



FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

**Aplicație pentru analiza textelor financiare utilizând
modele de Procesare de Limbaj Natural**

LUCRARE DE LICENȚĂ

Absolvent: **Mădălina-Ionela STROE**

Coordonator **Conf.Dr.Ing. Anca MĂRGINEAN**
științific:

2022



FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

DECAN,
Prof. dr. ing. Liviu MICLEA

DIRECTOR DEPARTAMENT,
Prof. dr. ing. Rodica POTOLEA

Absolvent: **Mădălina-Ionela STROE**

Aplicație pentru analiza textelor financiare utilizând modele de Procesare de Limbaj Natural

1. **Enunțul temei:** *Aplicație pentru analiza textelor financiare, care oferă o suită de servicii prin care utilizatorii pot să analizeze și să sintetizeze articole de știri. Utilizatorii își pot crea conturi în care își pot salva sumarul analizelor anterioare.*
2. **Conținutul lucrării:** *Pagina de prezentare, Introducere, Obiectivele proiectului, Studiu Bibliografic, Analiză și Fundamentare Teoretică, Proiectare de Detaliu și Implementare, Testare și Validare, Manual de Instalare și Utilizare, Concluzii, Bibliografie*
3. **Locul documentării:** Universitatea Tehnică din Cluj-Napoca, Departamentul Calculatoare
4. **Consultanți:**
5. **Data emiterii temei:** 1 Noiembrie 2021
6. **Data predării:** 2 Septembrie 2022

Absolvent: Mădălina-Ionela Stroe

Coordonator științific: Conf.Dr.Ing. Anca Mărginean



FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

**Declarație pe propria răspundere privind
autenticitatea lucrării de licență**

Subsemnatul(a) Stroe Mădălina-Ionela legitimat(ă) cu CI seria MH nr. 666345 CNP 2991231250809, autorul lucrării Aplicație pentru analiza textelor financiare utilizând modele de Procesare de Limbaj Natural elaborată în vederea susținerii examenului de finalizare a studiilor de licență la Facultatea de Automatică și Calculatoare, Specializarea Tehnologia Informatiei din cadrul Universității Tehnice din Cluj-Napoca, sesiunea septembrie 2022 a anului universitar 2021-2022, declar pe propria răspundere că această lucrare este rezultatul propriei activități intelectuale, pe baza cercetărilor mele și pe baza informațiilor obținute din surse care au fost citate, în textul lucrării și în bibliografie.

Declar că această lucrare nu conține porțiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislației române și a convențiilor internaționale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în fața unei alte comisii de examen de licență.

În cazul constatării ulterioare a unor declarații false, voi suporta sancțiunile administrative, respectiv, *anularea examenului de licență*.

Data

Nume, Prenume

Stroe Mădălina-Ionela

Semnătura

De citit înainte (această pagină se va elimina din versiunea finală):

1. Cele trei pagini anterioare (foaie de capăt, foaie sumar, declarație) se vor lista pe foi separate (nu față-verso), fiind incluse în lucrarea listată. Foaia de sumar (a doua) necesită semnătura absolventului, respectiv a coordonatorului. Pe declarație se trece data când se predă lucrarea la secretarii de comisie.
2. Pe foaia de capăt, se va trece corect titulatura cadrului didactic îndrumător (consultați pagina de unde ați descărcat acest document pentru lista cadrelor didactice cu titulaturile lor).
3. Documentul curent **nu** a fost creat MS Office. E posibil să fie mici diferențe de formatare.
4. Cuprinsul începe pe pagină nouă, impară (dacă se face listare față-verso), prima pagină din capitolul Introducere tot așa, fiind numerotată cu 1.
5. Vizualizați (recomandabil și în timpul editării) acest document.
6. Fiecare capitol începe pe pagină nouă.
7. Folosiți stilurile predefinite (Headings, Figure, Table, Normal, etc.)
8. Marginile la pagini nu se modifică.
9. Respectați restul instrucțiunilor din fiecare capitol.

Cuprins

Capitolul 1 Introducere - Contextul proiectului	1
1.1 Contextul proiectului	1
1.1.1 Context general	1
1.1.2 Conturarea domeniului exact al temei	2
1.2 Structura lucrării	3
Capitolul 2 Obiectivele Proiectului	4
2.1 Obiectul principal	4
2.2 Obiective secundare	4
Capitolul 3 Studiu Bibliografic	6
3.1 Modul backend	6
3.1.1 Arhitectura Client-Server	6
3.1.2 Arhitectura 3-Tier	7
3.1.3 Mediator Design Pattern	8
3.1.4 CQRS	9
3.1.5 Fluent Validation	9
3.2 Modul frontend	10
3.2.1 React	10
3.2.2 Ant Design	10
3.2.3 Styled Components	11
3.2.4 i18n	11
3.3 Modele de Procesare a limbajului natural	11
3.3.1 FinBERT	11
3.3.2 Sumarizator de text	13
3.3.3 Keyphrase Extractor	14
3.4 Aplicații similare	14
3.4.1 Serviciul de înțelegere a limbajului natural dezvoltat de IBM	14
3.4.2 Google Cloud Natural Language	15
3.4.3 Microsoft AI	16
3.4.4 Google Trends	16
Capitolul 4 Analiză și Fundamentare Teoretică	18
4.1 Cerințe funcționale și non-funcționale	18
4.1.1 Cerințe funcționale	18
4.1.2 Cerințe pentru funcții de analiză a unui text	19
4.1.3 Cerințe non-funcționale	19
4.2 Cazuri de utilizare	19
4.2.1 Utilizator neautentificat	19
4.2.2 Utilizator autentificat	21
4.3 Algoritmi utilizați	26
4.3.1 Algoritmul pentru autorizarea accesului utilizatorului la anumite resurse	26
4.3.2 Algoritmul pentru crearea unei parole puternice	26
4.4 Protocole utilizate	27

4.4.1	Protocolul HTTP	27
4.4.2	Protocolul TCP/IP	27
4.4.3	Protocolul SMTP	28
Capitolul 5	Proiectare de Detaliu și Implementare	29
5.1	Structura aplicației	29
5.2	Structura bazei de date	31
5.3	Structura modulului de backend	33
5.4	Structura modulului de frontend	42
5.5	Analizarea textului	46
Capitolul 6	Testare și Validare	49
6.1	Testare backend	49
6.1.1	Testare login cu credențiale greșite	49
6.1.2	Testare login cu credențiale corecte	49
6.1.3	Testare editare detalii utilizator când un câmp este gol	50
6.2	Testare frontend	51
6.3	Testare analiză a textului	51
Capitolul 7	Manual de Instalare și Utilizare	53
7.1	Instalare	53
7.1.1	Condiții prealabile generale	53
7.1.2	Condiții prealabile pentru baza de date	53
7.1.3	Condiții prealabile pentru modulul de backend	53
7.1.4	Condiții prealabile pentru modulul de frontend	54
7.2	Rulare	54
7.2.1	Setup baza de date	54
7.2.2	Rulare modul backend	54
7.2.3	Rulare modul frontend	55
7.3	Manual de utilizare	56
Capitolul 8	Concluzii	65
8.1	Analiza rezultatelor obținute	65
8.2	Dezvoltări și îmbunătățiri ulterioare	66
Bibliografie		67

Capitolul 1. Introducere - Contextul proiectului

1.1. Contextul proiectului

1.1.1. Context general

Pornim de la acest concept: *Finance* (*în română, finanțe*). O primă căutare în Google Trends ne oferă o perspectivă asupra evoluției acestui subiect sau domeniul din ultimii ani. Mai exact, începând cu anul 2004, Google Trends oferă informații referitoare la numărul căutarilor ce conțin acest cuvânt cheie.

Putem vedea că există 2 perioade de apogeu: Octombrie 2008 și Martie 2020. În ambele perioade, au existat evenimente care au avut atât consecințe imediate, dar și de lungă durată. Luând ca exemplu anul 2020, încadrăm la consecințe imediate pierderea locurilor de muncă a persoanelor. Desigur, acest lucru fiind unul temporar și existând soluții, oamenii au început să se orienteze spre alte locuri de muncă sau metode de a obține un venit pasiv. Investițiile în criptomonede au fost cele mai populare. Libertatea oferită de aceste tipuri de investiții, alături de şansele unui profit mare, într-un context favorabil dezvoltării unor noi hobby-uri, a permis unui grup extins de oameni să încerce să facă lucruri pe care, în trecut, doar oamenii care aveau studii în domeniu le puteau înțelege sau realiza.

Totuși, libertatea de a alege în ce să investești dintr-o listă vastă de opțiuni putea aduce cu sine un dezavantaj - o decizie neinformată putea să afecteze negativ persoana în cauză. Partea bună este că, fiind un subiect atât de actual și popular, prezent aproape peste tot în rețelele de socializare, reclame, este publicat aproape zilnic un articol legat de aceasta. Așadar, acest lucru a permis și oamenilor care nu au avut partea de finanțe ca ocupație principală, să își extindă interesul și în această zonă.

Tot în această perioadă, pe lângă grupurile de investitori, s-au dezvoltat și aplicații care să ajute persoanele să realizeze tranzacții, să poată observa fluctuația criptomonedelor și acțiunilor, să compună un istoric cu informațiile necesare pentru ca o persoană să ia decizii informate.

Totuși, un istoric al prețurilor, de multe ori nu explică motivul fluctuației. Valoarea depinde în acest context, de multe ori, de știri și articole. Un articol perceptuat ca fiind rău de un investitor, va influența prețul și tranzacțiile viitoare, dar nu e întotdeauna o regulă.

Să luăm următoarele exemple:

- CEO-ul de la renomata companie X s-a retras din activitate, dar urmează să fie înlocuit de Y, care a realizat proiectul Z ce a avut mare succes.
 - Faptul că CEO-ul s-a retras din activitate, ar putea fi perceptuat ca fiind rău, putând duce la o scădere a prețurilor.
 - Faptul că CEO-ul urmează să fie înlocuit de o persoană care a avut succes în domeniu poate fi perceptuat ca un lucru bun, pentru viitor.

- Analizând punctele de mai sus, şansele de a fi o scădere a preţurilor, urmată de o creştere în viitor sunt destul de mari.
- O companie urmează să se extindă, deschizând un sediu într-o altă regiune.
 - Ştirea despre extindere poate fi percepătă ca pozitivă, deci ar putea duce la creşterea preţurilor.
- O companie a lansat un produs care a fost sub aşteptările clienţilor, deşi, în general compania are produse bune.
 - Ştirea despre produsul cu defecte poate fi percepătă ca fiind negativă.
 - În considerare faptul că până acum compania a avut produse ce au satisfăcut clienţii, şansele sunt mari ca următorul produs lansat după feedback-ul produsului actual, să fie din nou mulţumitor pentru piaţă.
 - Analizând punctele de mai sus, există o şansă ca preţurile să scadă acum şi să crească în următoarea perioadă, dar există un risc.

1.1.2. Conturarea domeniului exact al temei

Domeniul investiţiilor atât de variabile necesită documentare continuă şi atenţie ridicată pentru că sunt mulţi factori care pot influenţa, în câteva secunde, o decizie. Multe dintre aplicaţiile create în ultima vreme, s-au concentrat pe a uşura procedurile de tranzacţionare pentru utilizatori.

Uşurinţa de a face aceste lucruri, la care se adaugă potenţialul căştig, a făcut utilizatorii să urmărească momentul potrivit. Aici putem găsi 2 grupuri: cei care iau deciziile prin precizie matematică şi cei care iau deciziile prin studiul pieţei. Persoanele care înțeleg evoluţia pe grafuri se ghidăză după algoritmi. Celălalt grup înțelege evoluţia prin impactul pe care îl au materialele scrise, anume ştiri, articole, etc. Dezavantajul major al accesului la atâta materiale scrise constă în timpul petrecut pentru a-l citi pe fiecare dintre ele, a extrage ideile principale şi a-ţi forma o decizie financiară informată.

