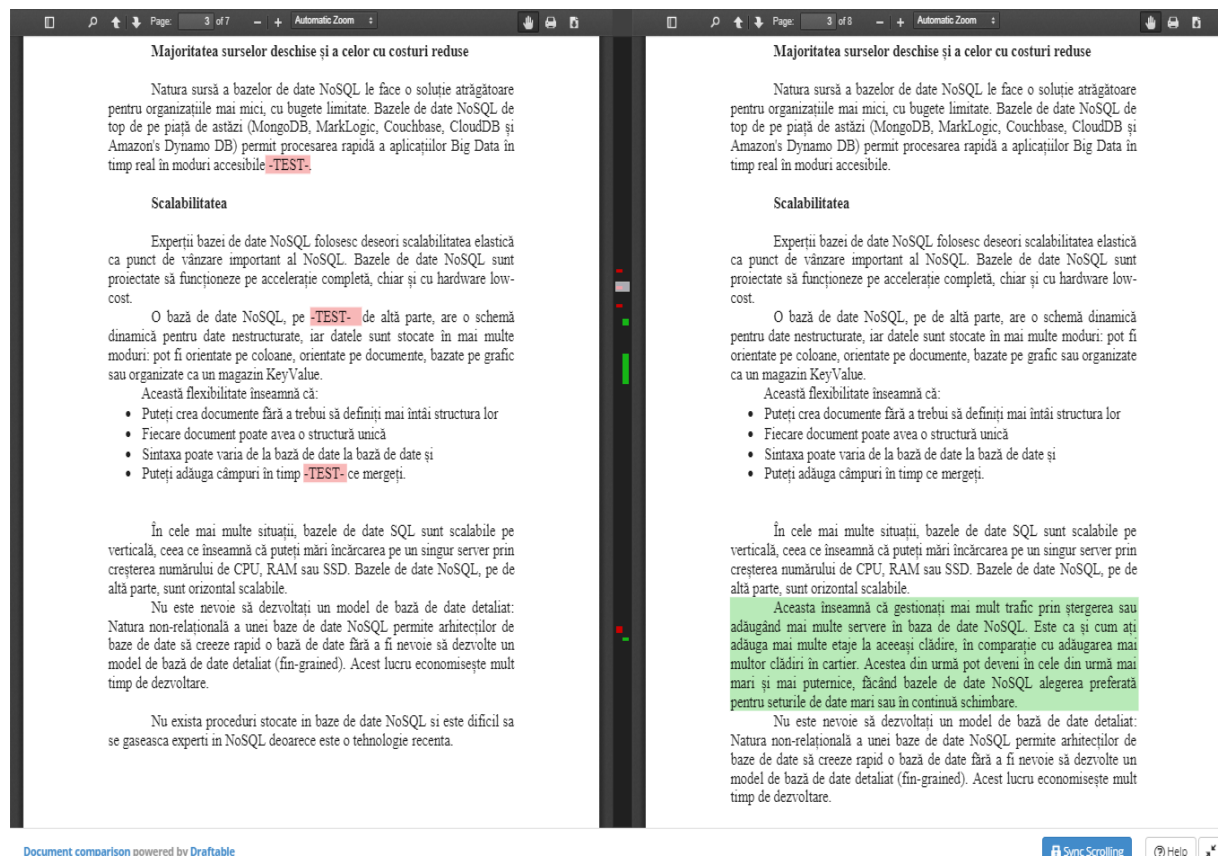


Figura 48 Teste - "Prezentare ISBD SQL vs NoSQL "

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

Este de mentionat faptul ca poate evidenta diferente la diacritice pentru cuvinte (asta in 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. Bineinteles 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-".

Exista si o sectiune in plus in documentul dreapta ce este evidentiata cu fundal verde al textului:

"Aceasta înseamnă că gestionați mai mult trafic prin ștergerea sau adăugând mai multe servere în baza de date NoSQL. Este ca și cum ați adăuga 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 bazele de date NoSQL alegerea preferată pentru seturile de date mari sau în continuă schimbare."

