



Universitatea Tehnică „Gheorghe Asachi” din IAȘI

Facultatea de Automatică și Calculatoare

Domeniul: *Calculatoare și Tehnologia Informației*

Specializarea: *Calculatoare/Tehnologia Informației*



Platformă web pentru analiza și vizualizarea emoțiilor din site-uri de socializare

Lucrare de Licență

Absolvent

Alexandru Ivanov

Coordonator științific

Ș.l. dr. Iulian PETRILA

Iași, 2024

DECLARAȚIE DE ASUMARE A AUTENTICITĂȚII LUCRĂRII DE LICENȚĂ

Subsemnatul(a) Prenume NUME,

legitimat(ă) cu BI/CI..

autorul lucrării **Titlu**

elaborată în vederea susținerii examenului de finalizare a studiilor de licență organizat de către Facultatea de Automatică și Calculatoare din cadrul Universității Tehnice „Gheorghe Asachi” din Iași, sesiunea iulie a anului universitar 2016-2017, luând în considerare conținutul Art. 34 din Codul de etică universitară al Universității Tehnice „Gheorghe Asachi” din Iași (Manualul Procedurilor, UTI.POM.02 – Funcționarea Comisiei de etică universitară), declar pe proprie răspundere, că această lucrare este rezultatul propriei activități intelectuale, nu conține porțiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislației române (legea 8/1996) și a convențiilor internaționale privind drepturile de autor.

Data

Zi – Lună – An

Semnătura

Platformă web pentru analiza și vizualizarea emoțiilor din site-uri de socializare

Alexandru Ivanov

Rezumat

Lucrarea de față își propune dezvoltarea unei aplicații web care să ofere o vizualizare amplă a sentimentelor exprimate de către utilizatori în cadrul platformelor de socializare. Scopul soluției propuse este de a satisface nevoia utilizatorilor de a înțelege emoțiile colegilor din mediul online cu privire la anumite subiecte cotidiene sau de o importanță semnificativă pentru consumatorul aplicației.

Soluția proiectată în cadrul acestei lucrări își dorește să extindă abordarea aleasă de majoritatea lucrărilor anterioare care clasifică sentimentele în mod binar, în sentimente pozitive și negative. Abordările existente definesc intensitățile acestor sentimente pe care dorim să le clasificăm în sentimente concrete și să oferim o vizualizare conexă pentru a observa diferite comportamente și tendințe emoționale în timp real.

În cadrul soluției dezvoltate, propunem utilizarea modului *NLTK* ce dispune de *VADER* lexicon, un dicționar de cuvinte construit pentru extragerea sentimentelor din postări de pe platforma Twitter, platformă cu multe asemănări în relație cu site-ul principal de pe care facem extragerea, *Reddit*. Implementarea folosește metode de inteligență artificială pentru clasificarea și calculul intensității sentimentelor din componentele postărilor. De asemenea, este împărțită în componente cu responsabilități singulare pentru gestionarea eficientă a tuturor operațiilor ce pot fi realizate. Astfel, aplicația este împărțită în 3 componente, prima fiind menționată anterior, componenta de analiză și extragere.

Următoarea componentă este o aplicație *Spring* ce gestionează persistența și securitatea sistemului, având rolul de intermediar între componenta de analiză și cea de vizualizare. Ultima componentă este interfața cu care interacționează utilizatorul. Aceasta este construită utilizând framework-ul *React.js* împreună cu alte biblioteci precum *MaterialUI* și *Recharts.js* pentru componentele oferite care facilitează stilizarea unui modul de tip front-end.

Cu toate acestea, există aspecte care pot fi îmbunătățite în viitor. De exemplu, integrarea unor algoritmi de învățare automată mai avansați ar putea îmbunătăți precizia clasificării sentimentelor. De asemenea, extinderea suportului pentru alte platforme de socializare ar putea oferi o imagine și mai completă asupra sentimentelor utilizatorilor. Astfel, deși aplicația propusă aduce inovații semnificative în domeniul analizei sentimentelor pe platformele de socializare, există încă oportunități de dezvoltare și perfecționare care pot fi explorate în continuare.

Prin urmare, lucrarea de față prezintă o soluție pentru vizualizarea sentimentelor exprimate de utilizatori pe platformele de socializare, având ca scop oferirea unei înțelegeri mai profunde a emoțiilor acestora.

Cuprins

Introducere	1
Capitolul 1.Fundamentarea teoretică și tehnologiile utilizate.....	3
1.1. Analiza sentimentelor	4
1.1.1. VADER Lexicon.....	5
1.1.2 Metode bazate pe rețele neuronale.....	6
1.2. Vizualizarea datelor	7
1.3. Tehnologii utilizate	8
1.3.1 Spring Framework	8
1.3.2. React.js.....	8
Capitolul 2.Proiectarea aplicației	10
2.1. Componenta de extragere a conținutului și analiză a sentimentelor	11
2.1.1 Extragerea postărilor	11
2.1.2 Analiza emoțiilor	11
2.2. Componenta de securitate și persistență a datelor	12
2.3. Componenta de vizualizare.....	14
Capitolul 3.Implementarea aplicației	16
3.1. Componenta de extragere a conținutului și analiză a sentimentelor	16
3.2. Componenta de securitate și persistență	21
3.3. Componenta de vizualizare.....	25
Concluzii	37
Bibliografie	38
Anexe	39
Anexa1 – Detalii elemente teoretice	Error! Bookmark not defined.
Anexa2 – Codul sursă	39

Introducere

În era digitală contemporană, rețelele de socializare au devenit parte integrantă a vieților noastre, oferind o platformă pentru comunicare, partajare de informații și exprimarea emoțiilor. Aceste platforme generează un volum vast de date textuale care pot fi analizate pentru a înțelege mai bine emoțiile utilizatorilor. Analiza sentimentelor din aceste date a evoluat semnificativ de-a lungul timpului, trecând prin diferite etape și abordări, fiecare cu propriile sale avantaje și limitări.

La începutul dezvoltării analizelor sentimentelor, abordările lexicon-based au fost cele mai utilizate. Aceste metode implică utilizarea unor dicționare de sentimente, care asociază cuvinte individuale cu scoruri de polaritate (pozitiv, negativ, neutru). Primele implementări notabile includ AFINN și SentiWordNet, care au furnizat baze de date extinse de cuvinte asociate cu scoruri de sentiment [1]. Avantajul principal al acestor metode este simplitatea și interpretabilitatea lor, dar ele au limitări semnificative în gestionarea complexității și a nuanțelor emoționale din textele mai lungi și mai elaborate. Un avans semnificativ în această direcție a fost realizat prin dezvoltarea VADER (Valence Aware Dictionary and sEntiment Reasoner). VADER, integrat în biblioteca NLTK (Natural Language Toolkit), este conceput special pentru analiza sentimentelor din datele generate pe rețelele sociale. Acesta utilizează un dicționar de sentimente, completat de reguli heuristice care capturează intensitatea și nuanțele contextuale ale sentimentelor. Studiile au demonstrat că VADER poate atinge performanțe comparabile cu modelele bazate pe învățare automată în anumite contexte, având avantajul de a fi rapid și ușor de implementat [2]. Cu evoluția tehnologiei, metodele de învățare automată au devenit din ce în ce mai populare în analiza sentimentelor. Algoritmi precum SVM (Support Vector Machines) și Naive Bayes au fost printre primele tehnici utilizate pentru a clasifica sentimentele pe baza trăsăturilor extrase din text [3]. Aceste metode au demonstrat o acuratețe superioară comparativ cu abordările lexicon-based, deoarece pot învăța din datele etichetate și pot captura relațiile complexe dintre cuvinte.

Rețelele neuronale au adus o revoluție în domeniu datorită capacității lor de a învăța reprezentări de caracteristici complexe și nuanțate. Rețelele neuronale recurente (RNN) și variantele lor, cum ar fi LSTM (Long Short-Term Memory) și GRU (Gated Recurrent Units), au permis capturarea dependențelor pe termen lung în texte, oferind o înțelegere mai profundă a contextului. Un salt major în analiza sentimentelor a fost realizat prin dezvoltarea modelelor pre-antrenate, cum ar fi BERT (Bidirectional Encoder Representations from Transformers) și GPT-3. Aceste modele utilizează arhitecturi Transformer și sunt pre-antrenate pe corpuri masive de date textuale, permițându-le să înțeleagă contextul și nuanțele subtile ale limbajului [4]. BERT, de exemplu, este antrenat bidirecțional, capturând contextul din ambele părți ale unui cuvânt într-o propoziție, ceea ce îi permite să înțeleagă mai bine sentimentele complexe și contextuale. GPT-3, cu capacitatea sa de a genera text coerent și contextual relevant, poate fi utilizat pentru a analiza și genera texte cu diverse tonuri emoționale. Aceste modele oferă o acuratețe superioară și pot fi adaptate pentru diverse domenii și aplicații.

Analiza sentimentelor pe rețelele sociale a devenit un subiect de interes major datorită cantității mari de date generate de utilizatori și a relevanței acestor date pentru diverse domenii. Una dintre cele mai comune aplicații ale analizei sentimentelor pe rețelele sociale este

monitorizarea opiniei publice. Studiile au demonstrat că analiza sentimentelor poate oferi perspective valoroase asupra opiniei publice, tendințelor sociale, prezice rezultatele electorale și chiar predictibilității pieței financiare [5]. Companiile utilizează analiza sentimentelor pentru a monitoriza feedback-ul clienților pe platformele de socializare și pentru a îmbunătăți serviciile și produsele oferite. Evaluarea recenziilor și comentariilor postate de utilizatori poate oferi informații valoroase despre punctele forte și slabe ale unui produs sau serviciu, permițând companiilor să reacționeze rapid la problemele identificate. De asemenea, analiza sentimentelor poate fi utilizată pentru a identifica și analiza tendințele sociale emergente. De exemplu, cercetătorii pot analiza conversațiile online pentru a detecta schimbările în atitudinile față de anumite subiecte, cum ar fi sănătatea publică, schimbările climatice sau mișcările sociale. Aceste informații pot fi folosite pentru a dezvolta politici și strategii mai bine informate.

Un exemplu notabil este studiul realizat de Thelwall, care a explorat sentimentele exprimate pe platforma MySpace folosind o combinație de metode lexicon-based și statistici textuale [6]. De asemenea, Pang și Lee au fost pionieri în utilizarea recenziilor de filme pentru a dezvolta și testa modele de analiză a sentimentelor, demonstrând eficiența tehnicilor de învățare automată în acest domeniu [7]. Acestea au studiat doar intensitățile emoțiilor într-un sens liniar, de la negativ la pozitiv și stare neutră. Pornind de la acest fapt, lucrarea își propune să facă o distincție mai largă a emoțiilor exprimate în cadrul ideilor/discuțiilor captate din rețele de socializare consacrate. Abordările tradiționale în analiza sentimentelor au studiat intensitățile emoțiilor într-un sens liniar, variind de la negativ la pozitiv și incluzând o stare neutră. Deși aceste metode au furnizat o bază solidă pentru înțelegerea emoțiilor în mediul online, ele sunt limitate în captarea complexității și diversității emoțiilor umane. În acest context, lucrarea de față își propune să extindă această abordare prin realizarea unei distincții mai largi și mai nuanțate a emoțiilor exprimate în discuțiile și ideile captate din rețelele de socializare consacrate. Prin utilizarea tehnicilor de procesare a limbajului natural (NLP), această cercetare vizează identificarea și clasificarea unui spectru mai larg de emoții, oferind de asemenea și o vizualizare amplă pentru identificarea de pattern-uri și extragerea de informații importante despre domeniile aferente. Astfel, vizualizarea rezultatelor prin grafice și diagrame interactive va oferi utilizatorilor o imagine clară și intuitivă a datelor analizate, evidențiind modele recurente și anomalii.

Capitolul 1. Fundamentarea teoretică și tehnologiile utilizate

Analiza emoțiilor pe site-urile de socializare reprezintă un domeniu de cercetare emergent și deosebit de relevant în contextul digital actual. Cu milioane de utilizatori activi zilnic pe platforme precum Facebook, Twitter, Instagram și altele, volumul de date generate este enorm. Aceste date includ opinii, gânduri, reacții și emoții ale utilizatorilor, oferind o sursă bogată de informații pentru diverse scopuri. Pentru a asigura o înțelegere clară a subiectului, este important să definim conceptele cheie utilizate în analiza emoțiilor pe site-urile de socializare.

Emoțiile reprezintă reacții psihologice și fiziologice complexe la stimuli interni sau externi. Ele sunt esențiale pentru supraviețuire și adaptare, influențând comportamentul și deciziile noastre zilnice. Emoțiile pot fi clasificate în diferite moduri. Conform teoriei lui Paul Ekman [8], există șase emoții de bază universale recunoscute prin expresii faciale specifice: fericire, tristețe, frică, furie, surpriză și dezgust. Aceste emoții sunt considerate universale deoarece sunt recunoscute în toate culturile umane. Modelul circumplex al emoțiilor, propus de James Russell [9], sugerează că emoțiile pot fi reprezentate într-un spațiu bidimensional, definit de două axe: valența (pozitiv-negativ) și excitația (ridicat-scăzut). Acest model ajută la înțelegerea modului în care emoțiile se pot combina și varia.

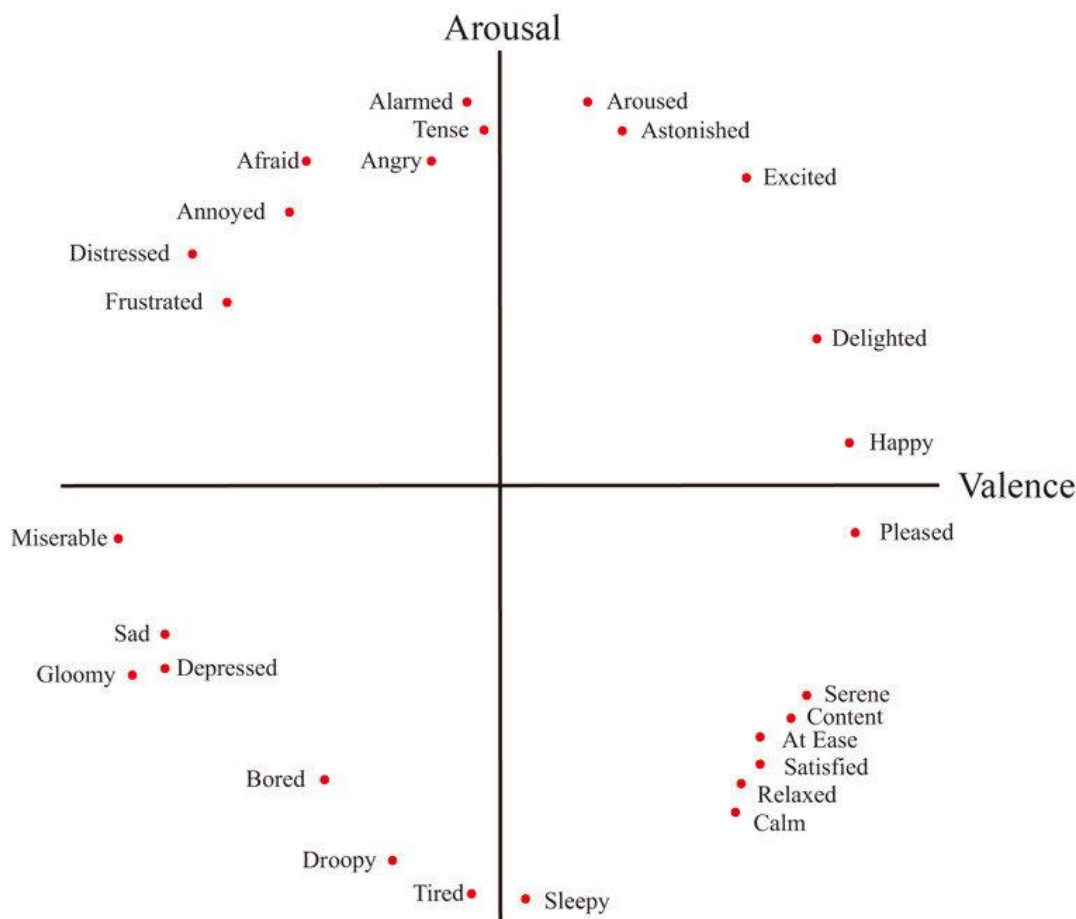


Figura 1.1. Modelul circumplex al lui Russel.