Există, desigur, companii mai mari care au diferite instrumente interne pentru a obține o posibilă analiză a sentimentului pieţei. Algoritmii de analiză au o acurateţe ridicată, dar disponibilitatea lor limitată îi face destul de inaccesibil pentru persoanele obişnuite.

Aşadar, tema proiectului propune o aplicaţie accesibilă care obține o analiză detaliată a articolelor financiare prin:

- Alegera textului de către utilizator
- Analiza de sentimente financiare
- Extragerea cuvintelor sau frazelor cheie
- Sumarizarea textului

Integrând într-o aplicaţie funcţionalităţile de mai sus, creăm un instrument pentru investitori, fie ei specializaţi în domeniul sau nu, care oferă o suiată de servicii prin care utilizatorii pot să analizeze şi să sintetizeze articole de ştiri ce oferă informaţii financiare. Mai mult decât atât, utilizatorii îşi pot crea un cont unde îşi pot salva analizele anterioare pentru a o analiză retrospectivă, dar şi pentru a observa topicurile populare.

1.2. Structura lucrării

Capitolul 1 - Introducere - Acest capitol prezintă contextul general și contextul exact al aplicației.

Capitolul 2 - Obiectivele proiectului - Acest capitol cuprinde obiectivul principal al proiectului și obiectivele secundare care au dus la realizarea obiectivului principal.

Capitolul 3 - Studiu bibliografic

Capitolul 4 - Analiză și Fundamentare teoretică - În acest capitol sunt descrise principiile funcționale ale aplicației, algoritmii și protocolele utilizate, explicații ale soluției alese, structura logică și funcțională a aplicației.

Capitolul 5 - Proiectare de detaliu și Implementare - În acest capitol sunt prezentate arhitectura, diagramele importante și descrierile componentelor la nivel de modul.

Capitolul 6 - Testare, Validare și Evaluare - În acest capitol sunt prezentate metodele de testare și validare ale aplicației.

Capitolul 7 - Manual de Instalare și Utilizare - În acest capitol sunt descrișii pașii de instalare ai proiectului și modul de utilizare a aplicației.

Capitolul 8 - Concluzii- În acest capitol sunt prezentate rezultatele obținute și descrise posibilele dezvoltări ulterioare.

Capitolul 2. Obiectivele Proiectului

2.1. Obiectul principal

Obiectivul principal al acestui proiect este de a crea un instrument de analiză amănunțită a unui text finanțiar. În acest domeniu, cu atenția orientată mai ales spre investiții, oamenii urmăresc continuu un moment potrivit și sunt predispuși în a lua decizii rapide care îi pot sau nu afecta în viitor. Deciziile pot fi influențate de mai mulți factori: informațiile transmise prin știri sau articole, cunoștințele precare, etc. Cu ajutorul acestei aplicații, utilizatorii pot înțelege mai rapid ce este transmis într-un articol voluminos pentru a putea să își formeze o părere informată înainte de a lua o decizie. Dat fiind faptul că datele de intrare care vor fi folosite pentru a fi analizate în această aplicație sunt majoritatea știri, în aplicație vor fi integrate module care se vor ocupa de înțelegerea și procesarea limbajului natural.

2.2. Obiective secundare

Unul dintre cele mai importante obiective secundare este modul în care este prezentată aplicația utilizatorului final. Statisticile arată că utilizatorii își doresc, în proporție de 49%, un design minimalist, intuitiv, cu informațiile principale în secțiuni cât mai ușor accesibile. Utilizatorul ar trebui să aibă într-un singur loc toate informațiile rezultate în urma analizei unui text finanțiar, pentru a putea face ușor conexiuni sau comparații. Pe lângă acestea, utilizatorul poate vedea evoluția pe zile a aparițiilor comune cuvintelor cheie identificate în textul său.

Al doilea obiectiv secundar este reprezentat de feedback-ul pe care îl oferă aplicația utilizatorului, prin indicațiile pentru ce trebuie introdus în anumite câmpuri, notificările primite ca rezultat al anumitor requesturi ale utilizatorului, email de confirmare după crearea unui cont nou, etc.

Pentru corectitudinea datelor, dar și pentru protecția lor, utilizatorul primește feedback vizual atunci când datele introduse nu au formatul corect sau numărul de caractere nu este în limita acceptată.

De asemenea, utilizatorul este notificat, în cazul introducerii sau modificării datelor personale precum numele de utilizator sau email dacă sunt deja utilizate.

Al treilea obiectiv secundar este securitatea aplicației, care va fi asigurată în următoarele moduri:

- La crearea unui cont nou, parola salvată în baza de date va fi trecută printr-o funcție de hashing SHA512, generând un output unic de 512 biți.
- După crearea cu succes a contului, la logare, utilizatorul va avea un token de acces pentru a putea accesa anumite resurse
- Accesul permis doar în anumite pagini ale aplicației, folosind rutele protejate

Al patrulea obiectiv secundar este reprezentat de abilitatea utilizatorului de modificare a anumitor date din cont, fie ele personale, precum datele contului, strict legate de analizele pe text, adică ce texte analizate dorește să pastreze în istoricul personal. Pentru datele persoanale, singurele restricții sunt impuse pentru datele care identifică un utilizator: username-ul și email-ul; acestea trebuie să fie unice atunci când există o încercare de actualizare a datelor, altfel utilizatorul va fi notificat cu un mesaj specific erorii care a apărut în procesul de salvare a datelor. Pentru partea de salvare în istoricul personal al analizelor efectuate, utilizator poate să aleagă dacă salvează o analiză în istoric, altfel rezultatele sunt decartate.

Al cincilea obiectiv secundar este realizat prin integrarea modelelor de procesare de limbaj natural pentru a fi folosite cu următoarele scopuri:

- Analiza textului prin extragerea sentimentelor financiare
 - După procesarea textului finanțiar, rezultatul analizei constă într-un scor procentual obținut pentru sentimentele negativ, neutru și pozitiv
- Sumarizarea textului
 - Pentru că nu întotdeauna un sentiment pozitiv înseamnă același lucru și în realitate, pentru mai multă claritate utilizatorul va avea extrase ideile principale din textul analizat
- Extragerea de cuvinte cheie
 - Rezultatul acestei analize a textului constă în cuvintele cheie identificate în text ce vor fi folosite ulterior pentru generarea graficelor de popularitate, la cerința utilizatorului și afișarea definițiilor

Alt săselea obiectiv secundar este de a oferi utilizatorului posibilitatea de a vedea un trending intern, constituit din top 10 cele mai populare cuvinte cheie sau fraze extrase din textele analizate într-o anumită perioadă de timp selectată de el, dar și evoluția anumitor cuvinte cheie introduse de utilizator. Toate aceste informații vor fi afișate într-un mod cât mai ușor de înțeles de utilizator, conținând informațiile relevante. Pentru cuvintele cheie extrase, se vor afișa definițiile acestora.

Pe lângă toate acestea, aplicația trebuie să fie extensibilă, pentru a putea adăuga funcționalități ulterioare. Proiectarea acesteia trebuie să fie în aşa fel încât modulele sau componentele actuale să nu fie afectate de adăugarea de funcționalitate nouă.

Capitolul 3. Studiu Bibliografic

3.1. Modul backend

3.1.1. Arhitectura Client-Server

Acest tip de arhitectură este folosită pentru a separa sarcinile între client și server. Folosind un mecanism de tip request-reponse, procesul este următorul:

- Clientul face un request la server pentru a obține o anumită resursă
- Serverul trimite un răspuns per request clientului
- Conexiunea poate fi închisă de oricare dintre cei doi

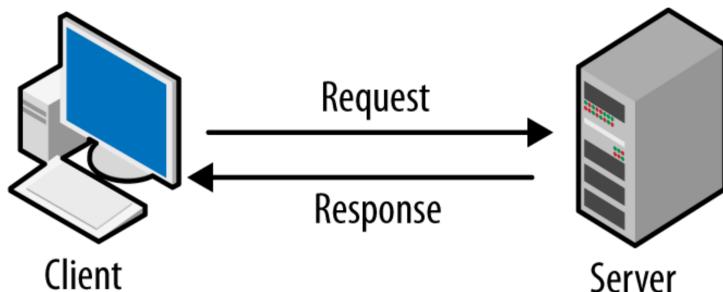


Figura 3.1: Arhitectura 3-tier. Sursa [1]

Protocolul HTTP este un protocol request-reponse, care se bazează pe o arhitectură client-server.

Un *HTTP request* reprezintă modalitatea prin care browserul web, în cazul acestui proiect, solicită resurse. Fiecare request conține:

- Versiunea HTTP
- URL (Uniform Resource Locator)
 - Informații despre locația resursei solicitată de client
- Metoda HTTP
 - GET
 - HEAD
 - POST
 - PUT
 - DELETE
 - TRACE

- OPTIONS
- CONNECT
- PATCH
- HTTP Request Header
 - Conține informații despre browser-ul clientului, resursa cerută, encoding, server, etc.
- Body (optional, doar pentru anumite requesturi)
 - Conține informații transmise serverului (pentru anumite metode HTTP)

Un *HTTP response* este informația pe care o primesc clienții de la server ca răspuns la request. Acest răspuns conține:

- Status
 - Cod din 3 cifre care indică dacă requestul a fost încheiat cu succes
- HTTP Response Header
 - Conține informații despre limba și formatul conținutului din body
- Body
 - Conține informația solicitată printr-un request

3.1.2. Arhitectura 3-Tier

Arhitectura 3-Tier sau 3-Layer este o arhitectură pe 3 nivele:

- Nivelul prezentării
 - Acest nivel este reprezentat de modulul de front-end al aplicație, interfață cu care utilizatorii vor interacționa în mod direct
 - Acest nivel a fost construit în aplicație folosind:
 - * În librăria *React Typescript*
 - * Pentru componente, este utilizată libraria *Ant Design*
 - * Pentru customizarea componentelor s-a utilizat *Styled Components*, un framework pentru stilizare, CSS-in-JS
- Nivelul aplicației
 - Acest nivel este reprezentat de modulul de back-end al aplicației
 - Aici sunt procesate informațiile primite de la nivelul superior, cel de prezentare
 - Nivelul interacționează în mod direct cu serverul. Aceasta transmite cererea clientului la server, la fel și răspunsul serverului către client
 - Pentru implementarea acestui nivel în aplicație, am folosit:
 - * .NET Framework
 - * Entity Framework Core, un Object-Relational Mapper, care creează un layer între limbaj și baza de date
 - * MediatR, implementarea în .NET a patternului
- Nivelul datelor
 - Cunoscut ca nivelul bazei de date

- Aici se găsesc servere de baze de date unde se pot stoca și prelua informații
- Pentru acest nivel, am folosit Microsoft SQL Server

Avantajul utilizării acestui pattern arhitectural este că permite ca un layer să fie modificat, fără să fie afectate celelalte.

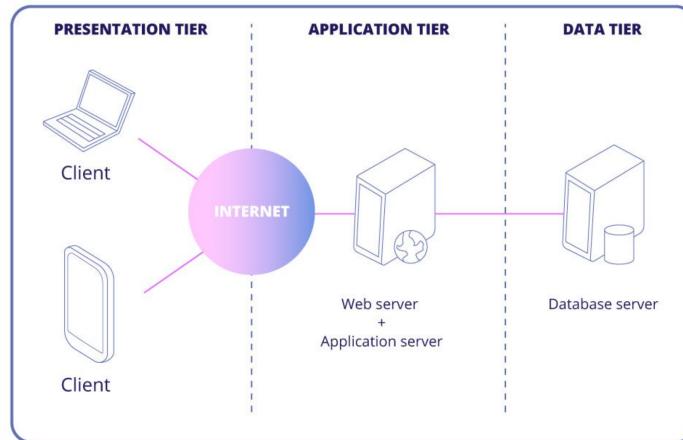


Figura 3.2: Arhitectura 3-tier. Sursa [2]

3.1.3. Mediator Design Pattern

Scopul utilizării acestui design pattern este de a reduce dependințele dintre obiecte, restricționând comunicarea între obiecte și mijlocind colaborarea printr-un mediator.