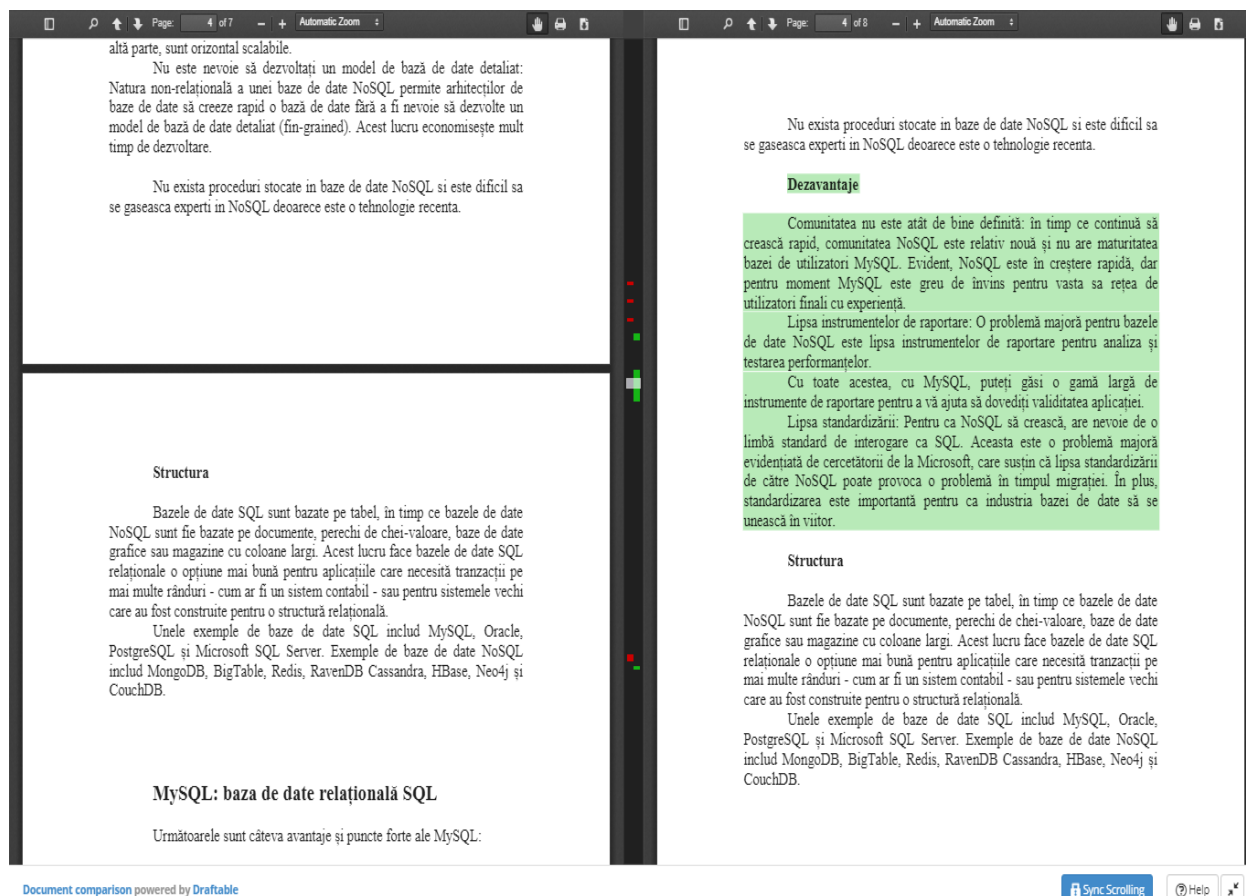


Figura 49 Teste - "Prezentare ISBD SQL vs NoSQL"

Secțiunea "Dezavantaje" a fost decupat dintr-un document și se poate observa fără probleme în imaginea de mai sus iar în următoarele rânduri 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 menționat că acest API poate să identifice textele cu diacritice pentru documentele în limba Română.