Impactul social media asupra societății este vast, influențând modul în care consumăm știri, formăm opinii, interacționăm cu branduri și participăm la discuții sociale și politice. Social media reprezintă o categorie vastă de platforme online care permit utilizatorilor să creeze, să partajeze și să interacționeze cu conținut generat de alți utilizatori. Aceste platforme au revoluționat modul în care comunicăm, interacționăm și consumăm informații. Există mai multe tipuri de platforme de social media, fiecare cu propriile caracteristici și scopuri. Platforme precum Facebook și LinkedIn care facilitează conectarea și interacțiunea între utilizatori prin crearea de profiluri personale, grupuri de interese și pagini de afaceri. Twitter, unde utilizatorii pot posta mesaje scurte (tweet-uri) și pot urmări actualizările altor utilizatori. Aceasta este cunoscută pentru rapiditatea și concizia comunicării. Un alt tip de platforme sunt cel de partajare a conținutului media. Instagram, YouTube și TikTok permit utilizatorilor să creeze, să distribuie și să vizualizeze conținut media, cum ar fi fotografii, videoclipuri și transmisiuni live. În plus, forumurile și comunitățile online contribuie semnificativ în discuțiile pe subiecte de interes și facilitează găsirea răspunsurilor pentru diferite probleme. Platforme precum Reddit și Quora sunt astfel structurate în sub-forumuri (subreddits) pentru discuții axate pe domenii singulare de discuție.

1.1. Analiza sentimentelor

Analiza de sentiment este un domeniu care combină tehnici din procesarea limbajului natural (NLP), învățarea automată (ML) și mining-ul de text pentru a identifica și extrage informații subiective din texte. Scopul principal al analizei de sentiment este de a determina atitudinea autorului față de un anumit subiect, aceasta putând fi pozitivă, negativă sau neutră.

Metodele lexicon-based pentru analiza de sentiment se bazează pe utilizarea dicționarului de cuvinte asociate cu sentimente specifice pentru a evalua textele. Aceste metode sunt directe și utile pentru a determina polaritatea unui text (pozitiv, negativ, neutru) fără a necesita date etichetate pentru antrenare. Principiul de bază al acestor metode este de a compara fiecare cuvânt din text cu un lexicon de sentimente și de a calcula un scor total de sentiment pe baza prezenței și frecvenței cuvintelor din lexicon. Construirea lexiconului de sentimente implică crearea unei liste de cuvinte și expresii asociate cu sentimente pozitive, negative sau neutre. Exemple de lexiconuri includ SentiWordNet, AFINN și VADER. SentiWordNet atribuie scoruri de polaritate fiecărui cuvânt din WordNet, AFINN oferă scoruri numerice care indică sentimentul fiecărui cuvânt, de la -5 (foarte negativ) la +5 (foarte pozitiv), iar VADER este conceput special pentru analiza de sentiment pe texte din social media, atribuind scoruri de intensitate sentimentelor. Funcționarea metodelor lexicon-based implică mai multe etape. Textul este mai întâi tokenizat, adică împărțit în unități mai mici (cuvinte sau fraze). Fiecare cuvânt din text este apoi comparat cu lexiconul de sentimente. Cuvintele găsite în lexicon sunt evaluate și li se atribuie un scor de sentiment. Scorurile tuturor cuvintelor din text sunt cumulate pentru a obține un scor total de sentiment. Cuvintele pozitive și negative sunt identificate, iar scorul total reflectă sentimentele exprimate în text. Cuvintele precum "nu" sau "niciodată" (negări) și "foarte" sau "extrem de" (amplificatori) sunt identificate pentru a ajusta scorurile de sentiment ale cuvintelor învecinate. Negările pot inversa sentimentul unui cuvânt, în timp ce amplificatorii cresc intensitatea sentimentului. Metodele lexicon-based au mai multe avantaje, printre care se numără simplitatea și ușurința în implementare, eficiența pentru texte scurte și bine structurate, și faptul că nu necesită date etichetate pentru antrenare. Totuși, aceste metode au și dezavantaje semnificative, inclusiv

limitările în captarea nuanțelor complexe ale limbajului natural, sensibilitatea redusă la context și sarcini complexe, și dependența de calitatea și acoperirea lexiconului utilizat.

Pe de altă parte, metodele de inteligență artificială pentru analiza de sentiment utilizează algoritmi de învățare automată și învățare profundă pentru a analiza textele și a determina sentimentele exprimate. Aceste metode implică antrenarea modelelor pe seturi mari de date etichetate pentru a învăța tipare și a prezice sentimentele în texte noi. Colectarea datelor și etichetarea implică adunarea de texte din diverse surse și etichetarea lor manuală pentru a indica sentimentele. Preprocesarea datelor este esențială pentru a elimina zgomotul și a standardiza formatul, incluzând tokenizarea, eliminarea stop word-urilor, stemming și lematizare. Caracteristicile relevante sunt extrase din texte, cum ar fi frecvența cuvintelor, n-gramme și reprezentările vectoriale ale cuvintelor (embeddings). Modelele de învățare automată, cum ar fi Naive Bayes și SVM, sau rețelele neuronale, cum ar fi LSTM și BERT [4], sunt antrenate pe seturile de date etichetate. Metodele de inteligență artificială oferă avantaje semnificative, inclusiv capacitatea de a captura nuanțele complexe ale limbajului natural și de a se adapta la diverse contexte. Aceste metode pot oferi o acuratețe mai mare și sunt capabile să învețe din seturi mari de date. Totuși, metodele de IA necesită date etichetate pentru antrenare, ceea ce poate fi costisitor și consumator de timp.

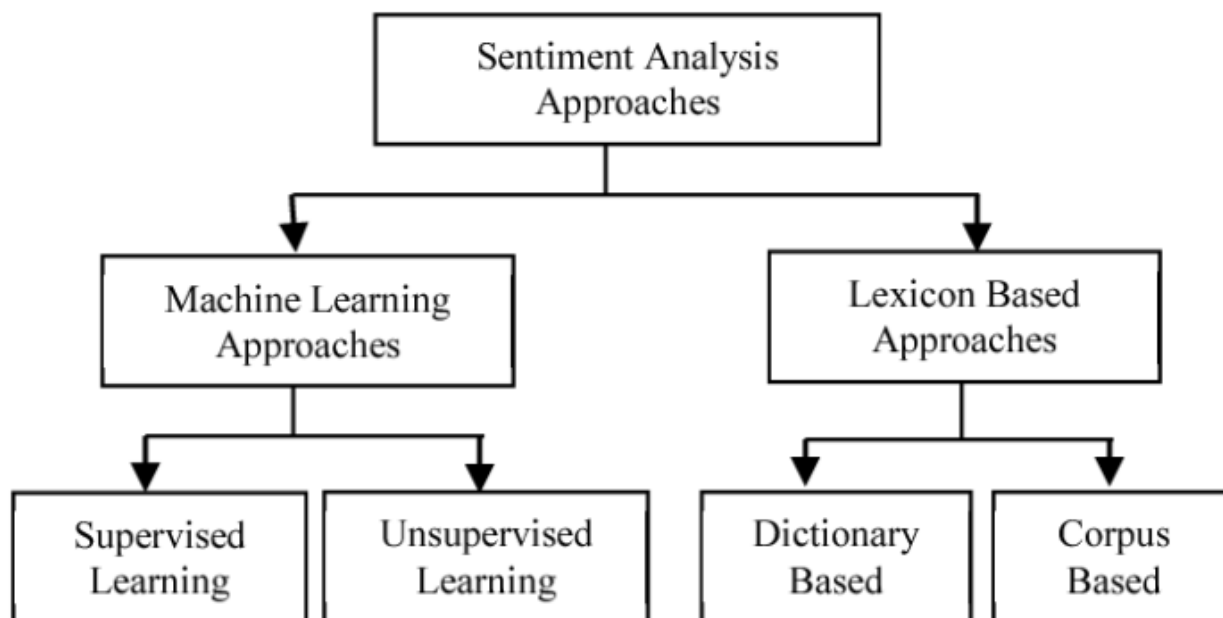


Figura 1.2. Metode de analiză a sentimentelor din text

1.1.1. VADER Lexicon

Printre diversele metode utilizate pentru această analiză, lexiconul VADER (Valence Aware Dictionary and sEntiment Reasoner) se evidențiază prin capacitatea sa de a analiza în mod precis sentimentele exprimate în textele de pe rețelele de socializare și alte forme de comunicare informală. VADER este un lexicon și un model de analiză de sentiment special conceput pentru a trata limbajul folosit în rețelele de socializare. Acesta a fost dezvoltat de C.J. Hutto și Eric Gilbert [2]. VADER se bazează pe o combinație de reguli și o listă de cuvinte cu

scoruri de polaritate predefinite, care au fost validate empiric prin multiple studii. VADER a fost creat pentru a analiza eficient textele scurte, cum ar fi tweet-urile, comentariile de pe Facebook și recenziile online. Acesta ia în considerare atât polaritatea (pozitivă, negativă, neutră) cât și intensitatea sentimentelor exprimate. Lexiconul VADER conține aproximativ 7,500 de cuvinte, fiecare cu un scor de sentiment asociat care variază de la -4 (foarte negativ) la +4 (foarte pozitiv). Aceste scoruri au fost determinate printr-un proces de crowdsourcing, unde oameni reali au evaluat sentimentele asociate fiecărui cuvânt.

Funcționarea VADER se bazează pe câteva principii esențiale. Fiecare cuvânt din lexiconul VADER are un scor de sentiment predefinit. Atunci când VADER analizează un text, acesta compară fiecare cuvânt din text cu lexiconul pentru a atribui un scor de sentiment. Acesta utilizează reguli heuristice pentru a ajusta scorurile de sentiment pe baza contextului. De exemplu, amplificatorii precum "foarte" sau "extrem de" cresc intensitatea sentimentului asociat unui cuvânt. Negările precum "nu" sau "niciodată" inversează polaritatea sentimentului. VADER ia în considerare utilizarea punctuației și a emoticoanelor. De exemplu, un text cu multe semne de exclamare poate avea un sentiment mai intens, iar emoticoanele sunt tratate ca indicatori direcți ai sentimentelor. Cuvintele scrise cu majuscule sunt considerate mai intense. De exemplu, "BINE" are un scor de sentiment mai mare decât "bine".

VADER prezintă mai multe avantaje care îl fac ideal pentru analiza de sentiment a textelor de pe rețelele de socializare și alte forme de comunicare informală. VADER a fost validat empiric și s-a dovedit a fi la fel de precis, dacă nu mai precis, decât metodele de învățare automată supravegheate pentru analiza de sentiment. Fiind un model bazat pe reguli, este simplu de implementat și rulează rapid, făcându-l ideal pentru analize în timp real. Acesta fost special conceput pentru a înțelege limbajul informal și colocvial utilizat pe rețelele de socializare, inclusiv utilizarea jargonului, a acronimelor și a emoticoanelor. Spre deosebire de modelele de învățare automată, VADER nu necesită un set de date etichetat pentru antrenare, fiind gata de utilizare imediat după instalare. Cu toate că are multe avantaje, există și câteva limitări de care utilizatorii trebuie să fie conștienți. Deoarece acest model se bazează pe un set predefinit de reguli și scoruri, poate să nu fie la fel de adaptabil ca modelele de învățare automată care pot învăța și evolua pe baza unor noi seturi de date. Deși gestionează bine limbajul informal, poate întâmpina dificultăți în tratarea unor construcții lingvistice foarte complexe sau ambigue. VADER analizează fiecare text în mod izolat și nu ia în considerare contextul mai larg în care textul a fost scris, ceea ce poate duce la interpretări greșite în anumite cazuri.

1.1.2 Metode bazate pe rețele neuronale

Metodele bazate pe rețele neuronale pentru analiza de sentiment utilizează algoritmi avansați de învățare profundă (deep learning) pentru a înțelege și interpreta emoțiile exprimate în textele scrise. Rețelele neuronale, în special rețelele neuronale convoluționale (CNN) și rețelele neuronale recurente (RNN), cum ar fi LSTM (Long Short-Term Memory) și GRU (Gated Recurrent Unit), sunt frecvent utilizate în acest context datorită capacității lor de a modela relațiile complexe dintre cuvinte și fraze.

Rețelele neuronale convoluționale (CNN) sunt utilizate în principal pentru analizarea datelor spațiale, fiind extrem de eficiente în detectarea caracteristicilor locale în textele scurte. CNN-urile aplică filtre (kernels) asupra datelor de intrare pentru a extrage trăsături relevante, cum ar fi n-gramme și combinații de cuvinte. Aceste filtre sunt antrenate să detecteze tipare

specifice în text. De exemplu, un filtru poate fi sensibil la secvențe de cuvinte pozitive, în timp ce altul poate detecta secvențe negative. Rezultatele acestor filtre sunt apoi combinate și analizate pentru a produce o predicție finală a sentimentului. Straturile de pooling sunt utilizate pentru a reduce dimensionalitatea și a consolida informațiile esențiale, permițând rețelei să devină mai robustă și mai eficientă.

Rețelele neuronale recurente (RNN), în special variantele LSTM și GRU, sunt concepute pentru a lucra cu date secvențiale, fiind capabile să captureze dependențele pe termen lung din texte. Aceste rețele au structuri speciale care le permit să "își amintească" informații anterioare și să le utilizeze pentru a face predicții mai precise. LSTM-urile utilizează celule de memorie care pot păstra și uita informații în mod selectiv, datorită mecanismelor de poartă (gates) – poarta de intrare, poarta de uitare și poarta de ieșire. Aceste mecanisme reglează fluxul de informații prin celulă, permițând rețelei să gestioneze eficient contextul pe termen lung și să facă predicții precise chiar și pentru secvențe lungi de text. GRU-urile sunt similare LSTM-urilor, dar au o structură mai simplificată, combinând poarta de intrare și cea de uitare într-o singură poartă, ceea ce le face mai eficiente din punct de vedere computațional.

În practică, rețelele neuronale pentru analiza de sentiment sunt antrenate pe seturi mari de date etichetate. În timpul antrenării, modelul ajustează greutatea conexiunilor sale interne pentru a minimiza eroarea dintre predicțiile sale și etichetele reale. Acest proces implică utilizarea algoritmului de optimizare, cum ar fi stochastic gradient descent (SGD) sau Adam, pentru a actualiza greutatea pe baza gradientelor calculate în timpul backpropagation. Reprezentările vectoriale ale cuvintelor (word embeddings), cum ar fi Word2Vec, GloVe sau embeddings din modele mai avansate precum BERT, sunt adesea utilizate ca intrare în rețea, capturând semantica și relațiile dintre cuvinte în spații vectoriale multidimensionale. Rețelele neuronale convoluționale și cele recurente sunt adesea combinate pentru a beneficia de avantajele ambelor arhitecturi. De exemplu, un model hibrid poate utiliza CNN-uri pentru a extrage trăsături locale din text și RNN-uri pentru a capta dependențele pe termen lung, oferind astfel o analiză mai cuprinzătoare și precisă a sentimentului.