Avantajele utilizării mediatorului sunt:

- Comunicarea între mai multe componente poate fi extrasă într-un singur loc, fiind mai ușor de înțeles și modificat, acolo unde e cazul (Single Responsibility Principle)
- Se pot introduce noi medieri fără a schimba componentele deja existente
- Cuplarea între componente este redusă (*Low coupling*)

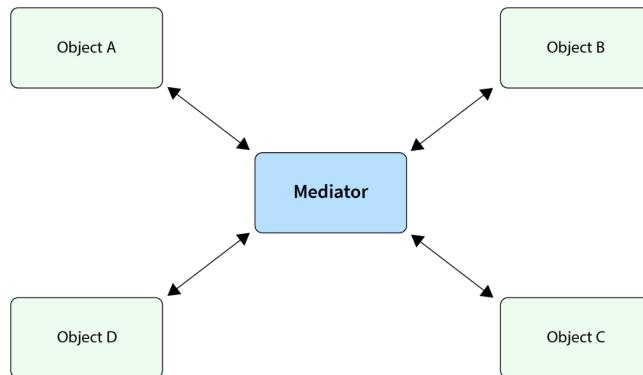


Figura 3.3: Diagramă Mediador Design Pattern. Sursa [3]

3.1.4. CQRS

Command-Query Responsibility Segregation sau CQRS este un design pattern care separă comenzi de interogări.

Comenzi sunt cele care scriu sau actualizează date în baza de date, iar o interogare citește datele din baza de date.

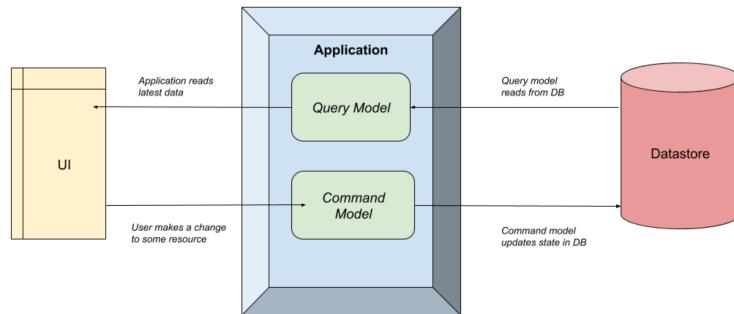


Figura 3.4: Diagramă CQRS. Sursa [4]

3.1.5. Fluent Validation

Fluent Validation este o librerie din .NET pentru validarea modelelor.

Avantajul folosirii acestei librarii constă în faptul că logica de validare poate fi separată de modele și se renunță la adnotări.

În exemplul din figura 3.6, câmpul de firstName a fost completat cu un string gol, deși în validator modelul trebuie să aibă acest câmp este marcat ca fiind *required*. De asemenea, a fost adăugată o validare extra pentru numărul de caractere și, fiind încălcată, este menționată în răspunsul din Swagger.

```
public class UserConfiguration : IEntityTypeConfiguration<User>
{
    public void Configure(EntityTypeBuilder<User> builder)
    {
        builder.HasKey(b => b.Id);
        builder.Property(b => b.FirstName).IsRequired();
        builder.Property(b => b.LastName).IsRequired();
        builder.Property(b => b.Username).IsRequired();
        builder.Property(b => b.Email).IsRequired();
        builder.Property(b => b.PasswordHash).IsRequired();
        builder.Property(b => b.PasswordSalt).IsRequired();
    }
}
```

Figura 3.5: Configurare validări pentru modelul unui utilizator

```

Request body
{
  "firstName": "",
  "lastName": "testLastName",
  "username": "testUsername",
  "email": "email@email.com",
  "password": "passwordTest"
}

Server response
Code: 400
Details: Error: response status is 400
Response body
{
  "type": "https://tools.ietf.org/html/rfc7231#section-6.5.1",
  "title": "One or more validation errors occurred.",
  "status": 400,
  "traceId": "00-e8274a44af3df787f72c644ec3774a26-7d676f3e44ae8a4f-00",
  "errors": [
    "FirstName": [
      "First name can not be empty!",
      "First name should have between 2 and 100 alpha characters!"
    ]
  ]
}

```

Figura 3.6: Request - Response folosind Fluent Validation

3.2. Modul frontend

3.2.1. React

Este un framework pentru dezvoltarea interfeței utilizator. Cu ajutorul acesteia, codul scris și componentele devin reutilizabile.

A fost dezvoltat de Facebook în 2011 și este cel mai popular framework din acest moment. Avantajele folosirii React sunt:

- Ușor de învățat și folosit
- Componentele sunt reutilizabile
- Oferă flexibilitate
- SEO friendly

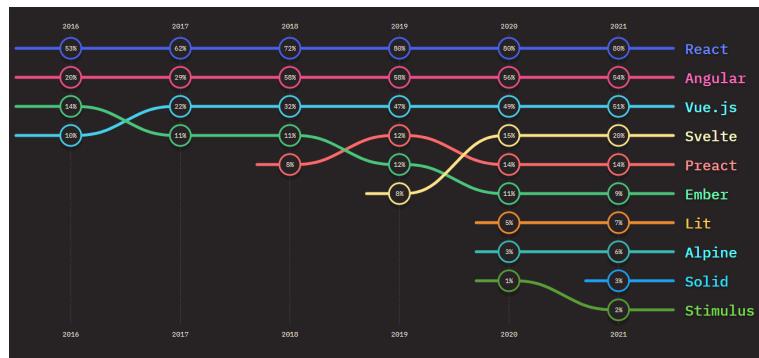


Figura 3.7: Cele mai populare frameworkuri pentru UI. Sursa [5]

3.2.2. Ant Design

Este o librărie din React, care conține componente ușor de utilizat. Componentele pot fi customizate cu ajutorul designului oferit de librarie, dar și din exterior, cu ajutorul CSS. Avantajele folosirii acestei librării sunt:

- Design-ul consistent și accesibil
- Suport pentru Typescript și Javascript
- Form-uri ușor de customizat
- Variatii pentru fiecare componentă

3.2.3. Styled Components

Folosind CSS-in-JS, cu ajutorul libreriei styled-components, componentele pot fi ajustate aplicând direct un stil customizat. Avantajele folosirii sunt:

- Reutilizarea - precum componente din React, pentru a evita duplicarea, acesta poate fi reutilizabil atunci când se aplică pe componente
- CSS
- Generează nume unice pentru fiecare clasa pentru targetarea corectă a componentelor

```

8 const CustomMainContainer = styled.div`  

9   height: 100%;  

10  width: 100%;  

11  display: flex;  

12  align-items: center;  

13  justify-content: center;  

14`;

```

Figura 3.8: Exemplu utilizare styled-components

3.2.4. i18n

Libraria i18next este folosită pentru internaționalizarea proiectului; mai exact, cu ajutorul acesteia rezolvăm partea de traducere a aplicației.

În acest fel, foarte simplu, tot conținutul aplicației poate fi schimbat în limba dorită, dacă sunt adăugate traducerile corespunzătoare.

3.3. Modele de Procesare a limbajului natural

3.3.1. FinBERT

Bidirectional Encoder Representations from Transformers sau BERT este unul dintre cele mai folosite modele în Procesarea de Limbaj Natural din ultimii ani. Este diferit față de modelele din anii anteriori, având un mecanism de self-attention, procesând textul bidirecțional.

Ce reprezintă un mecanism *self-attention*? În secțiunea "Self-Attention at a High Level" din [6], rezumând câteva idei principale din publicația Attention Is All You Need [7], self-attention este o metodă folosită pentru a "privi" alte cuvinte relevante din enunț în timpul procesării cuvântului curent. Bert [8] se bazează pe arhitectura Transformer [7], un model care a apărut în 2017, care depinde de aproximativ o sută de milioane de parametri.

Arhitectura Transformer, în figura 3.9, este compusă din următoarele părți:

- Tokenizarea textului
- Codificarea pozitională care injectează informații despre poziția inputului
- Codificarea contextuală a secvenței de intrare, pe nivelul de self-attention
- Nivelul feed forward care funcționează ca o memorie statică, cheie-valoare

- Nivelul de cross-attention decodifică secvența de ieșire

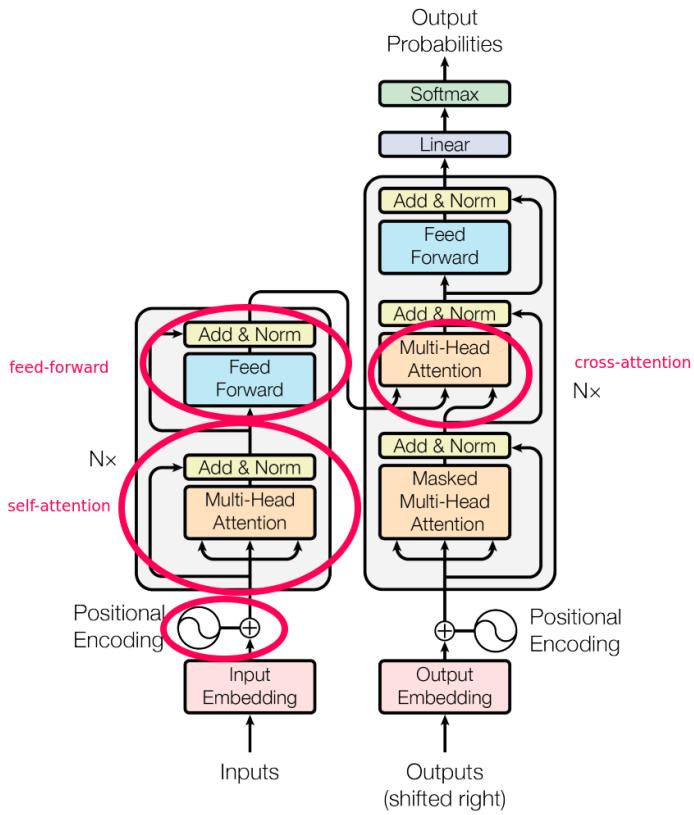


Figure 1: The Transformer - model architecture.

Figura 3.9: Arhitectura Transformers. Sursa [9]

În BERT, inputul este adus în primul nivel, apoi este transformat în alte codificări în următoarele nivele. Spre deosebire de abordările dinainte, prima etapă este etapa de pre-antrenare (en: pre-training), unde folosește 2 task-uri nesupervizate. După această etapă, modelul poate să fie perfecționat pentru o sarcină specifică, precum analiza de sentimente.

Analiza de sentimente pentru un text reprezintă extragerea sentimentelor sau opinioilor oamenilor dintr-un text scris. Analiza sentimentelor financiare diferă de analiza de sentimente prin scopul ei, pentru că rezultatul acestei analize va determina modul în care piața va fi influențată de deciziile financiare.

În [10], autorii spun despre FinBERT că este un model de analiză a textului bazat pe BERT, cu un domeniu specific. Acest model a fost antrenat pe un corpus de 4.9 miliarde de tokeni compuși din rapoarte ale companiilor, transcripturi ale conferințelor și rapoarte de analiză, conform [11].

În Statele Unite ale Americii, toate companiile cotate la bursă depun rapoarte anuale, cunoscute sub numele de Form 10-K și rapoarte trimestriale, cunoscute sub numele de 10-Q. Aceste documente sunt publice și oferă o imagine de ansamblu a situației financiare a companiei.

Cele menționate mai sus, alături de rapoarte ale analiștilor, care oferă măsuri rezumative cu recomandări, și teleconferințe trimestriale pe care directorii companiilor le au cu

investitorii pentru a discuta despre performanța firmei, formează un corpus finanțiar. Un corpus sau corpora reprezintă o colecție de texte sau fișiere audio organizate în seturi de date (en: datasets). Un corpus este, în general, folosit pentru antrenarea sistemelor bazate pe Inteligență Artificială.