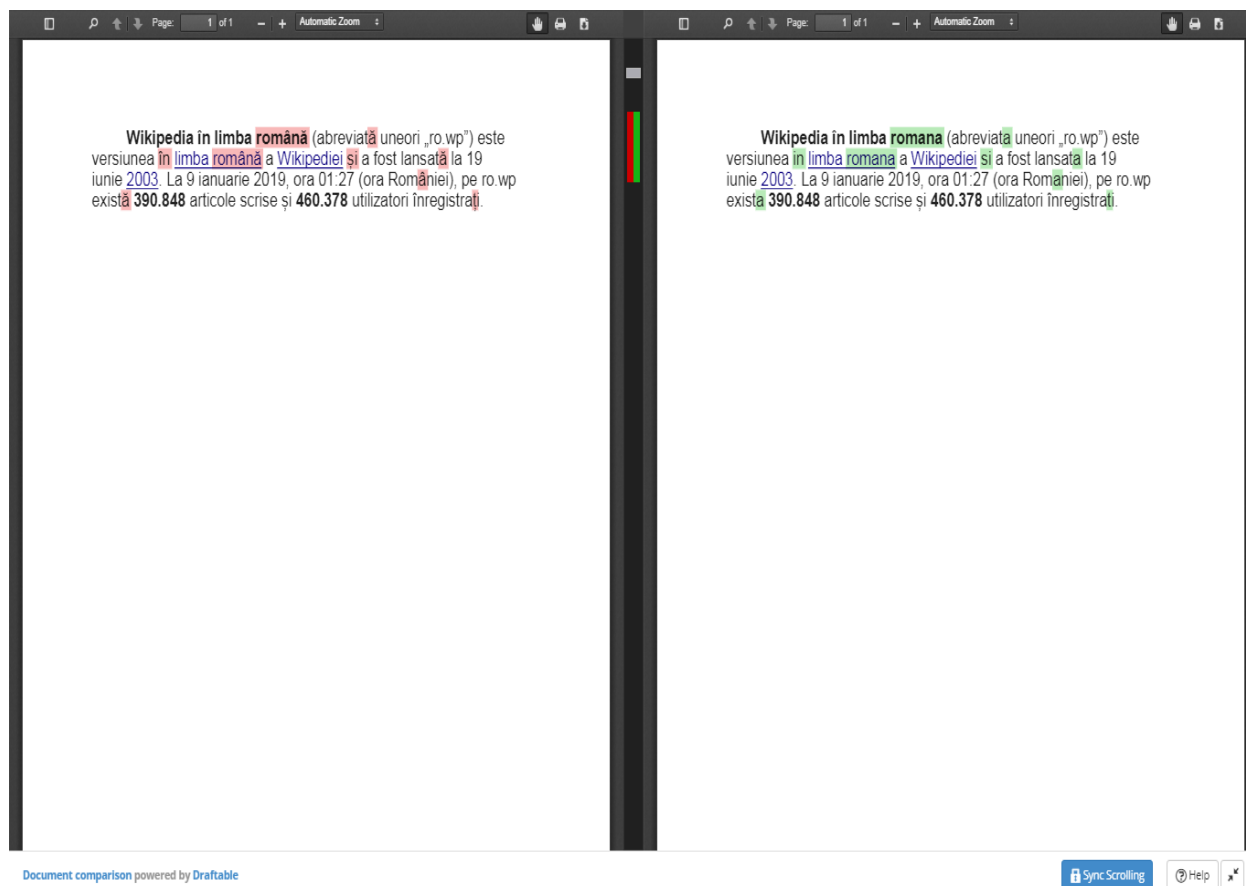


Figura 50 Teste - "Diacritice"

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

9. Concluzii

Pentru detectarea plagiatului si in urma testelor mentionate s-a dovedit faptul ca acest API - DRAFTABLE poate fi folosit pentru indeplinirea scopului propus . Exista insa si un drawback legat de acest API si anume ca trebuie platit pentru a avea acces la un numar ridicat de comparari si 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://en.wikipedia.org/wiki/JavaScript>

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

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