1.2. Vizualizarea datelor

Vizualizarea sentimentelor reprezintă un aspect esențial în analiza datelor, în special în contextul analizei de sentiment, unde scopul este de a înțelege și de a comunica emoțiile exprimate în textele scrise, cum ar fi postările pe rețelele de socializare. Aceasta implică transformarea datelor brute de sentiment într-o formă vizuală care să permită identificarea tiparelor, tendințelor și relațiilor. Vizualizările eficiente pot ajuta la interpretarea rapidă și clară a rezultatelor analizei de sentiment, facilitând luarea deciziilor bazate pe date.

Există numeroase tehnici de vizualizare a datelor, fiecare fiind adecvată pentru anumite tipuri de date și scopuri analitice:

- **Graficele de bare** sunt utile pentru compararea valorilor între diferite categorii, oferind o reprezentare vizuală clară a diferențelor și similarităților.
- **Graficele de linie** sunt folosite pentru a arăta evoluția datelor în timp, evidențiind tendințele și fluctuațiile.
- **Diagramele de tip plăcintă (pie charts)** sunt eficiente pentru a reprezenta proporțiile diferitelor componente dintr-un întreg, facilitând înțelegerea distribuției procentuale.

- **Graficele scatter** sunt utilizate pentru a vizualiza relațiile dintre două variabile, ajutând la identificarea corelațiilor și a valorilor atipice.
- **Hărțile de căldură (heatmaps)** sunt folosite pentru a reprezenta intensitatea valorilor într-un spațiu bidimensional, evidențiind zonele cu valori ridicate sau scăzute.

Bunele practici în vizualizarea datelor includ claritatea și simplitatea, utilizarea tipului corect de grafic, utilizarea eficientă a culorilor, etichetarea clară și oferirea contextului necesar pentru a asigura transparența și credibilitatea vizualizărilor. Vizualizările trebuie să fie ușor de înțeles și să evite supraîncărcarea cu informații inutile. Alegerea corectă a tipului de grafic este esențială pentru a transmite mesajul dorit. Culorile trebuie folosite pentru a evidenția informațiile importante și pentru a diferenția între categorii. Etichetarea clară a axelor, barelor și punctelor de date este necesară pentru a oferi context și claritate. În cele din urmă, oferirea de context și specificarea sursei datelor sunt esențiale pentru a asigura transparența și credibilitatea vizualizărilor.

1.3. Tehnologii utilizate

1.3.1 Spring Framework

Spring este un cadru de dezvoltare pentru aplicații Java, care oferă o platformă cuprinzătoare pentru dezvoltarea de aplicații robuste și scalabile. Creat de Rod Johnson în 2003, Spring Framework a devenit unul dintre cele mai populare și utilizate framework-uri din lumea Java datorită flexibilității sale și a arhitecturii modulare [10].

Spring Security este un sub-proiect al framework-ului Spring care se concentrează pe furnizarea de soluții de securitate pentru aplicațiile Java [11]. Acesta oferă o infrastructură cuprinzătoare pentru autentificare și autorizare, integrându-se cu alte componente Spring pentru a asigura o securitate robustă și flexibilă. Aceasta este dezvoltată pentru a fi ușor de extins și configurat în funcție de cerințele aplicației și a utilizatorilor. Una dintre caracteristicile principale ale Spring Security este **autentificarea**. Spring Security suportă diverse metode de autentificare, inclusiv autentificarea bazată pe formulare, autentificarea HTTP Basic, și autentificarea OAuth. Configurarea autentificării se face prin utilizarea unui `AuthenticationManager`, care gestionează sursele de autentificare (de exemplu, baze de date sau servicii externe).

Spring ORM (Object-Relational Mapping) este un modul al framework-ului Spring care facilitează integrarea cu tehnologii ORM populare, cum ar fi Hibernate, JPA (Java Persistence API). ORM este o tehnologie care permite maparea obiectelor din limbajul de programare la tabelele din baza de date, simplificând accesul și manipularea datelor. Unul dintre principalele avantaje ale utilizării Spring ORM este simplificarea gestionării tranzacțiilor. Spring oferă suport pentru tranzacții declarative și programatice, permițând dezvoltatorilor să definească comportamentul tranzacțional. Spring ORM se integrează cu **Hibernate**, unul dintre cele mai populare framework-uri ORM pentru Java. Această integrare facilitează utilizarea Hibernate în contextul Spring, oferind suport pentru gestionarea sesiunilor Hibernate și configurarea simplificată a entităților. De asemenea, Spring ORM suportă **JPA**, standardul Java pentru ORM, oferind o abordare flexibilă pentru accesul la date.

1.3.2 React.js

React.js, creat de Facebook în 2013, este o bibliotecă JavaScript open-source utilizată pentru construirea interfețelor de utilizator (UI), în special a componentelor UI pentru aplicații web. React se concentrează pe dezvoltarea de aplicații web single-page. React se bazează pe

conceptul de **componente**, care sunt bucăți de cod reutilizabile ce definesc elementele UI. Fiecare componentă React este independentă și poate fi combinată cu alte componente pentru a crea interfețe complexe.

React utilizează un **Virtual DOM** (Document Object Model), care este o reprezentare în memorie a DOM-ului real. Atunci când starea unei componente React se schimbă, Virtual DOM calculează cele mai eficiente modificări pentru a actualiza DOM-ul real, minimizând costurile de performanță și îmbunătățind viteza de răspuns a aplicației. Acest proces se numește reconciliere și asigură că modificările se aplică rapid și eficient. Un alt aspect esențial al React este **JSX** (JavaScript XML), o sintaxă care permite dezvoltatorilor să scrie structuri HTML în cadrul codului JavaScript. În cadrul unei aplicații bazate pe React.js, pot fi folosite o multitudine de biblioteci cu componente predefinite. De exemplu, folosit în lucrare, Recharts.js, o librărie specializată pe crearea de grafice și diagrame utilizate în vizualizarea datelor [12]. Recharts oferă o gamă largă de componente predefinite, inclusiv grafice de bare, grafice de linie, grafice de plăcintă, grafice scatter și multe altele. Aceste componente sunt modulare și pot fi combinate și personalizate. Fiind proiectat special pentru React, Recharts se integrează perfect cu alte componente și biblioteci React. Acesta utilizează paradigmele și mecanismele React, cum ar fi state și props, pentru a gestiona datele și a actualiza graficele în timp real.

Capitolul 2. Proiectarea aplicației

Pentru a aborda în mod eficient această problemă complexă, propunem utilizarea modelului VADER din librăria NLTK, un instrument bine cunoscut pentru analiza sentimentelor în texte scrise. VADER este deosebit de eficient în interpretarea limbajului colocvial și a expresiilor utilizate frecvent pe platformele de socializare, ceea ce îl face ideal pentru scopurile noastre. Pentru a colecta și prelucra datele necesare din Reddit, vom folosi biblioteca PRAW [13], [14]. Soluția noastră va include o analiză aprofundată a postărilor de pe Reddit, concentrându-se pe elemente cheie precum titlul postării, conținutul textului (în cazul în care este prezent), comentariile aferente și numărul de upvotes, toate acestea fiind factori relevanți pentru înțelegerea și evaluarea sentimentelor utilizatorilor.

Aplicația este structurată în mai multe componente interconectate care au scopul de a crea un pipeline consistent de transmitere a datelor:

- Componenta de extragere a conținutului și analiză a sentimentelor
- Componenta de persistență a datelor și de securitate a aplicației
- Componenta de vizualizare a datelor și interacțiune cu utilizatorul

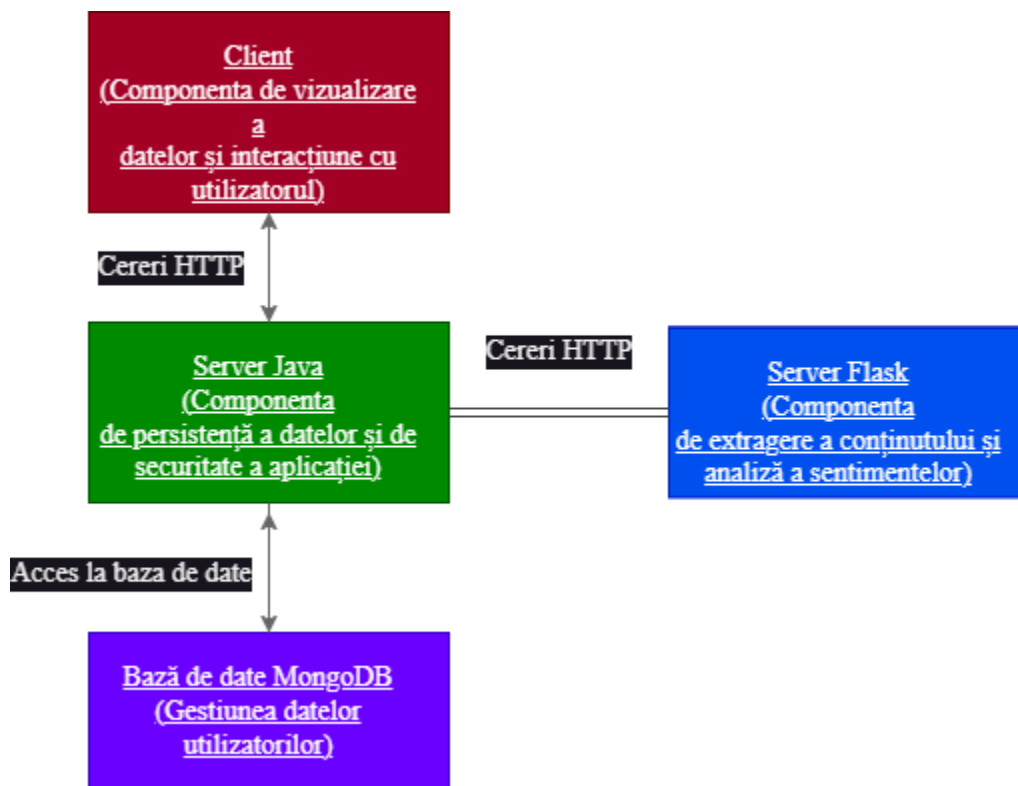


Figura 2.1. Diagramă de comunicație a componentelor

Fluxul de date al aplicației este următorul: pipeline-ul va procesa postări extrase de pe Reddit pentru un anumit topic, va extrage sentimentele, le va trimite componentei de persistență, unde vor fi salvate în baza de date MongoDB și trimise mai departe către componenta de vizualizare, unde vor fi create graficele aferente vizualizării de date.

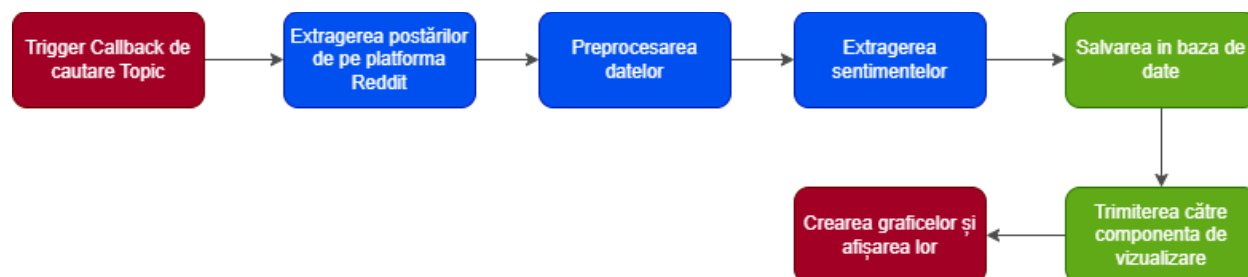


Figura 2.2. Diagramă de flux a datelor în aplicație

2.1. Componenta de extragere a conținutului și analiză a sentimentelor

2.1.1 Extragerea postărilor

Acest modul are rolul de a prelua și analiza diverse componente din postările de pe Reddit care sunt legate de un subiect specific ales de utilizator. Modulul utilizează biblioteca PRAW [15] pentru a extrage aceste componente. Pentru a putea accesa API-ul Reddit, sunt necesare două chei de acces esențiale: `client_id` și `client_secret`. Doar după ce s-a realizat conectarea la API, modulul poate trimite cereri către platforma Reddit pentru a prelua datele necesare. Extragerea postărilor implică utilizarea a trei parametri importanți: subiectul căutat, numărul maxim de postări dorite și subreddit-ul specific unde căutarea poate fi restrânsă, acesta din urmă fiind un parametru opțional. Pentru a menține eficiența procesului, există o limită în ceea ce privește numărul de postări și comentarii care pot fi extrase. Utilizatorul poate specifica numărul dorit de postări, însă numărul de comentarii este restricționat la primele 10 comentarii cele mai relevante. După ce postările sunt extrase conform parametrilor specificați, rezultatele sunt stocate într-o listă de dicționare. Această listă este apoi trimisă către un alt modul destinat analizei sentimentelor, care va prelucra informațiile extrase pentru a determina sentimentele asociate fiecărei componente a postărilor.

2.1.2 Analiza emoțiilor

Acest modul este destinat analizei sentimentelor postărilor și comentariilor de pe Reddit. Utilizând biblioteca NLTK pentru analiza sentimentelor și alte biblioteci pentru traducere și detectarea limbii, modulul extrage și prelucrează textele postărilor pentru a determina emoțiile asociate cu acestea. Modulul colectează datele despre postări pe baza unui subiect dat și a unui număr specific de postări. Textul fiecărei postări și comentariile acestora sunt analizate pentru a detecta limba și, dacă este necesar, sunt traduse în engleză.

Înainte de traducere, textul este curățat de caractere non-ASCII și de punctuația excesivă pentru a îmbunătăți acuratețea traducerii. Scorurile sentimentelor sunt calculate folosind **SentimentIntensityAnalyzer** din biblioteca NLTK, iar aceste scoruri sunt apoi interpretate pentru a clasifica textele în diferite emoții, cum ar fi "Happy" sau "Very Angry". Modulul atribuie ponderi specifice diferitelor componente ale postărilor (titlu, corpul postării, comentarii) pentru a calcula un scor mediu compus al sentimentelor. Rezultatele analizei sunt organizate într-o listă de dicționare, fiecare dicționar conținând informații despre titlu, URL, emoția determinată, scorurile sentimentelor și numărul de upvotes.

În sensul alegerii pragurilor pentru sentimente, este creat un modul suplimentar. Rolul acestui modul este de a crea un set de date pentru aplicarea unor algoritmi de clasificare. Utilizând biblioteca **pandas** și modulul de extragere al emoțiilor, se construiește un fișier cu extensia .csv în care vor fi salvate date despre diverse subiecte care generează un spectru amplu de emoții, de la emoții de extaz până la furie excesivă. Scopul principal al acestui script este de a prelua datele de sentiment dintr-un fișier, de a efectua clustering pentru a grupa aceste scoruri și de a utiliza regresia liniară pentru a modela relația dintre variabile.