3.3.2. Sumarizator de text

Sumarizarea este procesul de producere a unei versiuni mai scurte a unui text sau document, păstrând ideile importante din textul original. Deși modelul BERT este folosit pentru multe task-uri de NLP, în diferite domenii și subdomenii, acesta nu este soluția și pentru sumarizarea textului sau traduceri. În acest caz, apare BART (Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension), care folosește arhitectura sequence-to-sequence din Transformers și este un model folosit pentru clasificarea și generarea textului.

Este implementat ca un model sequence-to-sequence cu un codificator bidirectional pentru textul corupt și un decodificator stânga-dreapta. Arhitectura este asemănătoare cu arhitectura BERT, cu câteva diferențe, menționate în [12]:

- Fiecare nivel al decodificatorului efectuează suplimentar funcția de cross-attention pe ultimul strat ascuns al codificatorului
- BERT folosește o rețea feed-forward suplimentară, spre deosebire de BART

Acest model este pre-antrenat folosind un framework *Corrupt and Reconstruct* [13] care ia un text, aplică o funcție de zgomot, apoi antrenează modelul să reconstruiască textul original. Printre funcțiile de zgomot avem:

- Mascarea tokenilor (Token Masking) - se maschează anumiți token și se încearcă reconstrucția textului original
- Ștergerea tokenilor (Token deletion) - se șterg anumiți token și se încearcă reconstrucția textului original
- Rotația documentului (Document Rotation) - Mutarea anumitor tokeni pentru ca modelul să identifice începutul secvenței
- Permutarea enunțului (Sentence Permutation) - Se iau toate frazele din document și sunt amestecate
- Completarea texului (Text infilling) - Un număr fix de tokeni este șters și înlocuit cu o singură mască. Modelul învăță să prezică numărul de tokeni ce lipsesc dintr-un span și conținutul acestora [14]

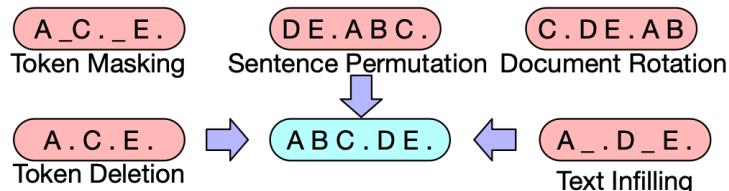


Figura 3.10: BART - Funcții de zgomot (Noising Functions). Sursa [15]

BART este folosit pentru sumarizarea textului; în funcție de abordare, aceasta poate fi de două tipuri: abstractivă sau extractivă.

Sumarizarea extractivă identifică și extrage cele mai semnificative enunțuri din text, pe când, prin sumarizarea abstractivă, se încearcă înțelegerea întregului text și generarea unui sumar cu un text parafrazat, cea de-a doua metodă fiind utilizată și de BART. Modelul ales în acest proiect este bazat pe modelele DistilBART și este evaluat folosind metrica ROUGE [16], bazat pe suprapunerea dintre secvența produsă și secvența corectă.

3.3.3. Keyphrase Extractor

Extragerea de cuvinte sau fraze cheie reprezintă o tehnică prin care sunt extrase cuvintele părțile importante dintr-un text.

În [17], autorii spun că frazele sau cuvintele cheie captează cele mai importante idei și identificarea lor automată ajută foarte mult în sarcini precum clasificarea, sumarizarea, etc. Modelul KBIR folosește o combinație de mascare a tokenilor (Masked Language Modeling - MLM), mascare a expresiilor (Keyphrase Bounday Infilling - KBI) și învățarea constrativă (Keyphrase Replacement Classification - KRC).

3.4. Aplicații similare

În momentul de față, există destul de puține aplicații publice pentru analiza textelor financiare.

3.4.1. Serviciul de înțelegere a limbajului natural dezvoltat de IBM

Acest serviciu are următoarele funcționalități:

- Analiza textului în mai multe domenii - Legal, Media și Financiar
- Textul analizat poate să fie copiat în field-ul din aplicație sau poate să fie referit prin URL
- Pentru partea de extractie din text, pot fi ca rezultat următoarele:
 - Entitatele extrase care pot apartine unui tip (ex: Organization, Job title, etc.)
 - Cuvintele cheie extrase
 - Conceptele extrase
 - Relațiile dintre entități
- Pentru partea de clasificare a textului, pot fi ca rezultat următoarele:
 - Scorul sentimentului pe întregul document
 - Scorul sentimentului pe fiecare entitate extrasă
 - Scorul sentimentului pe fiecare entitate cuvânt cheie sau frază
 - Clasificarea emoțiilor pe întregul document (ex: Sadness, Joy, Disgust, Fear, Anger, etc.)
 - Clasificarea emoțiilor pentru fiecare entitate
 - Clasificarea emoțiilor pentru fiecare cuvânt cheie sau frază
 - Clasificarea în categorii, ierarhic
- Pentru partea lingvistică a textului, pot fi ca rezultat următoarele:
 - Rolarile semantice (Subiect + Actiune + Forma obiectului)
 - Sintaxa - pentru fiecare token se specifică partea de vorbire

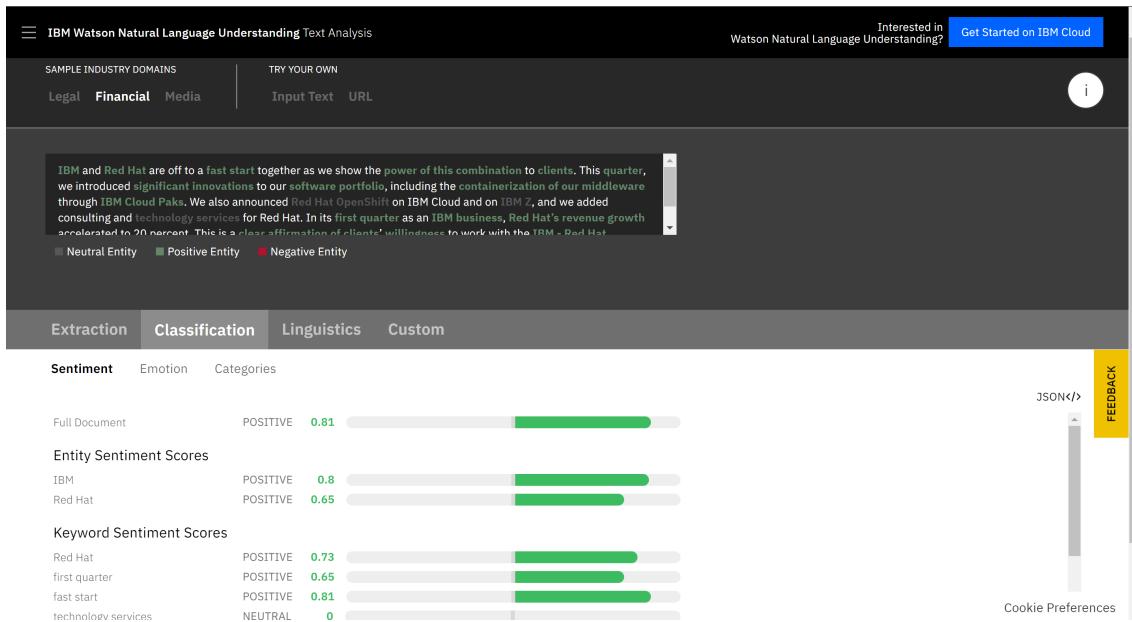


Figura 3.11: Servicii IBM

3.4.2. Google Cloud Natural Language

Acest serviciu are următoarele funcționalități:

- Entitatele extrase care pot apartine unui tip (ex: Organization, Location, Person, Event, Other, etc.)
- Scorul sentimentului și magnitudinea pe întregul document
- Scorul sentimentului și magnitudinea pe fiecare entitate extrasă
- Sintaxa - pentru fiecare token (inclusiv semne de punctuație) se specifică partea de vorbire(cu mai multe informații specifice pentru fiecare, spre exemplu, caz, gen, timp, număr), lemma, morfologie, etc.
- Clasificarea în categorii

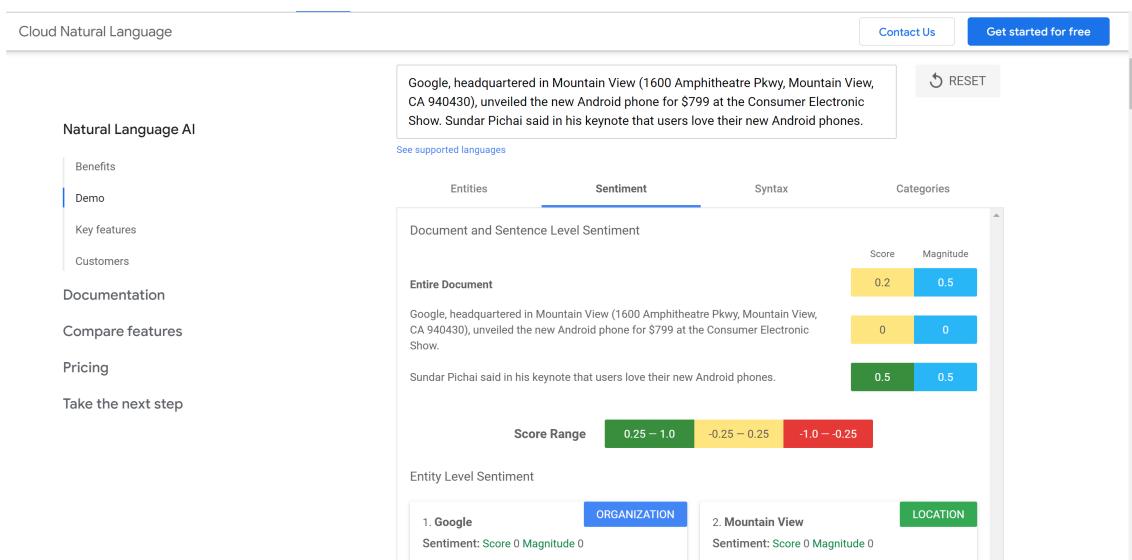


Figura 3.12: Servicii Google Cloud Natural Language

Language – Text Analytics and Beyond

Uncover insights such as key phrases, sentiment analysis, entities identification and more!

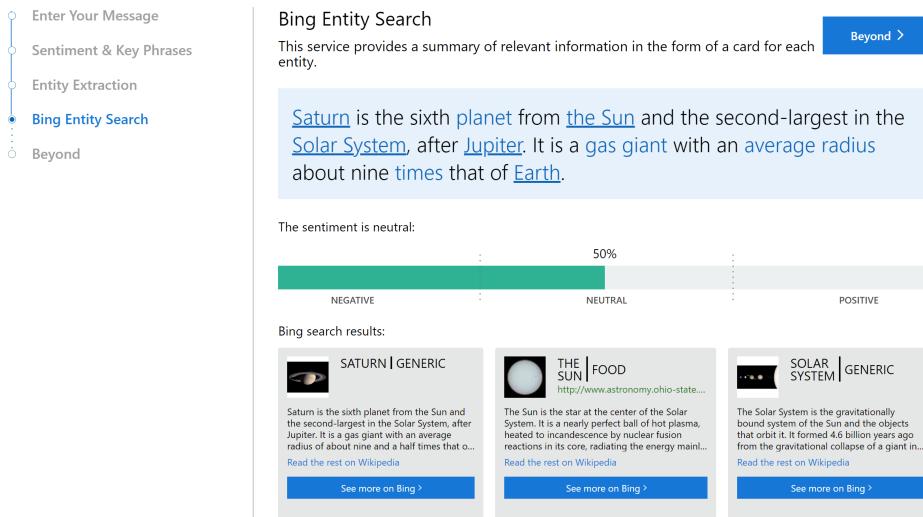


Figura 3.13: Servicii Microsoft AI

3.4.3. Microsoft AI

Acet serviciu are următoarele funcționalități:

- Scorul sentimentului pe întreg textul
- Extractia entitatilor, iar acolo unde este posibil, fiecare entitate se află sub un link de Wikipedia
- Bing Entity Search oferă un sumar al informațiilor relevante pentru fiecare entitate găsită la punctul anterior

3.4.4. Google Trends

Acet serviciu are următoarele funcționalități:

- Evoluția unui cuvânt sau a unei expresii într-o perioada de timp, într-o anumită regiune, dintr-o anumită categorie, căutare în mai multe formate
- Rezultatele sunt vizuale, afișate pe line chart-uri și hărți
- Sugestii de subiecte conexe și căutări similare
- Căutări populare zilnice și în timp real
- Căutările anului, pe categorii (Oameni, Emisiuni TV, Rețete, Simptome, etc.)

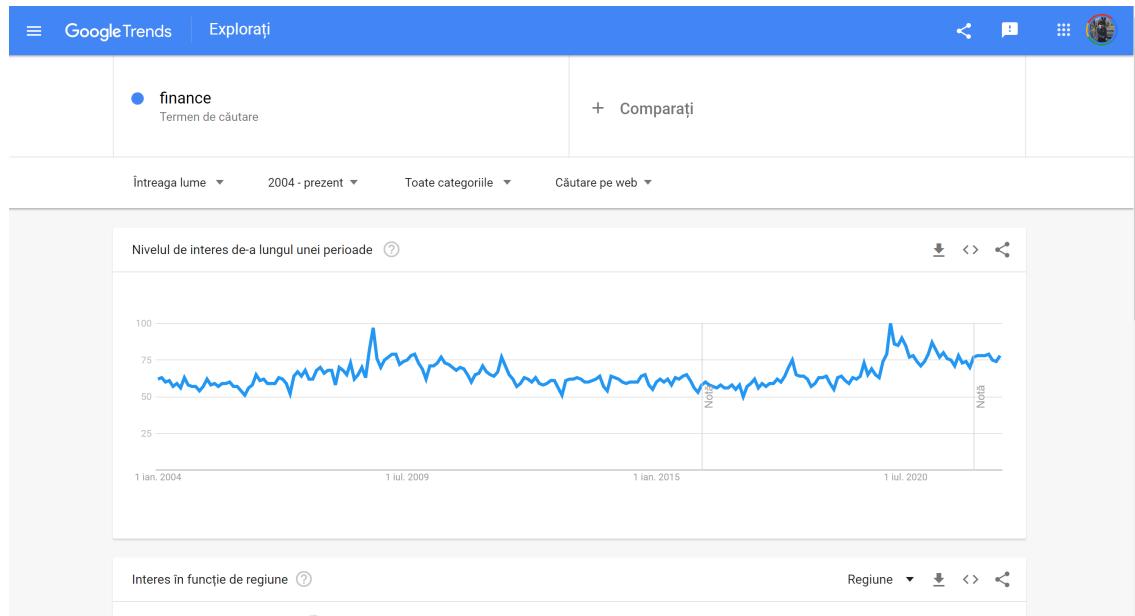


Figura 3.14: Servicii Google Trends

Tabela 3.1: Comparație între aplicația prezentată și ale servicii existente

	IBM	Microsoft AI	Google Search	Google Trends	Aplicație
Serviciu gratuit	NU	NU	NU	DA	DA
Sumarizare	NU	NU	NU	NU	DA
Extractie entități	DA	DA	DA	NU	DA
Analiză sentiment finanțiar	DA	NU	NU	NU	DA
Analiză popularitate	NU	NU	NU	DA	DA
Salvare analiză în cont	NU	NU	NU	NU	DA
UI ușor de înțeles și folosit	NU	NU	NU	DA	DA

Capitolul 4. Analiză și Fundamentare Teoretică

4.1. Cerințe funcționale și non-funcționale

4.1.1. Cerințe funcționale

- Pentru un utilizator neautentificat, sunt realizabile următoarele acțiuni:
 - Accesare pagină de Logare
 - Accesare pagină de Creare cont nou
 - Încercare de logare
 - Încercare de creare cont nou
- Pentru un utilizator autentificat, sunt realizabile următoarele acțiuni:
 - Accesare pagină de Dashboard
 - * Adăugare text în caseta de text
 - * Trimitere text spre procesare prin apăsarea butonului de Submit
 - * Vizualizare rezultate
 - * Generare grafic pentru vizualizarea evoluției cuvintelor cheie extrase într-o perioadă de timp selectată de utilizator
 - * Salvare articol (text analizat și rezultatele obținute)
 - Accesare pagină de Profil
 - * Editare câmp nume
 - * Editare câmp prenume
 - * Editare câmp nume de utilizator, cu condiția să nu fie atribuit altui cont
 - * Editare câmp email, cu condiția să nu fie atribuit altui cont
 - * Editare câmp parolă
 - Accesare pagina de Trenduri
 - * Generare grafic pentru vizualizarea celor mai populare 10 subiecte într-o perioadă de timp selectată de utilizator
 - * Vizualizarea unui WordCloud, constituit din cuvintele cheie sau expresiile extrase din textele analizate
 - Accesare pagină de Evoluție
 - * Generare grafic pentru vizualizarea numărului de apariții pentru un cuvânt ales și o perioadă de timp selectată de utilizator
 - Accesare pagină de Istoric
 - * Vizualizare articole salvate în istoric
 - * Accesarea analizei detaliate a unui articol selectat
 - * Ștergerea unui articol selectat

4.1.2. Cerințe pentru funcții de analiză a unui text

- Funcție de analiză a sentimentelor dintr-un text financiar
- Funcție de sumarizare a unui text
- Funcție de extragere a cuvintelor

4.1.3. Cerințe non-funcționale

- Securitatea
 - Pentru a accesa anumite resurse, utilizatorul trebuie mai întâi să își creeze un cont, apoi să se logheze în aplicație
 - La crearea contului, înainte să fie salvată, parola trece print-o funcție de hashing cu salt, cu scopul de a avea un rezultat diferit pentru același input al mai multor utilizatori, dacă va fi cazul
 - La logarea în aplicație se generează un token pentru a putea autoriza utilizatorul atunci când dorește să acceseze anumite resurse
- Compatibilitatea
 - Aplicația a fost creată și rulată utilizând ca sistem de operare Windows OS
 - Pentru a utiliza alt sistem de operare, spre exemplu MacOS, singura diferență este baza de date, fiind necesar găsirea unui echivalent al Microsoft SQL Server Management

4.2. Cazuri de utilizare

Folosind îndrumările din [18] pentru crearea și detalierea cazurilor de utilizare a aplicației, trebuie avute în vedere următoarele aspecte:

1. *Enunțarea precondițiilor*, adică a stării sau stărilor în care sistemul se află înaintea începerii execuțării operațiunilor specifice cazului de utilizare
2. *Descrierea flow-ului principal*, care trebuie să acopere următoarele: când și unde începe cazul, când cazul de utilizare interacționează cu actorii și când schimbă date, când cazul folosește date stocate sau dorește să stocheze datele, când și cum este finalizat cazul
3. *Descrierea flow-ului alternativ*, care trebuie să descrie ce se întâmplă cu sistemul în cazurile în care apare o eroare
4. *Enunțarea postcondițiilor*, adică a stării sau stărilor în care sistemul ajunge după ce cazul este finalizat

4.2.1. Utilizator neautentificat

Descrierea cazurilor de utilizare pentru un utilizator neautentificat în figura 4.1

1. Creare cont nou
 - Un utilizator neautentificat poate să acceseze pagina de creare cont nou și să se înregistreze în aplicație
 - Precondiții: utilizatorul neautentificat trebuie să se afle pe pagina de Creare cont nou
 - Flow principal

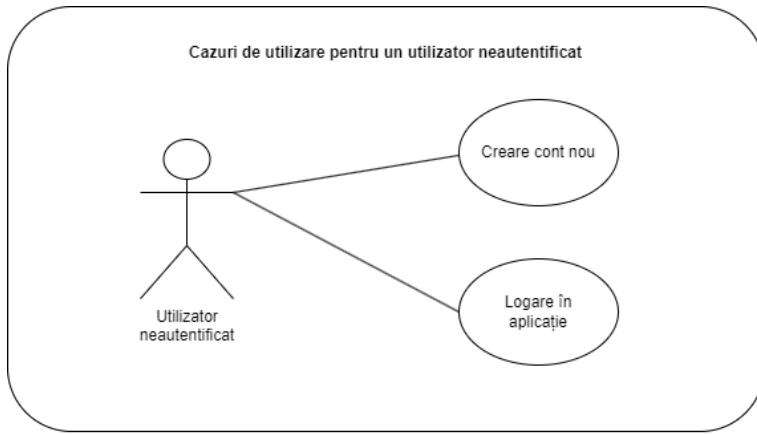


Figura 4.1: Cazuri de utilizare pentru un utilizator neautentificat

- (a) Utilizatorul neautentificat introduce informațiile necesare: nume, prenume, nume de utilizator, email și parola
 - (b) Utilizatorul apasă pe butonul de trimitere
 - (c) Crearea unui nou cont a fost realizată cu succes și utilizatorul este redirecționat la pagina de logare în aplicație
 - Flow alternativ
 - (a) Utilizatorul neautentificat introduce informațiile necesare: nume, prenume, nume de utilizator, email și parola
 - (b) Utilizatorul apasă pe butonul de trimitere
 - (c) Emailul sau numele de utilizator deja sunt folosite de alt cont
 - (d) Flow-ul este reluat
 - Postcondiții: Dacă a fost creat cu succes contul, utilizatorul este redirecționat la pagina de logare în aplicație
2. Logare
- Un utilizator neautentificat poate să acceseze pagina de logare și să intre în aplicație, dacă acesta are deja un cont creat
 - Precondiții: utilizatorul neautentificat trebuie să se afle pe pagina de logare
 - Flow principal
 - (a) Utilizatorul neautentificat introduce informațiile necesare: nume de utilizator sau email și parola
 - (b) Utilizatorul apasă pe butonul de trimitere
 - (c) Logarea s-a făcut cu succes și utilizatorul este acum autentificat și redirecționat la pagina de Dashboard
 - Flow alternativ
 - (a) Utilizatorul neautentificat introduce informațiile necesare: nume de utilizator sau email și parola
 - (b) Utilizatorul apasă pe butonul de trimitere
 - (c) Credențialele sunt greșite
 - (d) Flow-ul este reluat
 - Postcondiții: Dacă utilizatorul s-a logat în aplicație, este redirecționat la pagina de Dashboard

4.2.2. Utilizator autentificat

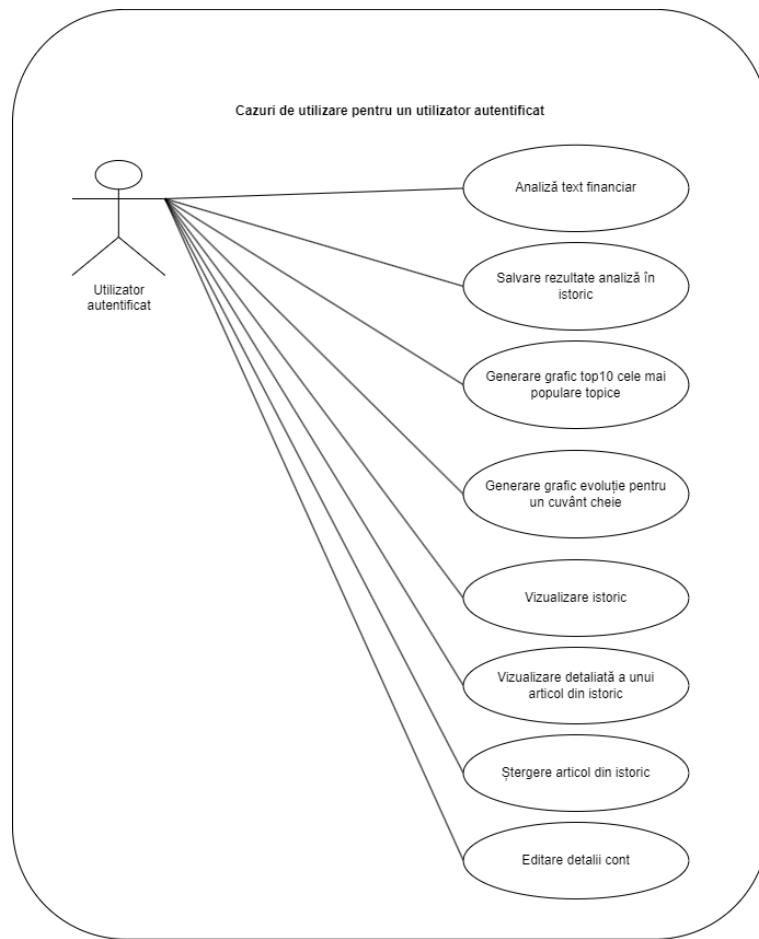


Figura 4.2: Cazuri de utilizare pentru un utilizator autentificat

Descrierea cazurilor de utilizare pentru un utilizator autentificat în figura 4.2