Algoritmul KMeans de clustering este utilizat pentru a împărți scorurile medii de sentiment în cluster distincte. Inițial, algoritmul selectează aleatoriu puncte din setul de date care vor servi drept centre inițiale ale clusterelor. Fiecare punct din setul de date este apoi atribuit celui mai apropiat centru de cluster pe baza distanței euclidiene. Centrele clusterelor sunt actualizate prin calcularea mediei aritmetice a tuturor punctelor din fiecare cluster. Acest proces de atribuire și actualizare se repetă până când centrele clusterelor nu mai schimbă semnificativ, indicând convergența. Rezultatul final al acestui proces include centrele clusterelor, care sunt afișate, și etichetele fiecărui punct de date, indicând la care cluster aparține fiecare punct.

Regresia liniară este folosită pentru a modela relația dintre variabilele independente (scorurile titlului, corpului postării și comentariilor) și variabila dependentă (scorul mediu de sentiment). Modelul matematic folosește o ecuație liniară pentru a găsi linia dreaptă care minimizează eroarea pătratică dintre predicțiile modelului și valorile reale. Modelul de regresie liniară este antrenat pe datele noastre pentru a găsi coeficienții optimi care minimizează eroarea pătratică medie. După antrenare, coeficienții (greutățile) sunt extrași pentru a înțelege importanța fiecărei variabile independente în predicția variabilei dependente. Modelul este apoi utilizat pentru a prezice valorile variabilei dependente.

2.2. Componenta de securitate și persistență a datelor

Rolul acestei componente este esențial pentru preluarea datelor de la componenta menționată anterior, gestionarea, securizarea și transmiterea lor către componenta de vizualizare. În plus, această componentă se ocupă de administrarea informațiilor și datelor utilizatorului. Având o conexiune strânsă cu celelalte module ale aplicației, ea acționează ca un intermediar, facilitând schimbul de informații între diferitele părți ale sistemului.

Datele utilizatorului sunt stocate într-o bază de date MongoDB, într-un singur document. Acest document conține informații critice ale utilizatorului, cum ar fi detaliile de autentificare, și din acest motiv necesită implementarea unor măsuri de securitate riguroase. Aceste măsuri sunt

menite să protejeze datele și să limiteze operațiile care pot fi efectuate asupra lor, asigurând astfel integritatea și confidențialitatea informațiilor. Alegerea stocării datelor într-un document MongoDB a fost influențată de natura volatilă a informațiilor primite de la componenta precedentă. Postările extrase pot avea multiple câmpuri lipsă, cum ar fi absența unui corp textual al postării sau lipsa comentariilor. Astfel, flexibilitatea modelului de stocare document-centric permite gestionarea eficientă a acestor date variabile.

User Document
<code>_id: ObjectID,</code>
<code>username: String,</code>
<code>password: String,</code>
<code>email: String,</code>
<code>historyModel: [</code>
<code> {</code>
<code> topic: String,</code>
<code> emotion: String,</code>
<code> postModel: [{</code>
<code> title: String,</code>
<code> url: String,</code>
<code> emotion: String,</code>
<code> upvotes: Integer,</code>
<code> avgScore: Float,</code>
<code> bodyScore: Float,</code>
<code> titleScore: Float,</code>
<code> commentScoresAvg: Float</code>
<code> }]</code>
<code> }]</code>

Figura 2.2.1. Model document salvat în serverul MongoDB

Securizarea aplicației este gestionată prin utilizarea de tokeni **JWT**. După crearea unui cont în prealabil, la logarea utilizatorului în cadrul aplicației, este creat un token JWT care conține informații esențiale despre utilizator, cum ar fi numele de utilizator, ID-ul și adresa de email. Tokenul este semnat folosind un algoritm de semnare și o cheie secretă pentru a asigura integritatea și autenticitatea acestuia. Tokenii sunt integrați în configurația de securitate a aplicației. Aceasta configurare specifică faptul că aplicația utilizează o politică de sesiune fără stare (**STATELESS**), ceea ce înseamnă că fiecare cerere trebuie să fie autorizată individual folosind tokenul JWT. Pentru procesarea lor se adaugă un filtru de securitate, înaintea filtrului de autentificare standard, pentru a asigura că tokenii sunt verificați înainte de orice altă procesare. Configurația de securitate permite accesul neautentificat doar la endpoint-urile de autentificare, în timp ce toate celelalte cereri necesită autentificare validă prin token JWT.

2.3. Componenta de vizualizare

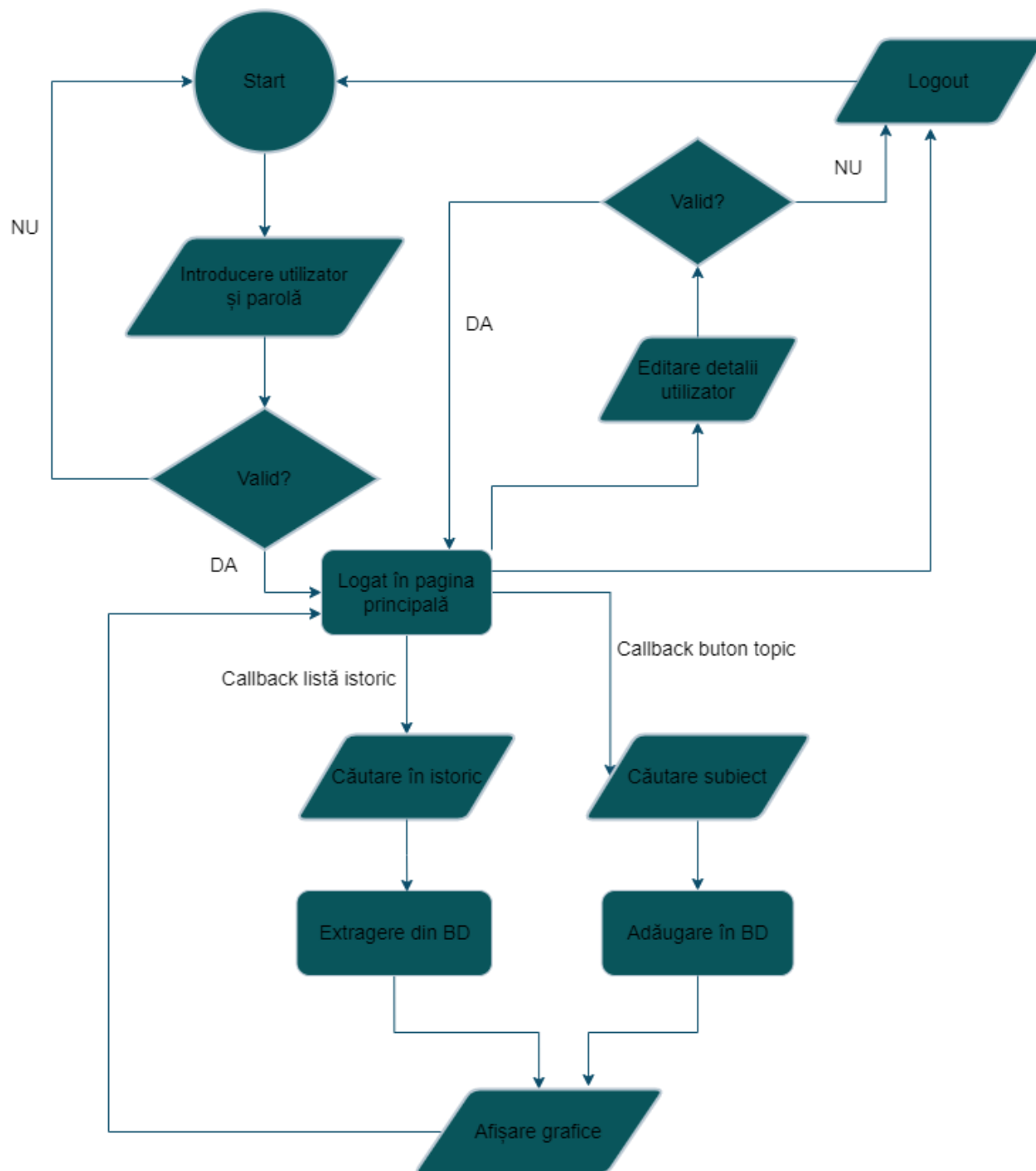


Figura 2.3.1. Schema de lucru pentru componenta de Frontend a aplicației

Flowchart-ul oferit descrie fluxul de lucru al componentei de vizualizare și interacțiunea cu utilizatorul în aplicație, evidențiind pașii de la autentificare până la afișarea graficelor și gestionarea interacțiunilor utilizatorului.

Procesul începe cu deschiderea aplicației de către utilizator, urmată de introducerea credențialelor de autentificare (nume de utilizator și parolă). Sistemul validează aceste credențiale; dacă sunt corecte, utilizatorul este redirecționat către pagina principală a aplicației. În caz contrar, utilizatorul este invitat să reintroducă informațiile corecte pentru autentificare. Odată autentificat, utilizatorul ajunge pe pagina principală, unde are posibilitatea de a interacționa cu diferite funcționalități ale aplicației. Printre aceste funcționalități se numără căutarea în istoricul său de activitate sau căutarea unui subiect nou. Dacă utilizatorul alege să caute în istoric, sistemul extrage datele relevante din baza de date. Aceste date sunt apoi afișate utilizatorului sub formă de grafice, oferindu-i o vizualizare clară și detaliată a informațiilor solicitate. Pe de altă parte, dacă utilizatorul optează pentru căutarea unui subiect nou, sistemul adaugă informațiile relevante în baza de date și procedează la afișarea graficelor corespunzătoare. În tot acest timp, utilizatorul are posibilitatea de a edita detaliile contului său, cum ar fi informațiile personale sau setările de securitate. Sistemul validează aceste modificări și le actualizează în baza de date. De asemenea, utilizatorul poate oricând alege să se deconecteze din aplicație, moment în care fluxul se reia de la început, permițând unei alte sesiuni de autentificare să aibă loc.

Capitolul 3.Implementarea aplicației

3.1. Componenta de extragere a conținutului și analiză a sentimentelor

Primul pas în extragerea emoțiilor de pe platforma Reddit este conectarea la API-ul lor. După conectare, următorul pas este extragerea elementelor dorite din postări. Acest lucru este implementat în funcția principală a modului, `fetch_submissions`. Această funcție gestionează căutarea și extragerea postărilor de pe Reddit. Aceasta primește trei parametri: `query` (subiectul căutat), `nr_posts` (numărul de postări dorit) și `subreddit` (subreddit-ul specific). Dacă nu este specificat un subreddit, căutarea se efectuează implicit pe toate subredditurile (`all`). De asemenea, dacă nu este specificat un număr de postări, limita implicită este de 10 postări.

Funcția utilizează metoda `search` pentru a găsi postările relevante pe baza query-ului dat și a limitei de postări. Pentru fiecare postare găsită, funcțiile `fetch_post_body` și `fetch_comments` sunt apelate pentru a extrage corpul postării și comentariile acesteia. Datele extrase sunt stocate într-un dicționar `post_comments_dict`, unde titlul postării este cheia, iar valoarea este un alt dicționar ce conține URL-ul postării, numărul de upvotes, corpul postării și comentariile.

```
def fetch_submissions(query, nr_posts, subreddit=None):
    post_comments_dict = {}

    if subreddit is None:
        subreddit = 'all'
    if nr_posts is None:
        nr_posts = 10

    search_results = reddit.subreddit(subreddit).search(query,
limit=nr_posts)

    for submission in search_results:
        post_body = fetch_post_body(submission)
        title, url, comments_list, score = fetch_comments(submission)
        post_comments_dict[title] = {'url': url, 'upvotes': score,
'post_body': post_body, 'comments': comments_list}

    return post_comments_dict
```

Listing 3.1.1. Codul pentru extragerea emoțiilor.

Următorul pas în extragerea și analiza emoțiilor este crearea unui set de date descriptiv pentru problema studiată, în sensul alegerii pragurilor pentru sentimentele exprimate în postări.

Pentru crearea setului de date, am ales o abordare bazată pe un dicționar de subiecte care încearcă să atingă intensități ale sentimentelor de fiecare nivel. Astfel, pentru crearea setului, se apelează funcția `fetch_submissions`, din modulul de extragere a conținutului pentru fiecare subiect în parte.

```
def main():
    topics = ["Russia", "Ukraine", "Palestine", "Multiversus",
"Fortnite", "Iran", "Donald Trump", "Dogs", "Cats",
"Teenage", "Boys", "Kids", "War", "Politics", "Angry",
"Happy", "Artificial Intelligence", "Internet",
"Programming", "Capybara", "Sad", "Music", "Lifestyle",
"Sun", "Zodiac", "Planets"]

    df = pd.DataFrame()

    for topic in topics:
        print(f"Fetching data for topic: {topic}")
        reddit_data = emotion_analysis.create_dataset_emotions(topic,
limit=40, subreddit='all')
        df = pd.concat([df, pd.DataFrame(reddit_data)],
ignore_index=True)

    output_file_path = 'reddit_dataset.csv'
    df.to_csv(output_file_path, index=False)

    print(f"Dataset saved to {output_file_path}")
```

Listing 3.1.2. Codul pentru crearea setului de date.

title	url	emotion	body_score	title_score	comment_s cores_avg	upvotes	avg_score
What does the west get wrong about Russia?	https://www.reddit.com/r/AskARussian/comments/1944suz/what_does_the_west_get_wrong_about_russia/	Irritated	0.0258	-0.4767	0.13335	68	-0.105846667

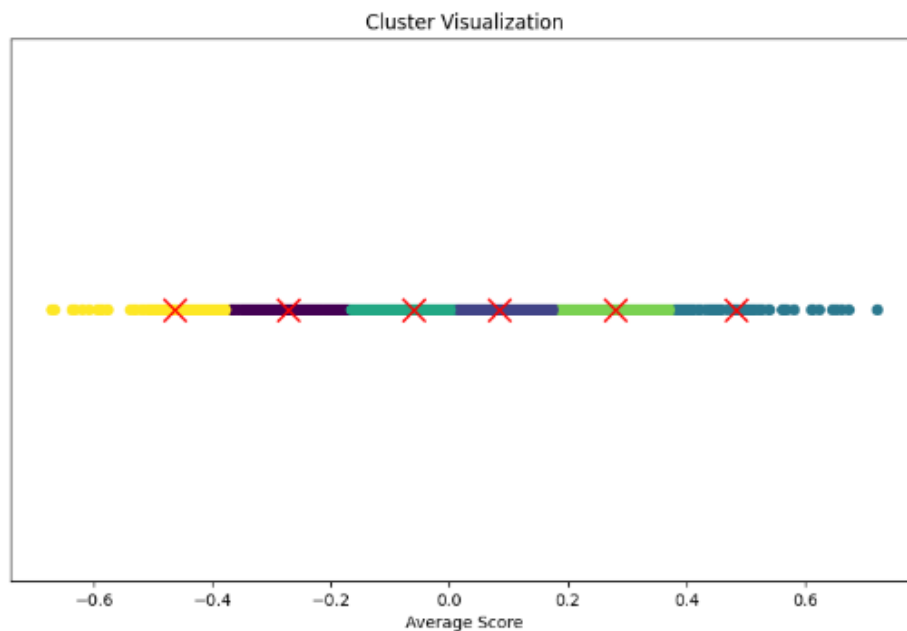
Tabel 3.1.1. Exemplu de linie obținută în setul de date

În continuare, după obținerea setului de date, vom aplica algoritmul de clusterizare **K-Means** pentru găsirea unor praguri realiste în sensul împărțirii sentimentelor în grupuri caracteristice. De asemenea, vom mai aplica o regresie liniară pentru calculul coeficienților care intră în ecuația calculului final al sentimentului mediu per postare.