1. Analizare text financiar
 - Un utilizator autentificat poate să introducă un text financiar și să îl obțină o analiză detaliată
 - Precondiții: utilizatorul autentificat trebuie să se afle pe pagina de Dashboard
 - Flow principal
 - (a) Utilizatorul autentificat introduce textul
 - (b) Utilizatorul apasă pe butonul de trimis
 - (c) Apare un buton pentru a-l trimite pe utilizator într-o pagină cu rezultatele obținute în urma analizei
 - Flow alternativ
 - (a) Utilizatorul autentificat introduce textul
 - (b) Utilizatorul apasă pe butonul de trimis

- (c) Modelul care trebuie să transmită rezultatele încă se încarcă
 - (d) Flow-ul este reluat
 - Postcondiții: Dacă textul a fost analizat și apare butonul care îl trimite pe utilizator într-o pagină cu rezultatele obținute în urma analizei
2. Salvare rezultate analiză în istoric
- Un utilizator autentificat poate să salveze rezultatele detaliate ale analizei textului
 - Precondiții: utilizatorul autentificat trebuie să se afle în pagina de Dashboard, în partea cu toate informațiile detaliate
 - Flow principal
 - (a) Utilizatorul autentificat apasă pe butonul de salvare
 - (b) Se deschide un modal pentru confirmare
 - (c) Se confirmă și articoul analizat este salvat în istoricul utilizatorului
 - Postcondiții: Dacă utilizatorul a salvat articolul, acesta va apărea în tabelul din pagina de istoric

Diagrama FlowChart se găsește în figura 4.3.

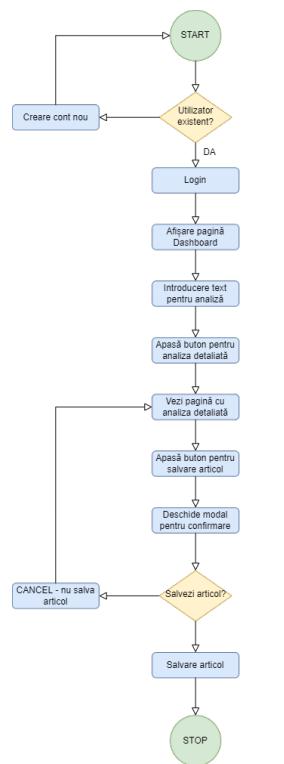


Figura 4.3: Diagramă FlowChart pentru salvarea rezultatelor unui text analizat în istoric

3. Generare grafic pentru top 10 cele mai populare subiecte din aplicație
- Un utilizator autentificat poate să genereze un grafic pentru a vedea top 10 cele mai populare cuvinte cheie (subiecte) din aplicație
 - Precondiții: utilizatorul autentificat trebuie să se afle în pagina de Trending

- Flow principal
 - (a) Utilizatorul autentificat selectează o perioadă
 - (b) Utilizatorul apasă pe butonul de trimis
 - (c) Apar rezultatele pe grafic
- Flow alternativ
 - (a) Utilizatorul autentificat selectează o perioadă
 - (b) Utilizatorul apasă pe butonul de trimis
 - (c) Utilizatorul este notificat că nu există date din perioada selectată de acesta
 - (d) Flow-ul este reluat, selectând altă perioadă
- Postcondiții: Utilizatorul rămâne în pagina de Trending, având posibilitatea selectării altrei perioade

Diagrama FlowChart se găsește în figura 4.4.

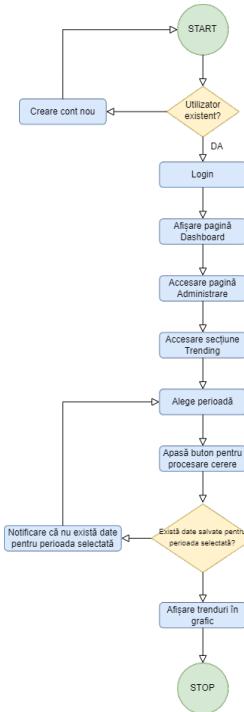


Figura 4.4: Diagramă FlowChart pentru generare grafic cu top 10 trenduri în aplicație

4. Generare grafic pentru a vizualiza evoluția unui cuvânt cheie, într-o anumită perioadă
 - Un utilizator autentificat poate să genereze un grafic pentru a vedea evoluția unui cuvânt cheie, într-o anumită perioadă
 - Precondiții: utilizatorul autentificat trebuie să se afle în pagina de Evoluție
 - Flow principal
 - (a) Utilizatorul autentificat selectează o perioadă
 - (b) Utilizatorul autentificat introduce cuvântul

- (c) Utilizatorul apasă pe butonul de trimitere
 - (d) Apar rezultatele pe grafic
- Flow alternativ
 - (a) Utilizatorul autentificat selectează o perioadă
 - (b) Utilizatorul autentificat introduce cuvântul
 - (c) Utilizatorul apasă pe butonul de trimitere
 - (d) Utilizatorul este notificat că nu există date, posibil din cauza perioadei sau din cauză că până acum nu a mai fost căutat cuvântul
 - (e) Flow-ul este reluat, selectând altă perioadă
 - Postcondiții: Utilizatorul rămâne în pagina de Evoluție, având posibilitatea selectării altei perioade sau introducerii altui cuvânt
5. Vizualizare istoric
- Un utilizator autentificat poate să vizualizeze istoricul
 - Precondiții: utilizatorul autentificat poate să se afle în orice pagină
 - Flow principal
 - (a) Utilizatorul autentificat selectează din bara de navigare butonul de Istoric
 - (b) Utilizatorul este redirecționat la pagina de Istoric, unde poate vedea tabelul cu articolele salvate până acum
 - Postcondiții: Utilizatorul este redirecționat la pagina de Istoric, unde poate vedea tabelul cu articolele salvate până acum
6. Vizualizarea detaliată a unui articol din istoric
- Un utilizator autentificat poate să vizualizeze mai multe detalii ale unui articol salvat în istoric
 - Precondiții: utilizatorul autentificat trebuie să se afle în pagina de Istoric
 - Flow principal
 - (a) Utilizatorul autentificat apasă pe butonul de Vezi mai mult din dreptul articoului pe care dorește să îl redeschidă
 - (b) Apare un modal de confirmare
 - (c) Utilizatorul confirmă că vrea să revadă analiza acelui articol
 - (d) Utilizatorul este redirecționat în pagina de Dashboard, de această dată fără opțiunea de a salva articolul (pentru că acesta deja există în istoric)
 - Flow alternativ
 - (a) Utilizatorul autentificat apasă pe butonul de Vezi mai mult din dreptul articoului pe care dorește să îl redeschidă
 - (b) Apare un modal de confirmare
 - (c) Utilizatorul nu confirmă că vrea să revadă analiza acelui articol
 - (d) Utilizatorul rămâne în pagina de Istoric
 - Postcondiții: Utilizatorul este direcționat în pagina cu analiza detaliată a articoului selectat

7. Stergere a unui articol din istoric

- Un utilizator autentificat poate să steargă un articol salvat în istoric
- Precondiții: utilizatorul autentificat trebuie să se afle în pagina de Istoric
- Flow principal
 - (a) Utilizatorul autentificat apasă pe butonul de Sterge din dreptul articolului pe care dorește să îl redeschidă
 - (b) Apare un modal de confirmare
 - (c) Utilizatorul confirmă că vrea să steargă articolul
 - (d) Articolul este șters și tabelul este actualizat
- Flow alternativ
 - (a) Utilizatorul autentificat apasă pe butonul de Sterge din dreptul articolului pe care dorește să îl redeschidă
 - (b) Apare un modal de confirmare
 - (c) Utilizatorul nu confirmă că vrea să steargă articolul
 - (d) Utilizatorul rămâne în pagina de Istoric și tabelul nu este actualizat
- Postcondiții: Se actualizează tabelul din istoricul utilizatorului și rămâne în pagina de Istoric

Diagrama FlowChart se găsește în figura 4.5.

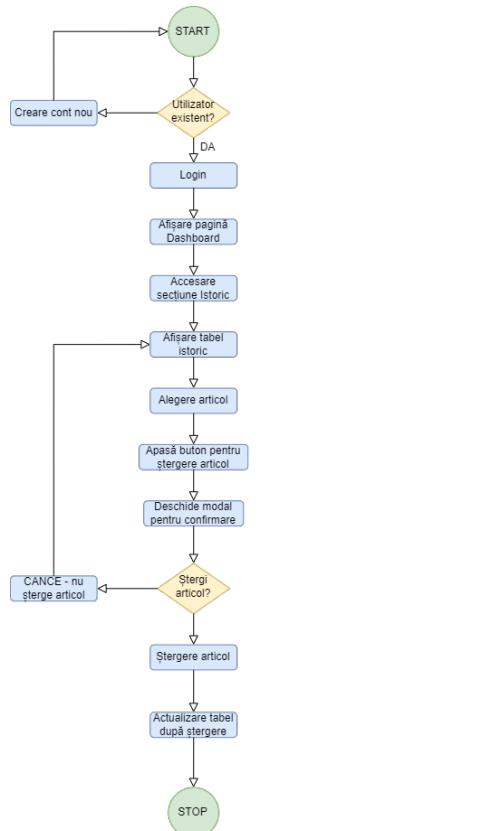


Figura 4.5: Diagramă FlowChart pentru stergere articol din istoric

8. Editare detalii cont

- Un utilizator autentificat poate să editeze detaliile contului
- Precondiții: utilizatorul autentificat trebuie să se afle în pagina de Profil
- Flow principal
 - (a) Utilizatorul autentificat editează unul sau mai multe din câmpurile: nume, prenume, nume de utilizator, email, parolă nouă și confirmă parola
 - (b) Utilizatorul apasă pe butonul de trimis
 - (c) Utilizatorul este notificat că detaliile contului au fost modificate cu succes
- Flow alternativ
 - (a) Utilizatorul autentificat editează unul sau mai multe din câmpurile: nume, prenume, nume de utilizator, email, parolă nouă și confirmă parola
 - (b) Unul dintre câmpurile de email sau nume de utilizator este deja folosit de alt cont
 - (c) Utilizatorul apasă pe butonul de trimis
 - (d) Utilizatorul este notificat că email-ul sau numele de utilizator este deja folosit de alt cont
 - (e) Flow-ul este reluat
- Postcondiții: Detaliile contului au fost modificate cu succes

4.3. Algoritmi utilizati

4.3.1. Algoritmul pentru autorizarea accesului utilizatorului la anumite resurse

Atunci când un utilizator nou își creează un cont și se loghează în aplicație, identitatea acestuia este verificată printr-un proces de autentificare.

Pentru a accesa anumite resurse, acesta trebuie să fie autorizat. Spre exemplu, pentru a efectua operații de ștergere a unor elemente importante, utilizatorul trebuie să aibă permisiunile necesare. Așadar, pentru partea de autorizare utilizator, mecanismul este următorul: în momentul în care clientul trimit un request de logare în aplicație, se returnează un token pentru autorizarea requesturilor viitoare, dacă nu există deja unul.

Validarea tokenului pentru un anumit utilizator depinde de utilizator (pentru că serviciul folosește un map cu userId și token), de valabilitatea tokenului și dacă valoarea stocată în map corespunde cu valoarea transmisă pentru verificare. Dacă toate cele 3 condiții sunt îndeplinite, utilizatorul are permisiunea de a efectua operația pentru care s-a făcut această verificare.

4.3.2. Algoritmul pentru crearea unei parole puternice

O funcție de hashing este folosită pentru a lua un mesaj de o orice lungime, apoi în urma procesării, va rezulta un răspuns de lungime fixă. Spre exemplu, funcția de hashing SHA512 va produce o valoarea de 512 biți, indiferent de lungimea valorii de intrare. Această funcție a fost folosită pentru a asigura securitatea parolei la înregistrarea unui utilizator nou.

Problema la funcțiile de hashing apare pentru că același input va produce același

output de lungime fixă. Pentru a asigura unicitatea rezultatului, se folosește un *salt*, un set de caractere care sunt adăugate la finalul parolei, în acest caz, înainte să treacă prin funcția de hashing.

4.4. Protocole utilizate

4.4.1. Protocolul HTTP

Protocolul HTTP este un protocol utilizat pentru a transmite date între un server Web și un browser (Google Chrome, Firefox, etc.). Se bazează pe protocolul de comunicare TCP/IP.

Se bazează pe mecanismul request-response, unde clientul (în acest caz, browserul) inițiază un request, apoi așteaptă un răspuns de la server.

Serverul procesează requestul primit de la client, apoi trimitе răspunsul, după care conexiunea se întrerupe. Așadar, clientul și serverul sunt conectați doar pentru requestul și răspunsul curent, concluzionând astfel că protocolul HTTP este un protocol fără conexiune.

Tot din acest motiv, clientul și serverul nu rețin informații unul despre celalălt, concluzionând că protocolul HTTP este un protocol fără stare, după cum putem afla din [19].

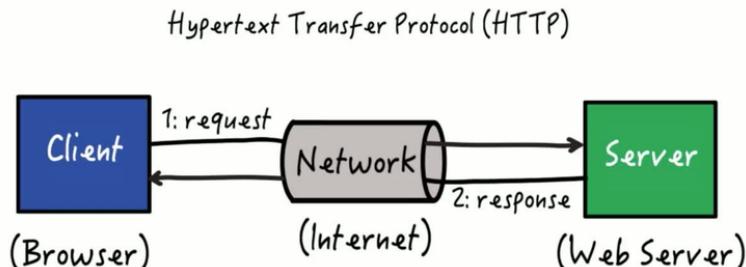


Figura 4.6: Protocolul HTTP. Sursa [20]

4.4.2. Protocolul TCP/IP

Pentru SQL Server, protocolul TCP/IP se utilizează cel mai des, pentru că acesta permite dispozitivelor să comunice între diferite dispozitive.

De fiecare dată când se trimit ceva prin intermediul Internetului, modelul TCP/IP structurează informația în pachete pe care apoi le transmite prin cele 4 nivele: Aplicație, Transport, Internet, Acces rețea. Informațiile sunt trimise în ordine, apoi sunt reasamblate în ordine inversă la celalalt capăt, aşa cum este menționat aici [21].

Protocolul TCP stabilește o conexiune între host-uri (server și baza de date, în acest caz), apoi asigură că pachetele de date sunt livrate.

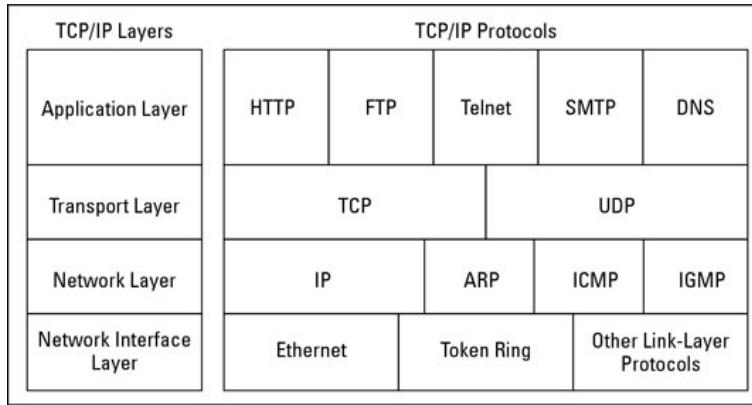


Figura 4.7: Modelul TCP/IP. Sursa [22]

4.4.3. Protocolul SMTP

SMTP sau Simple Mail Transfer Protocol, este un protocol de email utilizat pentru a trimite mesaje dintr-un cont de email în altul, prin intermediul Internetului, bazat pe adresele de email.

Permite trimitera unui mesaj spre unul sau mai mulți destinatari, includerea atașamentelor precum text, video, fotografii.

Atunci când un mail este trimis de la un client, acesta va ajunge la un server SMTP care e responsabil pentru transmiterea email-urilor mai departe, spre destinatar (serverul de mail al destinatarului).

De fiecare dată când un mail este trimis, o conexiune se va deschide între serverul SMTP și serverul destinatar, care va asigura transmiterea datelor spre un destinatar valid. Altfel, email-ul revine la expeditor.

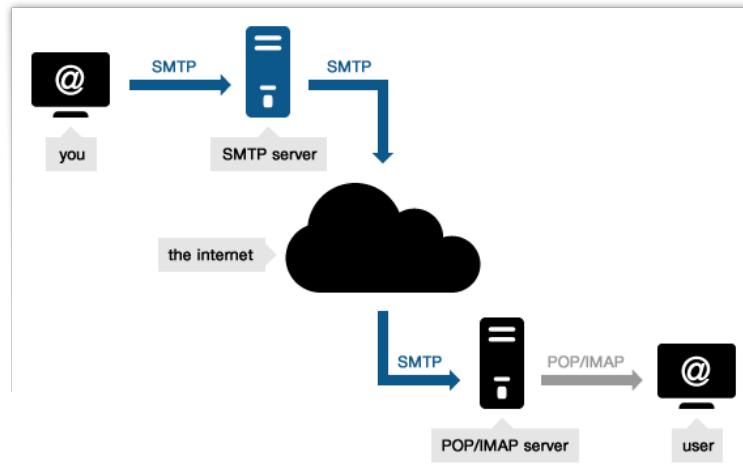


Figura 4.8: Protocolul SMTP. Sursa [23]

Capitolul 5. Proiectare de Detaliu și Implementare

5.1. Structura aplicației

Aplicația se bazează pe arhitectura client-server, folosind un mecanism de comunicare request-response. Clientul trimite un request spre server, iar serverul procesează cererea clientului și trimit un răspuns către acesta. Utilizarea acestui tip de arhitectură are următoarele avantaje, cum sunt menționate aici [24]

- Centralizarea informațiilor
 - Toate informațiile necesare pot fi găsite într-un loc
- Scalabilitatea
 - Numărul de clienți și servere poate fi ușor extins
 - Chiar dacă numărul clientilor sau al serverelor crește, nu reprezintă nicio problemă pentru că totul este deja centralizat și poate fi găsit și accesat cu ușurință de către persoanele autorizate
- Securitatea
 - Sistemul este bine protejat, putând adăuga anumite drepturi de acces pentru utilizatori

Un tip de arhitectură client-server este *Architectura 3-tier* sau Arhitectura pe 3 nivele. Acestea sunt: nivelul prezentare, fiind reprezentat de partea de interfață cu utilizatorul, nivelul aplicație, care cuprinde partea de logică și nivelul de date, care reprezintă mediul de stocare.

Nivelul de prezentare, de fapt, este reprezentat de partea de client a sistemului, iar partea de aplicație este reprezentată de partea de server alături de baza de date pentru a putea efectua diferite operații.

Partea de client a fost implementată utilizând React Typescript, iar partea de server a fost implementată utilizând ASP.NET Core. Pentru baza de date s-a utilizat Microsoft SQL Server Management Studio.

Clientul interacționează cu serverul folosind protocolul de comunicare HTTP. Acesta trimite request-uri spre server, urmând să fie procesate și returnat un răspuns. Pentru procesarea anumitor request-uri venite de la client, serverul interacționează și cu baza de date, folosind ca protocol de comunicare TCP/IP.

Operațiile pe care serverul le poate realiza la nivelul bazei de date constă în operații de stocare, citire a datelor deja stocate, actualizare a datelor și ștergere.

În figura 5.1 este reprezentată diagrama conceptuală a aplicației.

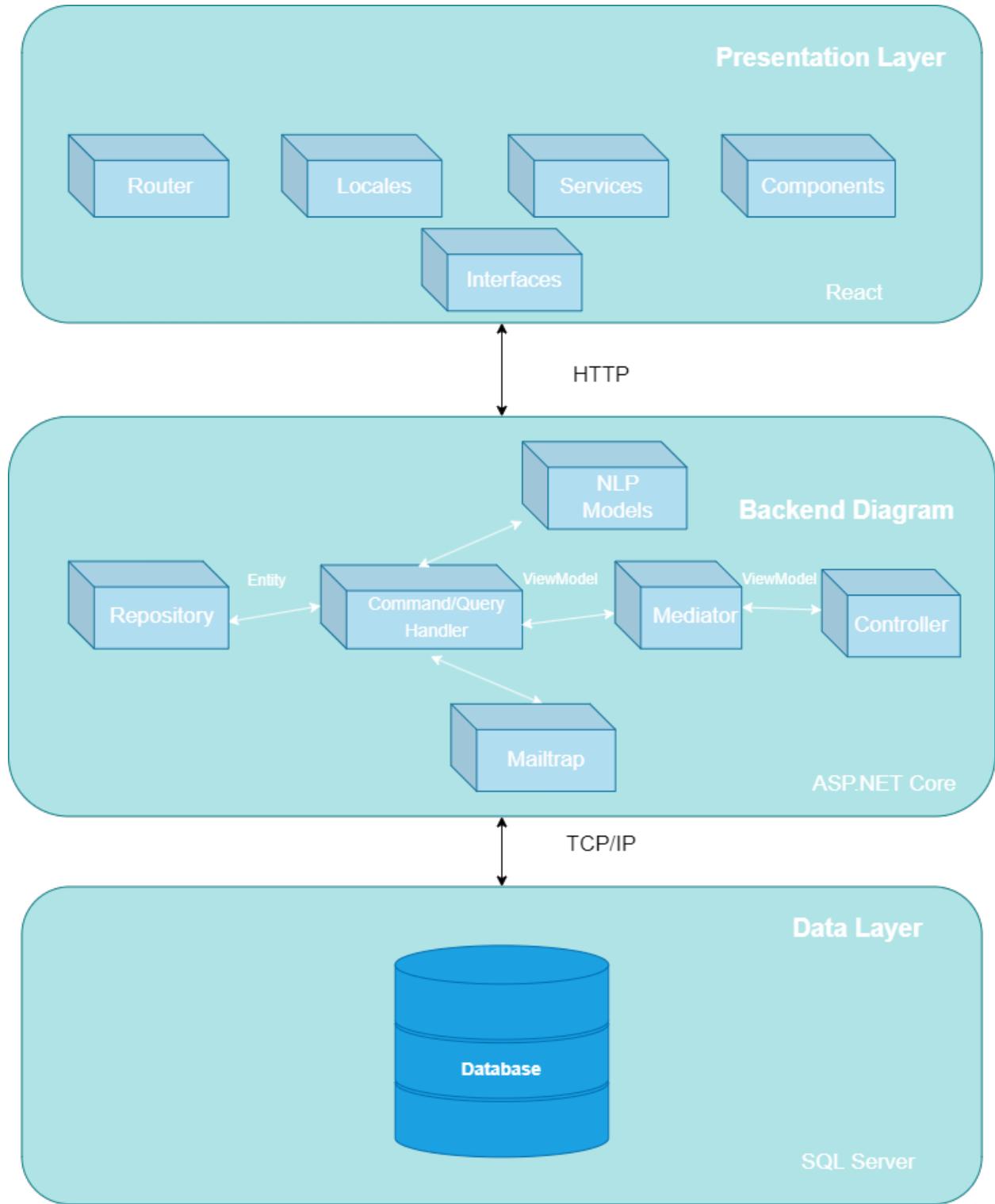


Figura 5.1: Arhitectura conceptuală a sistemului

5.2. Structura bazei de date

În figura 5.2 este reprezentată diagrama bazei de date. Această diagramă a fost creată cu ajutorul dbdiagram.io [25]