Pentru pregătirea aplicării algoritmului K-Means, se citesc datele din fișierul creat anterior și se extrage coloana `avg_score` care se redimensionează într-un array cu o singură coloană. Apoi se configurează și se aplică algoritmul K-Means pentru a grupa datele în n clustere. După aplicarea algoritmului, se extrag centroizii grupărilor și sunt sortați pentru crearea unei vizualizări. În vizualizarea rezultată sunt expuse valorile intrărilor și centroizii calculați pe axa X, pe axa Y fiind prezente doar valori de 0.

```
sentiment_scores = df['avg_score'].values.reshape(-1, 1)
kmeans = KMeans(n_clusters=6, random_state=10)
kmeans.fit(sentiment_scores)
cluster_centers = sorted(kmeans.cluster_centers_.flatten())
```

Listing 3.1.3. Codul pentru crearea clusterelor K-means.



Figură 3.1.1. Exemplu de clusterizare pentru $n=6$

Astfel, după reprezentarea graficului, numărul de clustere a fost ales empiric în funcție de numărul de grupări în care setul de date a avut grupările distribuite în cel mai egal mod. Rezultatul a fost ales a fi o grupare pe 6 clustere cu următorii centroizi:

`[-0.463, -0.272, -0.060, 0.084, 0.279, 0.483]`

Regresia liniară este un algoritm de învățare supervizată care modelează relația dintre o variabilă dependentă (y) și una sau mai multe variabile independente (X) folosind o linie dreaptă. Acest algoritm este folosit pentru a prezice scorul mediu (`avg_score`) bazat pe scorurile titlului (`title_score`), scorurile conținutului (`body_score`) și scorurile medii ale comentariilor (`comment_scores_avg`). Coeficienții modelului oferă informații despre importanța fiecărei variabile independente în prezicerea scorului mediu. În `scikit-learn`, algoritmul de regresie liniară este implementat în modulul `sklearn.linear_model`. Implementarea utilizează metoda celor mai mici pătrate pentru a estima coeficienții modelului.

```
regressor = LinearRegression()
regressor.fit(X, y)
weights = regressor.coef_
title_weight, body_weight, comment_weight = weights
```

Listing 3.1.3. Codul pentru aplicarea regresiei liniare.

Performanța modelului de regresie este evaluată folosind două metrici principale: Mean Squared Error (MSE) și coeficientul de determinare (R-squared). MSE măsoară media pătratelor erorilor de predicție, un MSE mai mic indicând un model mai precis. R-squared măsoară proporția variabilității din variabila dependentă explicată de model, un R-squared mai aproape de 1 indicând un model mai bun. Rezultatele aplicării acestui algoritm pentru soluția propusă sunt următoarele:

```
Title: 0.4847513606394562 Body: 0.3043335683537195 Comment:
0.471818219074203
Mean Squared Error: 0.000608087368300235
R-squared: 0.9902469407066096
```

În continuarea alegerii valorilor necesare, urmează modulul de procesare efectivă a emoțiilor. Modulul dispune de diferite funcții cu rol de preprocesare și postprocesare a datelor. Datele sunt preluate din modulul de extragere a conținutului și se aplica următoarele preprocesari:

- Limitarea semnelor de punctuație: algoritmul are probleme în procesarea textului care conține semne de punctuație continue care depășesc numărul 3 (Ex: "!!!" rămâne "!!!", dar "!!!!" devine "!!!", deoarece "!!!!" face ca valoarea returnată de algoritm să fie automat 0).
- Detecția limbii și traducerea: pentru fiecare titlu și comentariu, se detectează limba și se traduce textul în engleză dacă este necesar. Această preprocesare are rolul de a traduce textul primit deoarece algoritmul poate procesa doar texte din limba engleză. O nouă problemă adusă este pierderea sentimentelor transmise la traducerea textului.

- Eliminarea emoticoanelor: pentru funcționalitatea algoritmului de detecție și traducere a limbii, este nevoie de eliminarea emoji-urilor deoarece librăriile propuse nu pot procesa astfel de date.

```
def analyze_emotions(posts):
    analyzer = SentimentIntensityAnalyzer()
    scores = []

    title_weight = 0.484
    body_weight = 0.304
    comment_weight = 0.471

    for title, data in posts.items():
        if detect_language(title):
            title = translate_text(title)
            translated_comments = []
            for comment in data['comments']:
                if detect_language(comment):
                    comment = translate_text(comment)
                    translated_comments.append(comment)

            title_score =
analyzer.polarity_scores(limit_punctuation(title))['compound']
            comment_scores =
[analyzer.polarity_scores(limit_punctuation(comment))['compound'] for
comment in translated_comments]

if comment_scores and data['post_body']:
    body_score =
analyzer.polarity_scores(limit_punctuation(data['post_body']))['compou
nd']
    avg_score = title_weight * title_score + body_weight * body_score
+ comment_weight * statistics.mean(comment_scores)
```

Listing 3.1.4. Codul pentru procesarea datelor.

Astfel, după preprocesarea datelor, se aplică NLTK `SentimentIntensityAnalyzer` pentru analiza textului pentru fiecare element al postării, în funcție de existența sa. Pentru postprocesarea datelor, se urmărește clasificarea pe categorii de sentimente în funcție de intensitatea returnată de către algoritmul menționat anterior. Clasificarea se face prin funcția `get_emotion`, care preia intensitatea și aplică o conversie bazată pe pragurile calculate anterior în algoritmul K-Means. În plus, rezultatele obținute în urma aplicării regresiei liniare sunt aplicate în calculul sentimentelor medii per postare. Pentru fiecare element care contribuie la sentimentul final sunt aplicate ponderile aferente. În final, modulul reorganizează datele într-un nou dicționar care este trimis mai departe către componenta următoare.

3.2. Componenta de securitate și persistență

Implementarea acestei componente, care servește ca un intermediar între celelalte componente ale aplicației, este realizată având în vedere atât persistența datelor, cât și securitatea solicitărilor.

Persistența datelor este implementată utilizând framework-ul Spring, folosindu-ne de module precum JPA, Hibernate pentru facilitarea conexiunii și diverselor apeluri către baza de date MongoDB. Pentru salvarea și modelarea datelor, am folosit clase de tip `@Model` și adnotări Lombok pentru generarea de cod de tip boilerplate. În acest mod, pentru modelarea datelor vom folosi 3 astfel de clase: clasa `User`, clasa `History` și clasa `Post`. Având în vedere faptul că baza de date conține un singur document, și anume documentul bazat pe clasa `User`, celelalte 2 clase devin doar unelte de procesare a datelor.

```
@AllArgsConstructor
@NoArgsConstructor
@Getter
@Setter
public class UserModel {
    @Id
    private String id;
    private String username;
    private String password;
    private String email;
    List<HistoryModel> historyModels;
    public UserModel(String username, String password) {
        this.username = username;
        this.password = password;
    }
}
```

Listing 3.2.1. Codul pentru clasa `UserModel`.

Abordarea aleasă pentru implementarea acestei componente, constă astfel în patru straturi pentru fiecare astfel de model. Primul strat este stratul prezentat mai sus, stratul `Model`, acesta conține entitățile JPA care sunt mapate la tabelele bazei de date. Următorul strat, stratul `Repository` gestionează interacțiunea cu baza de date. Utilizând Spring Data JPA, acest strat oferă metode CRUD (Create, Read, Update, Delete) și posibilitatea de a defini interogări personalizate. Stratul `Service` conține logica de afaceri a aplicației. Acest strat folosește metodele definite în stratul `Repository` pentru a manipula datele. Ultimul strat, stratul `Controller`

gestionează cererile HTTP. Acesta primește solicitările de la utilizatori, apelează metodele din stratul Service pentru a procesa cererile și returnează răspunsurile corespunzătoare. Această arhitectură permite o separare clară a responsabilităților și de procesare fluidă a datelor.

```
public ResponseEntity<UserRes> getUserByIdFromDB(String id, String
token){
    if (validationUtil.verifyIdentity(token,id)){
        UserModel userModel =
userRepository.findById(id).orElse(null);
        if(userModel == null){
            return new ResponseEntity<>(null,HttpStatus.NOT_FOUND);
        }
        else return new ResponseEntity<>(new
UserRes(userModel.getUsername(),userModel.getEmail()),HttpStatus.OK);
    }
    else return new ResponseEntity<>(null,HttpStatus.FORBIDDEN);
}

public ResponseEntity<UserModel> deleteUserByIdFromDB(String id){
    UserModel userModel = userRepository.findById(id).orElse(null);
    if(userModel != null){
        userRepository.deleteById(id);
        return new ResponseEntity<>(userModel,HttpStatus.OK);
    }
    else return new ResponseEntity<>(null,HttpStatus.NOT_FOUND);
}
```

Listing 3.2.2. Exemplu de funcții pentru operații CRUD asupra documentului User.

Tot în cadrul acestei componente sunt prezente și modulele de securizare ale aplicației. Modalitatea aleasă este o abordare stateless, cu utilizarea tokenilor de tip JWT [16]. Modulul folosește de asemenea, Spring Security pentru gestiunea autorizărilor și a autentificării utilizatorilor aplicației. Pentru autentificarea propriu-zisă, este creat un controller care se ocupă de operațiile de logare și de crearea conturilor. Operațiile din cadrul acestui controller sunt permise oricărui utilizator, fie că este autentificat sau nu. Criptarea parolelor este realizată prin algoritmul Bcrypt.

```
@ResponseBody
@PostMapping("/login")
public ResponseEntity login(@RequestBody LoginReq loginReq){
    try{
        Authentication authentication = authenticationManager
            .authenticate(new
UsernamePasswordAuthenticationToken(loginReq.getUsername(),loginReq.ge
tPassword()));
```

```

        String username = authentication.getName();
        UserModel userModel = userService.getUserByUsername(username);
        String token = jwtUtil.createToken(userModel);
        LoginRes loginRes = new
LoginRes(userModel.getId(),username,token);

        return ResponseEntity.ok(loginRes);
    }
    catch (BadCredentialsException e){
        ErrorRes errorRes = new
ErrorRes(HttpStatus.BAD_REQUEST,"Invalid username or password");
        return
ResponseEntity.status(HttpStatus.BAD_REQUEST).body(errorRes);
    }
    catch (Exception e){
        ErrorRes errorResponse = new ErrorRes(HttpStatus.BAD_REQUEST,
e.getMessage());
        return
ResponseEntity.status(HttpStatus.BAD_REQUEST).body(errorResponse);
    }
}

```

Listing 3.2.3. Funcție pentru gestionarea logării unui utilizator.

Pentru gestionarea JWT-urilor am creat de asemenea, o clasă care se va ocupa de toate operațiile care implică acești tokeni. Această clasă, denumită `JwtUtil`, are mai multe câmpuri, inclusiv o cheie secretă folosită pentru semnarea tokenurilor, durata de valabilitate a tokenurilor (stabilită la 60 de minute), un parser JWT pentru analizarea tokenurilor, și câmpuri pentru antetul și prefixul tokenurilor din solicitările HTTP (Bearer). Constructorul clasei inițializează parserul JWT cu cheia secretă. Clasa oferă o metodă pentru crearea unui token JWT pentru un utilizator specific, folosind datele utilizatorului: numele de utilizator, id-ul și adresa de email. Tokenul este semnat cu un algoritm de criptare HS256 și are o dată de expirare stabilită la momentul creării plus durata de valabilitate. Pentru a analiza un token JWT, clasa oferă o metodă care returnează claim-urile conținute în token. De asemenea, există o metodă care extrage tokenul din antetul unei solicitări HTTP și returnează claim-urile, gestionând eventualele excepții precum tokenurile expirate sau invalide.

```

public String createToken(UserModel userModel){

    Claims claims = Jwts.claims().setSubject(userModel.getUsername());

    claims.put("id",userModel.getId());
    claims.put("email",userModel.getEmail());

    Date tokenCreateTime = new Date();
    Date tokenValidity = new Date(tokenCreateTime.getTime() +
TimeUnit.MINUTES.toMillis(accessTokenValidity));

```

```

        return Jwts.builder()
            .setClaims(claims)
            .setExpiration(tokenValidity)
            .signWith(SignatureAlgorithm.HS256, secret_key)
            .compact();
    }

    public Claims parseJwtClaims(String token){
        return jwtParser.parseClaimsJws(token).getBody();
    }

    public Claims resolveClaims(HttpServletRequest req){
        try{
            String token = resolveToken(req);
            if(token != null){
                return parseJwtClaims(token);
            }
            return null;
        }
        catch (ExpiredJwtException ex) {
            req.setAttribute("expired", ex.getMessage());
            throw ex;
        }
        catch (Exception ex) {
            req.setAttribute("invalid", ex.getMessage());
            throw ex;
        }
    }
}

```

Listing 3.2.4. Funcții de gestiune a tokenilor JWT

În continuare, pentru implementarea logicii securității am creat clasa de configurare `SecurityConfig`. Această bucată de cod definește o metodă care configurează lanțul de filtre de securitate pentru o aplicație web utilizând Spring Security. Metoda `securityFilterChain` primește un obiect `HttpSecurity` ca parametru și aruncă o excepție de tipul `Exception`. Scopul acestei metode este de a configura politicile de securitate pentru diferite rute și de a adăuga un filtru personalizat pentru autorizare bazată pe JWT.

```

@Bean
public SecurityFilterChain securityFilterChain(HttpSecurity http)
throws Exception{

    http.csrf().disable()
        .authorizeRequests()
        .requestMatchers("/api/auth/**").permitAll()
        .anyRequest().authenticated()

    .and().sessionManagement().sessionCreationPolicy(SessionCreationPolicy
        .STATELESS)
}

```

```

        .and().addFilterBefore(jwtAuthorizationFilter,
UsernamePasswordAuthenticationFilter.class);

    return http.build();
}

```

Listing 3.2.5. Funcție pentru configurarea securității.

În primul rând, protecția CSRF (Cross-Site Request Forgery) este dezactivată prin apelul `http.csrf().disable()`. Acest lucru este comun în aplicațiile care folosesc JWT pentru autentificare, deoarece JWT oferă deja un nivel de securitate împotriva atacurilor CSRF. Apoi, sunt configurate regulile de autorizare pentru solicitările HTTP. Toate solicitările care se potrivesc cu ruta `/api/auth/**` sunt permise fără autentificare, ceea ce înseamnă că utilizatorii pot accesa orice endpoint sub acest prefix fără a fi necesară autentificarea. Pentru toate celelalte solicitări (`anyRequest()`), autentificarea este obligatorie. Metoda configurează, de asemenea, politica de gestionare a sesiunilor. Setarea `SessionCreationPolicy.STATELESS` specifică faptul că aplicația nu va crea sesiuni de utilizator pe server. Acest lucru este tipic pentru aplicațiile care folosesc JWT, deoarece toate informațiile de autentificare sunt stocate în token și nu în sesiuni de server. În final, este adăugat un filtru personalizat `jwtAuthorizationFilter` înainte de filtrul standard de autentificare cu nume de utilizator și parolă (`UsernamePasswordAuthenticationFilter`). Acest filtru personalizat este responsabil pentru extragerea și validarea tokenurilor JWT din solicitările HTTP pentru a autoriza utilizatorii.

3.3. Componenta de vizualizare

Ultima componentă a aplicației este interfața web, care este implementată utilizând framework-ul React.js. Interfața este structurată folosind o arhitectură bazată pe componente, ceea ce înseamnă că fiecare parte a interfeței este împărțită în componente reutilizabile și independente. Pentru gestionarea navigării este folosită librăria `ReactRouter`. Aplicația dispune de 5 rute navigabile de către utilizator. Cele 5 pagini sunt următoarele:

- Pagina de logare
- Pagina de creare a contului utilizatorului
- Pagina de editare a detaliilor utilizatorului
- Pagina principală de vizualizare a sentimentelor
- Pagina de testare a extragerii sentimentelor pentru un text dat de utilizator

Componenta dispune de asemenea, de un script care conține toate apelurile către API-ul descris anterior.

```

export async function searchTopic(id,topic,limit = 10,subreddit =
'all',token) {

```

```

    try {
        const response = await
fetch(`${baseUrl}/${id}/history?topic=${topic}&limit=${limit}&subreddi
t=${subreddit}`, {
    method: 'POST',
    headers: {
        'Content-Type': 'application/json',
        'Authorization': `Bearer ${token}`
    }
});
    return await response.json();
} catch (error) {
    console.error('Error:', error);
}
}

```

Listing 3.3.1. Exemplu de apel către componenta de persistență

Funcționalitățile de bază ale aplicației propuse sunt disponibile odată cu autentificarea utilizatorului. La deschiderea ei, utilizatorul va fi direcționat către pagina de logare, unde își va putea introduce datele de autentificare (numele de utilizator și parola), în cazul în care acesta și-a creat un cont în prealabil. În cazul în care utilizatorul introduce un set greșit de credențiale, acesta va fi anunțat prin intermediul unei alerte care îi va comunica acest lucru.

Figura 3.3.1. Pagina de logare.

Dacă utilizatorul nu și-a creat un cont deja, acesta poate accesa pagina de înregistrare print-un click pe textul “*Don’t have an account?*”. Odată ce utilizatorul a accesat această pagină, acesta va trebui să introducă datele cerute. Restricția legată de datele introduse este crearea unui cont cu un nume de utilizator deja existent în baza de date. După completarea datelor, utilizatorul este redirecționat către pagina de logare.

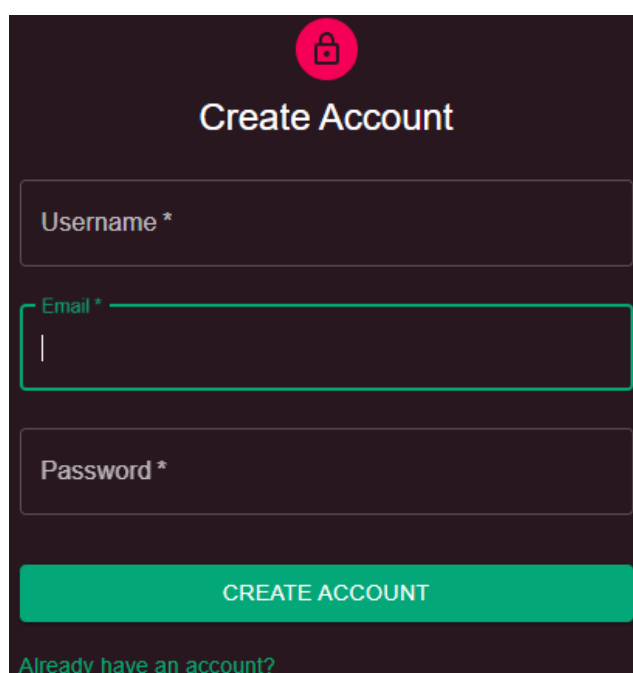
The image shows a 'Create Account' form on a dark purple background. At the top center is a red circular icon with a white padlock. Below it, the text 'Create Account' is displayed in white. The form consists of three input fields: 'Username *', 'Email *', and 'Password *', each with a light purple border. The 'Email *' field is currently active, indicated by a red border and a vertical cursor. Below the input fields is a red button with the text 'CREATE ACCOUNT' in white. At the bottom of the form, the text 'Already have an account?' is written in a smaller, light purple font.

Figura 3.3.2. Pagina de înregistrare

După autentificarea cu succes, utilizatorul este redirecționat către pagina principală a aplicației. Pe această pagină, utilizatorul poate vizualiza istoricul căutărilor, informațiile personale și o interfață pentru căutarea de subiecte.

Istoricul căutărilor este prezentat sub forma unei liste verticale, oferind o structură clară și ușor de urmărit. Fiecare element din listă conține informații succinte despre căutările anterioare, incluzând numele subiectului căutat și sentimentul mediu calculat pentru acel subiect. Numele căutării este colorat în funcție de sentimentul exprimat. Dacă utilizatorul dorește să ștergă o anumită căutare din istoricul său, acesta poate face acest lucru foarte simplu prin apăsarea butonului dedicat, care este poziționat în partea dreaptă a fiecărui element din listă. Acest buton este reprezentat printr-o pictogramă sub formă de coș de gunoi, oferind o indicație vizuală clară a funcționalității sale. Pe lângă opțiunea de ștergere, lista de căutări oferă și posibilitatea de a vizualiza detalii suplimentare pentru fiecare element. Atunci când utilizatorul face click pe un element din listă, se deschide o vizualizare detaliată specifică acelui element, situată în partea dreaptă a listei. Această vizualizare oferă informații mai detaliate despre căutarea respectivă, permițând utilizatorului să examineze mai în profunzime datele și rezultatele asociate cu acea căutare.

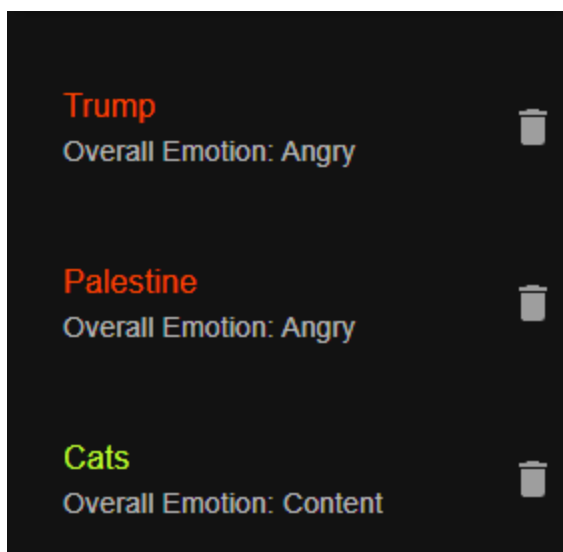


Figura 3.3.3. Lista de căutări.

Tot pe această pagină, sub butonul de setări este prezentă o serie de funcționalități care pot fi accesate. Prima dintre ele este o redirecționare către pagina în care utilizatorul își poate schimba informațiile aferente contului. Următoarea opțiune oferă posibilitatea de delogare. După selectarea acestei opțiuni se face o redirectare către pagina de logare. Ultima opțiune trimite utilizatorul către o pagină de testare a analizei sentimentelor pentru o secvență de text introdusă de către el.

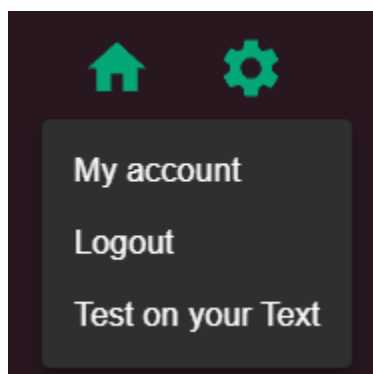


Figura 3.3.4. Interfața pentru setările aplicației.

Pentru testarea analizei sentimentelor din text, interfața dispune de o căsuță de text în care poate fi introdus text în orice limbă, deoarece acesta este tradusă automat. Însă, această traducere vine la pachet cu o pierdere a sentimentelor transmise. Ca exemplu, am inserat același text prima oară în limba engleză și următoarea dată în limba română. Putem observa că sentimentele generale sunt aceleași, dar intensitatea lor diferă.

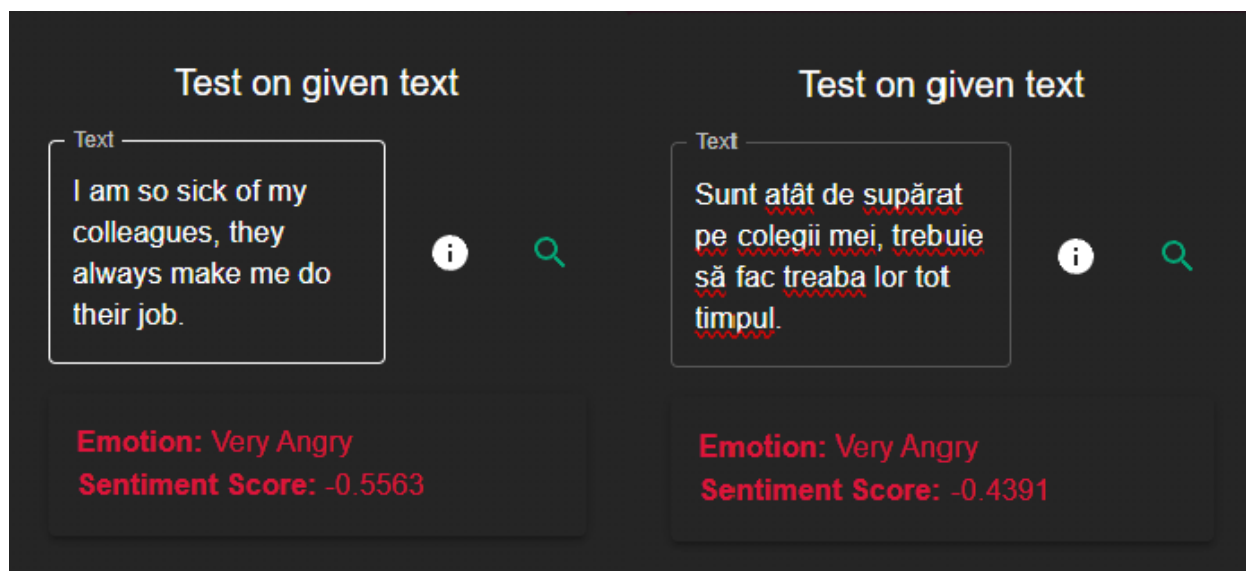


Figura 3.3.5. Comparație între analiza sentimentelor pentru limbi diferite.

De asemenea, funcționalitatea principală a interfeței este reprezentarea grafică a sentimentelor extrase din postări. Componenta dispune de 3 câmpuri ce trebuie completate astfel:

- *Topic* – Subiectul despre care se vor căuta postările
- *Subreddit* – În cazul în care se vrea restrângerea căutării pentru anumite sub-forumuri de pe platforma Reddit. Acest câmp este unul optional, astfel, în cazul în care nu este completat, căutarea va fi făcută pe toată platforma.
- *Nr. Of Posts* – Se setează numărul de postări ce vor fi returnate de către componenta de extragere.

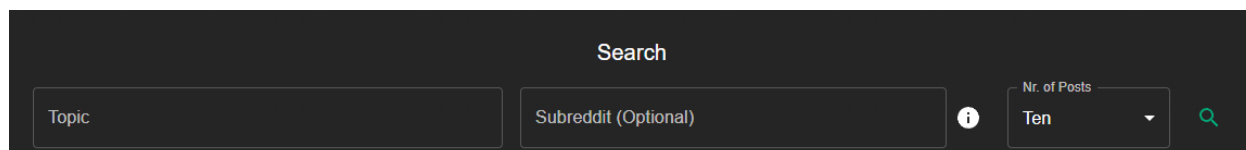


Figura 3.3.6. Câmpurile care trebuie completate pentru căutare.

După ce se trece de acest pas, rezultatele căutării sunt returnate către interfață, unde se va gestiona vizualizarea acestor date. Utilizând biblioteca Recharts.js, datele preluate de pe platforma Reddit sunt astfel organizate sub formă de grafice ce modelează datele pentru a găsi corelații între sentimentele exprimate în postările preluate.

Prima astfel de vizualizare este afișarea tuturor postărilor sub forma unei liste asemănătoare cu lista istoricului. Titlul postării este primul atribut al fiecărui element, urmat de sentimentul general și apoi de un hyperlink către postarea dată.

Pentru alegerea paletelor de culori asociate diferitelor stări emoționale, am optat pentru o gamă de culori care reflectă intensitatea și natura fiecărui sentiment. Culorile variază de la nuanțe intense de roșu și portocaliu pentru sentimentele negative, cum ar fi furia, la nuanțe de verde și galben pentru sentimentele pozitive, cum ar fi mulțumirea și fericirea. Sentimentele neutre sunt reprezentate de gri, oferind un echilibru vizual între extremele emoționale.



Figura 3.3.7. Lista postărilor.

Următoarele elemente grafice prezente în interfață sunt diferitele grafice în care elementele postărilor sunt distribuite astfel:

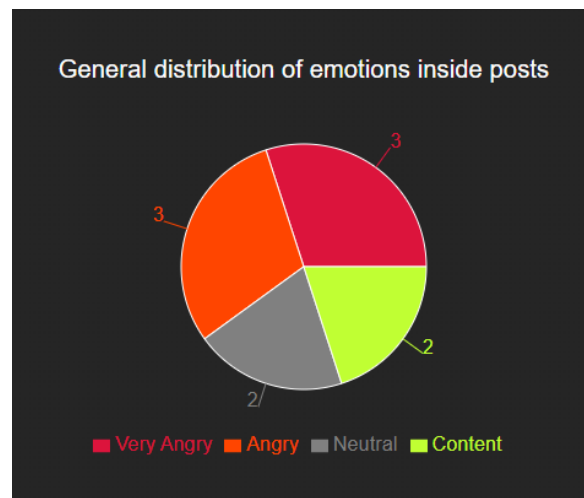


Figura 3.3.8. PieChart pentru distribuția emoțiilor.

- Distribuția sentimentelor în postări: Acest *PieChart* arată procentajul fiecărui tip de emoție prezent în textul postărilor. Fiecare segment al graficului reprezintă o emoție specifică, iar dimensiunea fiecărui segment este proporțională cu frecvența respectivei emoții în postări. Acesta ajută la identificarea tendințelor generale și a prevalenței anumitor emoții în întreaga postare.

- Distribuția sentimentelor medii ale comentariilor: Similar cu primul *PieChart* , fiecare segment reprezintă o emoție specifică, iar dimensiunea segmentului este proporțională cu frecvența acelei emoții în comentarii. Acest tip de grafic este util pentru a înțelege reacțiile și interacțiunile utilizatorilor la conținutul postat. Emoțiile exprimate în comentarii pot reflecta feedback-ul și starea de spirit a comunității față de postările inițiale.
- Punctajul corpului postărilor corelat cu media punctajelor comentariilor: Acest grafic cu bare ilustrează comparația între punctajele postărilor și media punctajelor comentariilor aferente fiecărei postări. Fiecare bară verticală reprezintă o postare specifică și punctajele corespunzătoare acestuia.

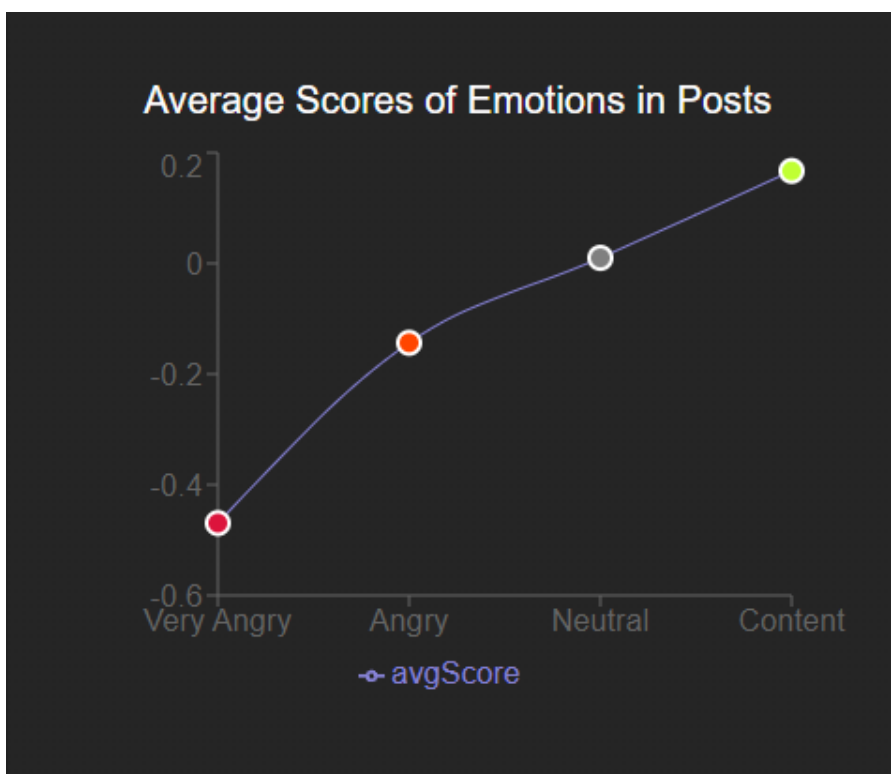


Figura 3.3.9. *LineChart* pentru vizualizarea intensităților emoțiilor din postări.

- Punctajele medii ale emoțiilor în postări: Acest grafic liniar prezintă punctajele medii ale emoțiilor exprimate în postări. Axa verticală indică valoarea medie a punctajelor emoționale, variind de la negativ la pozitiv. Axa orizontală enumeră diferitele emoții. Graficul diferența intensităților ale punctajelor emoțiilor.

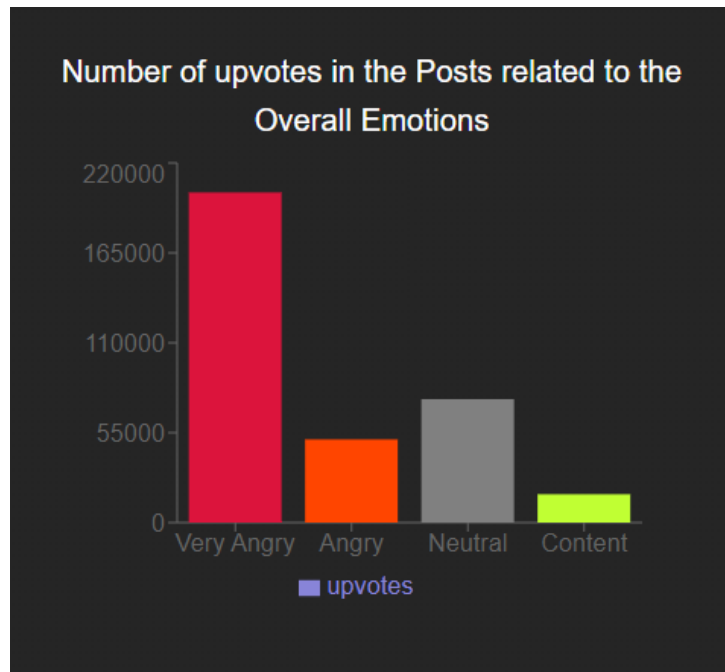


Figura 3.3.10. *BarChart* pentru corelația dintre emoțiile exprimate în postări și numărul de upvotes.

- Numărul de voturi pozitive în corelație cu sentimentul exprimat: Acest grafic cu bare ilustrează numărul de voturi pozitive (upvotes) primite de postări, în funcție de emoțiile exprimate în acestea. Fiecare bară verticală reprezintă o categorie de emoție și indică numărul total de voturi pozitive asociate postărilor care reflectă acea emoție. Graficul arată cum emoțiile exprimate în postări influențează numărul de voturi pozitive primite.

Capitolul 4. Testarea aplicației și rezultate experimentale.

Pentru a demonstra soluția, am ales testarea în principal a componentei de analiză a sentimentelor. În acest sens, vom testa componenta de analiză și cea de vizualizare, cât și pe cea de analiză singular. Astfel, pentru testarea vizualizării vom introduce 2 subiecte de actualitate, unul de la care ne așteptăm la niște rezultate destul de clare și unul care are o oarecare ambiguitate în emoțiile exprimate.

Primul astfel de subiect căutat este *Palestine*. De la această căutare ne așteptăm ca majoritatea sentimentelor exprimate să fie de furie având în vedere situația socio-politică din Palestina.

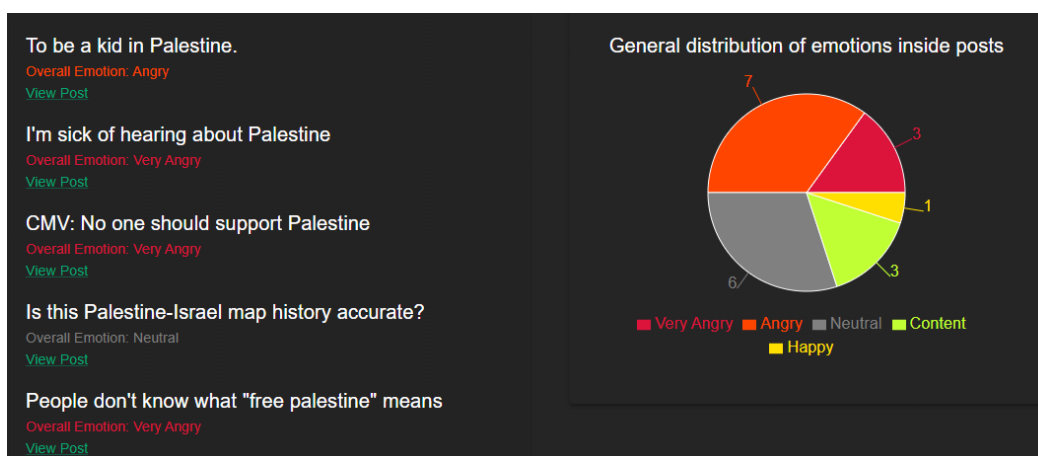


Figura 4.1. Rezultatul căutării subiectului *Palestine*.

Din figura prezentată mai sus, se poate observa prevalența sentimentelor negative legate de subiectul căutat. De asemenea, vom observa din următoarea figură că postările cu scor negativ sunt cele mai apreciate de către utilizatorii platformei.

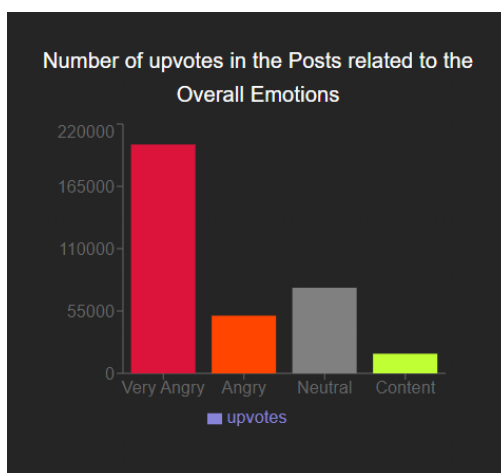


Figura 4.2. Numărul de *upvotes* al utilizatorilor pentru postări cu un anumit sentiment exprimat.

Al doilea subiect pentru care vom efectua o căutare este unul de la care ne așteptăm la rezultate ambigue. Subiectul ales este de asemenea unul actual, de categorie sportivă și anume meciul României împotriva Ucrainei de la Euro 2024. După această căutare ne așteptăm la reacții diverse având în vedere că utilizatorii din țările proveniente își vor susține țara de origine, lucru ce poate crea dispute sau poate duce la discuții constructive.

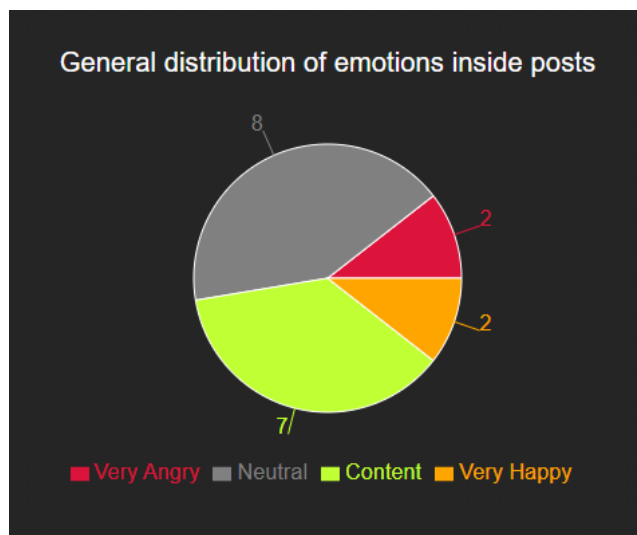


Figura 4.3. Rezultatul căutării pentru subiectul *Romania vs Ukraine*.

Din figura de mai sus se poate observa că sentimentele predominante exprimate în postări sunt neutre sau de intensitate pozitivă redusă, sentimente de mulțumire. Există și postări ce exprimă sentimente de intensitate mare, dar acestea sunt singulare. Prevalența sentimentelor neutre se poate explica prin extragerea unui număr mare de postări de tip articole sportive, în care scriitorul adoptă de obicei un stil neutru în construirea articolului.

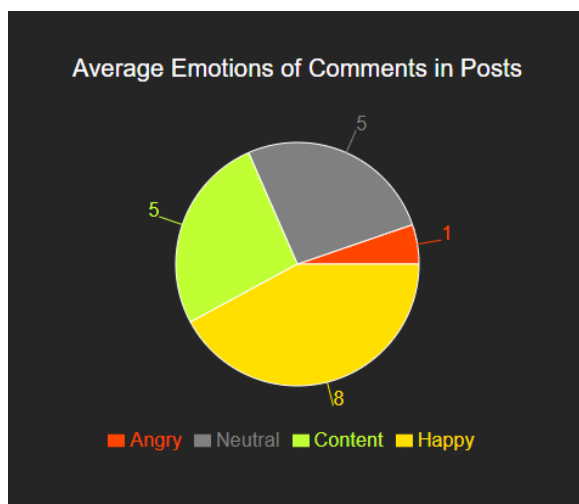


Figura 4.4. Sentimentele exprimate în comentarii pentru *Romania vs Ukraine*.

De asemenea, se observă o discrepanță între sentimentele exprimate în corpul postării și comentariile lor. Astfel, majoritatea utilizatorilor exprimă în comentarii sentimente de bucurie de intensitate mai mare față de cele prezente în cadrul postării propriu-zise.

În plus, pentru a testa analiza emoțiilor, vom introduce opera *Plumb* de George Bacovia pentru demonstrarea capabilității de traducere a textului și testarea extragerii sentimentelor din texte literare.

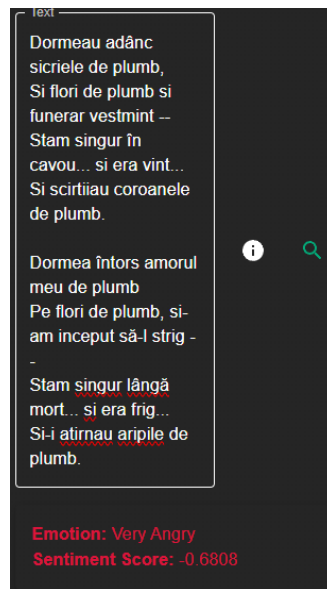


Figura 4.5. Rezultatul introducerii operei *Plumb*.

După cum era de așteptat, rezultatul pentru analiza acestui text literar este unul de o intensitate negativă sporită, având în vedere prezența unei multitudini de cuvinte care exprimă angoasă. La acest rezultat contribuie și semnele de punctuație prin seriile de “...” care sugerează o pauză îndelungată și tensionată.

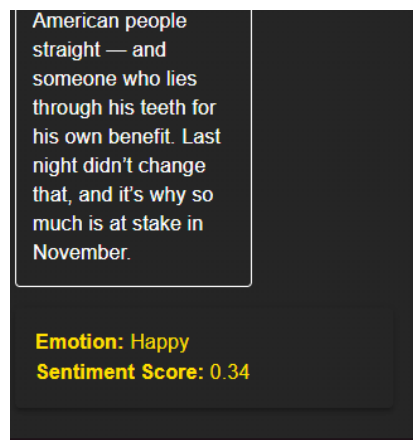


Figura 4.6. Rezultatul introducerii tweet-ului lui Barack Obama

Pentru demonstrarea funcționalității în cadrul analizei textelor colocviale, am introdus un pasaj de pe pagina de Twitter a lui Barack Obama în care vorbește despre un debate dintre Donald Trump și Joe Biden. Acest pasaj este un text reprezentativ pentru limbajul pe care îl adoptă lumea ce relaționează pe rețelele de socializare. Textul conține atât propoziții ce prezintă sentimente negative, cât și sentimente pozitive. În final, algoritmul ajunge la un rezultat aproximativ pozitiv.

Concluzii

În cadrul prezentei lucrări am propus o aplicație care oferă o vizualizare amplă a emoțiilor din cadrul platformelor de socializare. Aplicația este construită astfel încât orice utilizator să poată realiza o căutare despre un anumit subiect și să analizeze sentimentele exprimate de către alte persoane pe rețelele de socializare. Soluția astfel propusă își dorește să rezolve nevoia utilizatorului de a interpreta emoțiile colegilor din mediul online despre anumite subiecte cheie.

În acest sens, lucrarea folosește o abordare bazată pe dicționare, utilizând *VADER* lexicon ca metodă principală de analiză a sentimentelor. Acest lexicon se pliază perfect pe tema abordată, având în vedere faptul că acest dicționar este construit pe baza mesajelor de pe platforma de socializare Twitter, care conține același limbaj colocvial folosit de utilizatori. Un alt motiv pentru alegerea acestei metode este viteza sa în comparație cu metodele bazate pe rețele neuronale.

Subiectele similare au fost abordate în diverse lucrări și proiecte anterioare. Unii cercetători/utilizatori au dezvoltat aplicații web pentru analiza emoțiilor pe platforma Reddit. Cu toate acestea, puține dintre aceste proiecte au integrat o gamă completă de funcționalități pentru extragerea și vizualizarea emoțiilor din postări și comentarii. De exemplu, majoritatea extrag emoțiile sumare (doar coeficienții de pozitivitate și negativitate) și nu creează o vizualizare amplă. De asemenea, aceste abordări anterioare au limite legate de acuratețea și de acoperirea subiectelor și comunităților de pe Reddit.

Inițial, aplicația a fost gândită pentru extragerea conținutului din multiple platforme de socializare. Implementarea de căutare și extragere a informațiilor de pe mai multe platforme de socializare este limitată de costurile materiale ale folosirii API-urilor acestora (ex: platforma Twitter/X care nu oferă un mod de extragere al tweet-urilor în mod gratuit). Astfel, se justifică alegerea platformei Reddit din pricina gratuității și a calității datelor oferite. Soluția propusă poate fi completată cu modelarea datelor de pe mai multe platforme oferindu-ne o perspectivă mai extinsă asupra emoțiilor extrase și existând posibilitatea de a crea comparații între comportamentele umane în funcție de platformă. Altă metodă de îmbunătățire a aplicației ar putea fi reprezentată de utilizarea unor metode mai sofisticate de analiză a sentimentelor, modele pre-antrenate precum BERT sau GPT, deși prin acest mod s-ar sacrifica viteza aplicației.

În concluzie, soluția propusă poate fi utilă în monitorizarea și analiza activității și emoțiilor utilizatorilor pe platforma Reddit. Prin extragerea și analiza datelor din postările și comentariile acestora, aplicația poate oferi utilizatorilor o înțelegere mai profundă a percepțiilor și emoțiilor exprimate în diverse comunități și subiecte de pe Reddit.

Bibliografie

- [1] Esuli, Andrea, and Fabrizio Sebastiani. "SentiWordNet: A publicly available lexical resource for opinion mining." *Proceedings of LREC*. Vol. 6. 2006.
- [2] Hutto, C., și Eric Gilbert. "VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text." *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 8, no. 1, 2014, pp. 216-225, Disponibil la adresa: <https://ojs.aaai.org/index.php/ICWSM/article/view/14550/14399>.
- [3] Pang, Bo, și Lillian Lee. "Opinion Mining and Sentiment Analysis." *Foundations and Trends in Information Retrieval* 2.1-2 (2008): 1-135, Disponibil la adresa: <https://www.cs.cornell.edu/home/llee/omsa/omsa.pdf>.
- [4] Devlin, Jacob, et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." *arXiv preprint arXiv:1810.04805* (2018), Disponibil la adresa: <https://arxiv.org/abs/1810.04805>.
- [5] Bollen, Johan, Huina Mao, și Xiaojun Zeng. "Twitter Mood Predicts the Stock Market." *Journal of Computational Science* 2.1 (2011): 1-8, Disponibil la adresa: <https://www.sciencedirect.com/science/article/abs/pii/S187775031100007X>.
- [6] Thelwall, Mike, et al. "Sentiment Strength Detection in Short Informal Text." *Journal of the American Society for Information Science and Technology* 61.12 (2010): 2544-2558, Disponibil la adresa: <https://onlinelibrary.wiley.com/doi/abs/10.1002/asi.21416>.
- [7] Pang, Bo, Lillian Lee, și Shivakumar Vaithyanathan. "Thumbs up?: Sentiment Classification Using Machine Learning Techniques." *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing-Volume 10* (2002).
- [8] Paul Ekman. An argument for basic emotions. *Cognition & emotion*, 6(3-4):169–200, 1992.
- [9] Seo, Yeong-Seok & Huh, Jun-Ho. (2019). Automatic Emotion-Based Music Classification for Supporting Intelligent IoT Applications. *Electronics*. 8. 164. 10.3390/electronics8020164.
- [10] Spring, *Spring*. Disponibil la adresa: <https://spring.io/projects/spring-framework>.
- [11] Spring Security, *Spring Security*. Disponibil la adresa: <https://spring.io/projects/spring-framework>.
- [12] Recharts, *Recharts*. Disponibil la adresa: <https://recharts.org/en-US/api>.
- [13] NLTK, *NLTK*. Disponibil la adresa: <https://www.nltk.org/api/nltk.sentiment.vader.html>.
- [14] Reddit API Documentation, *Reddit API Documentation*. Disponibil la adresa: https://www.reddit.com/dev/api/#GET_search.
- [15] PRAW, *PRAW*. Disponibil la adresa: <https://praw.readthedocs.io/en/stable/>.
- [16] Michael B. Jones, John Bradley, Nat Sakimura, *RFC 7519*. Disponibil la adresa: <https://datatracker.ietf.org/doc/html/rfc7519>

Anexe

Anexa 1. Implementarea componentei de analiză a sentimentelor

```
translator = GoogleTranslator(source='auto', target='en')

def get_emotion(compound_score):
    if compound_score >= 0.7:
        return "Overjoyed"
    elif compound_score >= 0.48:
        return "Very Happy"
    elif compound_score >= 0.28:
        return "Happy"
    elif compound_score >= 0.08:
        return "Content"
    elif compound_score >= -0.06:
        return "Neutral"
    elif compound_score >= -0.27:
        return "Angry"
    else:
        return "Very Angry"

def remove_non_ascii(text):
    return re.sub(r'^\x00-\x7F+', '', text)

def limit_punctuation(text):
    text = re.sub(r'([!?]{3})([!?]+)', r'\1', text)
    return text

def translate_text(text):
    text = remove_non_ascii(text)
    translated_text = translator.translate(text)
    return translated_text

def detect_language(text):
    if len(text) < 3:
        return False
    try:
        return detect(text) != 'en'
    except Exception as e:
        print("Error occurred during language detection:", e)
        return False
```

```

def get_emotions_from_plain_text(text):
    analyzer = SentimentIntensityAnalyzer()

    if detect_language(text):
        text = translate_text(text)

    sentiment_score =
analyzer.polarity_scores(limit_punctuation(text))['compound']
    emotion = get_emotion(sentiment_score)

    return {
        "text": text,
        "emotion": emotion,
        "sentiment_score": sentiment_score
    }

def analyze_emotions(posts):
    analyzer = SentimentIntensityAnalyzer()
    scores = []

    title_weight = 0.484
    body_weight = 0.304
    comment_weight = 0.471

    for title, data in posts.items():
        if detect_language(title):
            title = translate_text(title)
            translated_comments = []
            for comment in data['comments']:
                if detect_language(comment):
                    comment = translate_text(comment)
                    translated_comments.append(comment)

            title_score =
analyzer.polarity_scores(limit_punctuation(title))['compound']
            comment_scores =
[analyzer.polarity_scores(limit_punctuation(comment))['compound'] for
comment in translated_comments]

            if comment_scores and data['post_body']:
                body_score =
analyzer.polarity_scores(limit_punctuation(data['post_body']))['compou
nd']

                avg_score = title_weight * title_score + body_weight *
body_score + comment_weight * statistics.mean(comment_scores)

                scores.append({"title": title, "url": data['url'],
"emotion": get_emotion(avg_score),
                            "body_score": body_score, "title_score":
title_score,

```

```

        "comment_scores_avg":
statistics.mean(comment_scores),
        "upvotes": data['upvotes'], "avg_score":
avg_score})
    elif comment_scores and not data['post_body']:
        avg_score = title_weight * title_score + comment_weight *
statistics.mean(comment_scores)

        scores.append({"title": title, "url": data['url'],
"emotion": get_emotion(avg_score),
        "body_score": 0, "title_score":
title_score,
        "comment_scores_avg":
statistics.mean(comment_scores),
        "upvotes": data['upvotes'], "avg_score":
avg_score})
    else:
        avg_score = title_score

        scores.append({"title": title, "url": data['url'],
"emotion": get_emotion(avg_score),
        "body_score": 0, "title_score":
title_score,
        "comment_scores_avg": 0,
        "upvotes": data['upvotes'], "avg_score":
avg_score})

    return scores

def get_sentiment_analysis(topic, limit, subreddit):
    posts = reddit_api.fetch_submissions(topic, limit, subreddit)
    return analyze_emotions(posts)

```

Anexa 2. Rutele API-ului de persistență și securitate.

```

@RestController
@RequestMapping("/api/users")
@CrossOrigin
public class UserController {
    private final UserService userService;
    private final HistoryService historyService;
    private final PostService postService;
    @Autowired
    public UserController(UserService userService, HistoryService
historyService, PostService postService) {
        this.userService = userService;
        this.historyService = historyService;
        this.postService = postService;
    }
}

```

```

    }

    @GetMapping
    public ResponseEntity<List<UserModel>> getAllUsers() {

        List<UserModel> userModels = userService.getAllUsersFromDB();
        if(userModels == null){
            return new ResponseEntity<>(null, HttpStatus.NOT_FOUND);
        }
        else return new ResponseEntity<>(userModels, HttpStatus.OK);

    }

    @GetMapping("/{id}")
    public ResponseEntity<UserRes> getUserById(@PathVariable String
id,

    @RequestHeader(HttpHeaders.AUTHORIZATION) String token){
        return userService.getUserByIdFromDB(id, token);
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<UserModel> deleteUserById(@PathVariable
String id){
        return userService.deleteUserByIdFromDB(id);
    }

    @PutMapping("/{id}")
    public ResponseEntity<UserModel> editUserById(@PathVariable String
id,

    @RequestBody UserRes
userRes,

    @RequestHeader(HttpHeaders.AUTHORIZATION) String token){
        return userService.editUserInfo(id, userRes, token);
    }

    @GetMapping("/{id}/history")
    public ResponseEntity<List<HistoryModel>>
getAllUsersHistory(@PathVariable String id,

    @RequestHeader(HttpHeaders.AUTHORIZATION) String token){
        return historyService.getAllHistoryForUser(id, token);
    }

    @GetMapping("/{id}/history/{historyId}")
    public ResponseEntity<HistoryModel>
getUsersHistoryById(@PathVariable String id,

    @PathVariable String historyId,

```



```

@RequestHeader(HttpHeaders.AUTHORIZATION) String token){
    return
    historyService.getHistoryForUserById(id, historyId, token);
}

@PostMapping("/{id}/history")
public ResponseEntity<HistoryModel> searchTopic(@PathVariable
String id,
                                                @RequestParam
String topic,

@RequestParam(required = false) Integer limit,

@RequestParam(required = false) String subreddit) throws
JsonProcessingException {
    return historyService.searchTopic(id, topic, limit, subreddit);
}

@DeleteMapping("/{id}/history/{historyId}")
public ResponseEntity<HistoryModel> deleteHistory(@PathVariable
String id,
                                                @PathVariable
String historyId,

@RequestHeader(HttpHeaders.AUTHORIZATION) String token){
    return historyService.deleteHistoryById(id, historyId, token);
}

@GetMapping("/{id}/history/{historyId}/post")
public ResponseEntity<List<PostModel>>
getAllPostsFromHistory(@PathVariable String id,

@PathVariable String historyId,

@RequestHeader(HttpHeaders.AUTHORIZATION) String token){
    return postService.getAllPostsForHistory(id, historyId, token);
}

@GetMapping("/{id}/history/{historyId}/post/{postId}")
public ResponseEntity<PostModel>
getPostFromHistoryById(@PathVariable String id,

@PathVariable String historyId,

@PathVariable String postId,

@RequestHeader(HttpHeaders.AUTHORIZATION) String token){
    return
    postService.getPostForHistoryById(id, historyId, postId, token);
}

```

```

        @PostMapping("/{id}/text")
        public ResponseEntity<TextModel>
getEmotionFromPlainText(@PathVariable String id,

@RequestParam String text,

@RequestParam(HttpHeaders.AUTHORIZATION) String token) throws
JsonProcessingException {
    TextModel textModel =
historyService.getEmotionsFromText(text);

    if(textModel == null){
        return new ResponseEntity<>(null, HttpStatus.NOT_FOUND);
    }
    return new ResponseEntity<>(textModel, HttpStatus.OK);
}

        @DeleteMapping("/{id}/history/{historyId}/post/{postId}")
        public ResponseEntity<PostModel>
deletePostFromHistoryById(@PathVariable String id,

@PathVariable String historyId,

@PathVariable String postId,

@RequestParam(HttpHeaders.AUTHORIZATION) String token){
    return
postService.deletePostFromHistoryById(id,historyId,postId,token);
}
}

```

Anexa 3. Implementarea componentei de vizualizare.

```

const EmotionPieChart = ({ posts }) => {
    const colorMap = {
        "Very Angry": '#dc143c',    // Crimson
        Angry: '#ff4500',           // OrangeRed
        Neutral: '#808080',         // Grey
        Content: '#c0ff33',         // Light Green
        Happy: '#ffdf00',           // Yellow
        "Very Happy": '#ffa500',    // Dark Orange
        Overjoyed: '#ffb347'        // Orange
    };
};

```

```

const emotionOrder = {
  "Very Angry": 1,
  Angry: 2,
  Neutral: 3,
  Content: 4,
  Happy: 5,
  "Very Happy": 6,
  Overjoyed: 7
};

const processData = () => {
  const counts = posts.reduce((acc, post) => {
    acc[post.emotion] = (acc[post.emotion] || 0) + 1;
    return acc;
  }, {});

  const data = Object.keys(counts).map(key => ({
    name: key,
    value: counts[key],
    color: colorMap[key] || '#000000'
  }));

  return data.sort((a, b) => emotionOrder[a.name] -
emotionOrder[b.name]);
};

const data = processData();

return (
  <Container>
    <Card elevation={3} sx={{ p: 2, margin: 'auto', maxWidth: 600,
flexGrow: 1 }}>
      <CardContent>
        <Typography variant="h6" component="div" gutterBottom
align="center">

```

```

        General distribution of emotions inside posts
      </Typography>
      <ResponsiveContainer width="100%" height={300}>
        <PieChart>
          <Pie data={data} dataKey="value" nameKey="name" cx="50%"
cy="50%" outerRadius={100} fill="#8884d8" label>
            {data.map((entry, index) => (
              <Cell key={`cell-${index}`} fill={entry.color} />
            ))}
          </Pie>
          <Tooltip />
          <Legend />
        </PieChart>
      </ResponsiveContainer>
    </CardContent>
  </Card>
</Container>
);
};

export default EmotionPieChart;

const UpvotesEmotionBarChart = ({ posts }) => {
  const colorMap = {
    "Very Angry": '#dc143c', // Crimson
    Angry: '#ff4500', // OrangeRed
    Neutral: '#808080', // Grey
    Content: '#c0ff33', // Light Green
    Happy: '#ffdf00', // Yellow
    "Very Happy": '#ffa500', // Dark Orange
    Overjoyed: '#ffb347' // Orange
  };
};

const emotionOrder = {
  "Very Angry": 1,

```

```

    Angry: 2,
    Neutral: 3,
    Content: 4,
    Happy: 5,
    "Very Happy": 6,
    Overjoyed: 7
  };

  const processData = () => {
    const counts = posts.reduce((acc, post) => {
      if (!acc[post.emotion]) {
        acc[post.emotion] = { emotion: post.emotion, upvotes: 0 };
      }
      acc[post.emotion].upvotes += post.upvotes;
      return acc;
    }, {});

    const data = Object.values(counts);

    return data.sort((a, b) => emotionOrder[a.emotion] -
emotionOrder[b.emotion]);
  };

  const data = processData();

  return (
    <Container>
      <Card elevation={3} sx={{ p: 2, margin: 'auto', maxWidth: 600,
flexGrow: 1 }}>
        <CardContent>
          <Typography variant="h6" component="div" gutterBottom
align="center">
            Number of upvotes in the Posts related to the Overall
Emotions
          </Typography>

```

```

    <ResponsiveContainer width="100%" height={300}>
      <BarChart data={data} margin={{ top: 5, right: 30, left:
20, bottom: 5 }}>
        <XAxis dataKey="emotion" />
        <YAxis />
        <Tooltip />
        <Legend />
        <Bar dataKey="upvotes" fill="#8884d8">
          {data.map((entry, index) => (
            <Cell key={`cell-${index}`}
fill={colorMap[entry.emotion] || '#000000'} />
          ))}
        </Bar>
      </BarChart>
    </ResponsiveContainer>
  </CardContent>
</Card>
</Container>
);
};

export default UpvotesEmotionBarChart;

```