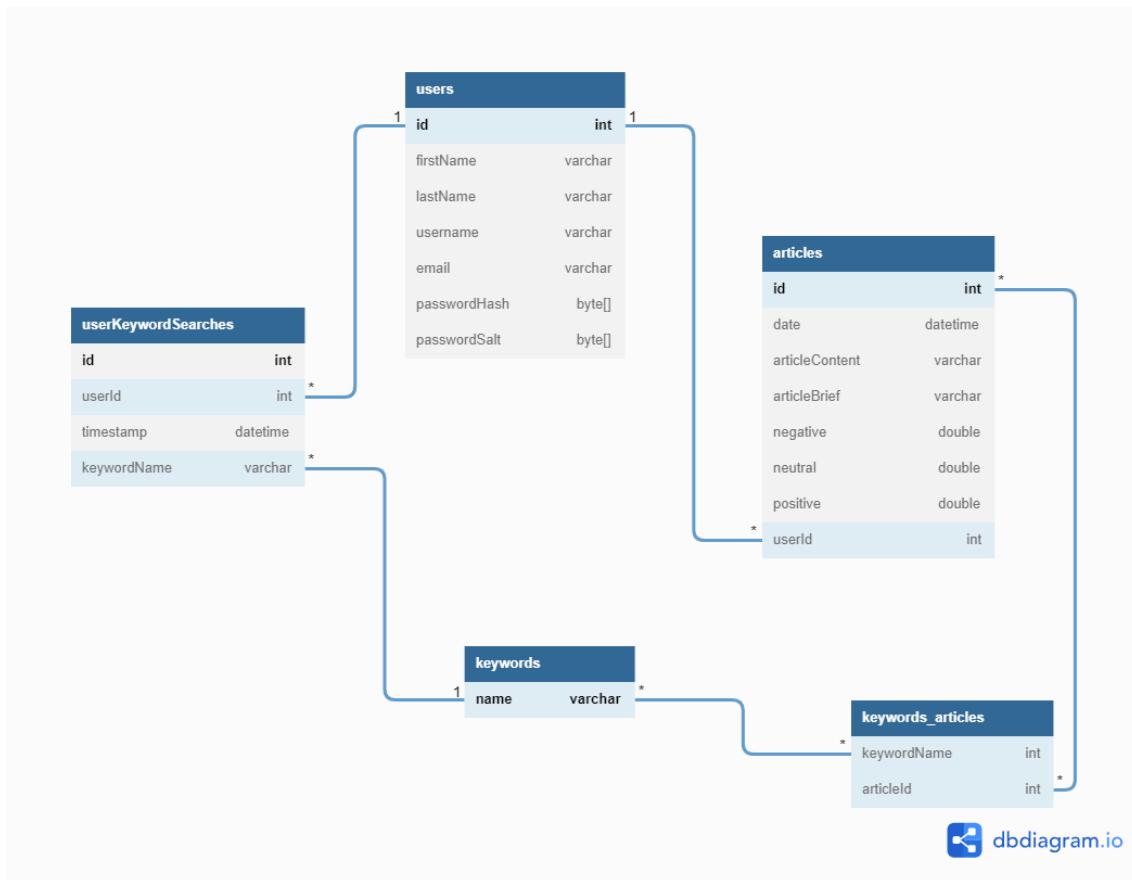


Figura 5.2: Diagrama bazei de date

Tabelele din baza de date sunt următoarele: Users, Keywords, UserKeywordSearches, Articles și o tabelă pentru a ilustra relația dintre Keywords și Articles.

Tabela Users

Această tabelă conține informațiile despre un utilizator.

- Id - reprezintă Id-ul utilizatorului, un identificator unic și este cheie primară în acest tabel
- FirstName - reprezintă prenumele utilizatorului, nu poate fi null și nu include caractere speciale sau cifre
- LastName - reprezintă numele utilizatorului, nu poate fi null și nu include caractere speciale sau cifre
- Email - reprezintă email-ul utilizatorului, nu poate avea duplicat, trebuie să aibă formatul corect pentru email
- Username - reprezintă numele de utilizator, nu poate fi null, poate avea caractere speciale sau cifre

- PasswordHash și PasswordSalt - reprezintă parola utilizatorului (explicată în partea de nivel de aplicație), nu poate fi null

Tabela Keywords

Această tabelă conține informațiile despre un keyword.

- Name - reprezintă identificatorul keyword-ului (cuvântului cheie sau frazei) și este cheie primară în acest tabel

Tabela UserKeywordSearches

Această tabelă conține informațiile despre un cuvânt cheie sau expresie identificată în textul analizat de utilizator.

- UserId - reprezintă cheia străină în tabel, legătură cu utilizatorul care în a cărui analiză a fost identificat cuvântul cheie sau expresia
- Timestamp - reprezintă data când a fost efectuată analiza textului în care a fost identificat cuvântul cheie sau expresia
- KeywordName - reprezintă cheie străină în tabel, legătură cu cuvântul cheie sau expresia identificată în textul analizat de utilizator

Tabela Articles

Această tabelă conține informațiile despre un articol. În acest caz, articol reprezintă un rezultat unei analize de text cu mai multe atribute care vor fi enumerate și explicate mai jos.

- Id - reprezintă Id-ul utilizatorului, un identificator unic și este cheie primară în acest tabel
- Date - reprezintă data când a fost salvat articolul (rezultatele obținute în urma analizei)
- ArticleContent - reprezintă conținutul inițial al textului, înainte de a fi efectuată analiza
- ArticleBrief - reprezintă conținutul summarizat al textului, după ce a fost efectuată analiza
- Negative - reprezintă scorul negativ obținut în urma analizei de sentiment pe text financiar
- Neutral - reprezintă scorul neutru obținut în urma analizei de sentiment pe text financiar
- Positive - reprezintă scorul pozitiv obținut în urma analizei de sentiment pe text financiar
- UserId - reprezintă cheie străină în tabel, legătură cu utilizatorul care a analizat textul

Relațiile între tabele sunt stabilite în felul următor: un utilizator poate realiza mai multe analize pe texte financiare, iar cuvintele cheie sau expresiile extrase reprezintă căutările utilizatorului. Deci un utilizator poate avea mai multe căutări, iar o căutare este legată

de un singur utilizator.

Mai mult decât atât, în urma unei analize, utilizatorul poate salva rezultatele în istoricul său personal, creând o înregistrare nouă pentru un articol în baza de date. Un utilizator poate avea mai multe articole salvate, iar fiecare articol aparține unui utilizator.

Într-un articol sunt salvate cuvintele cheie sau expresiile identificate în urma analizei, astădat un articol poate avea mai multe cuvinte cheie și un cuvânt cheie poate fi identificat în mai multe articole, pentru diferiți utilizatori.

5.3. Structura modulului de backend

În figura 5.3 este reprezentată diagrama modulului de backend.

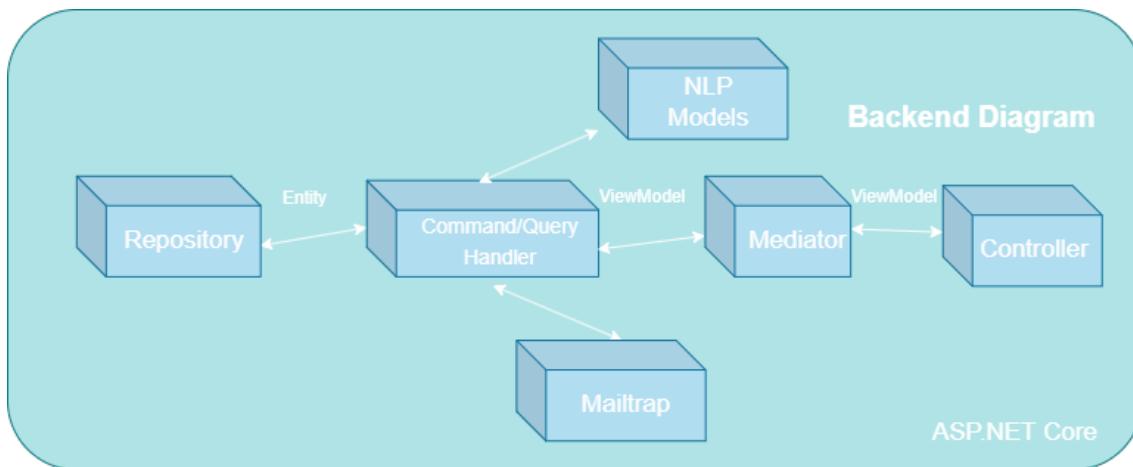


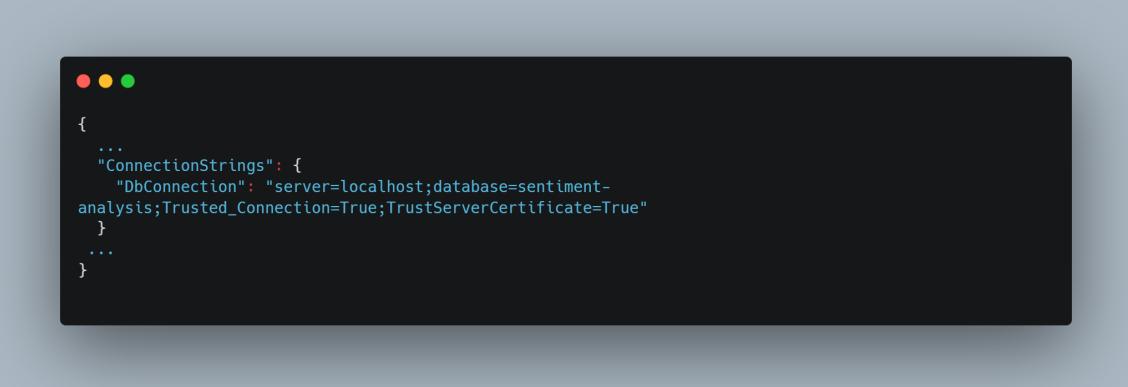
Figura 5.3: Diagrama modulului de backend

Modulul de backend este, de fapt, serverul din arhitectura client-server. Acesta se ocupă de procesarea request-urilor venite de la utilizator folosind un mediator.

Problema pe care o rezolvă adăugarea unui mediator este decuplarea componentelor unui sistem. Acestea nu vor comunica în mod direct între ele, interacțiunea fiind mediată. Avantajul folosirii interacțiunii mijlocite constă în faptul că, dacă vor apărea modificări noi, componentele deja existente nu vor fi afectate.

Poate exista, într-adevăr, și un dezavantaj - poate ajunge destul de complex în funcție de cerințele sistemului, aşa cum este menționat aici [26].

Legătura dintre backend și baza de date se face în felul următor: în fișierul din modulul de backend există un fișier *appsettings.json* unde se declară un ConnectionString pentru conexiunea la baza de date (poate fi văzut în figura 5.4, creată utilizând [27]). Acel string de conexiune conține numele bazei de date și alte câteva atrbute care permit conectarea, dar pot fi optionale.



```
{
  ...
  "ConnectionStrings": {
    "DbConnection": "server=localhost;database=sentiment-
analysis;Trusted_Connection=True;TrustServerCertificate=True"
  }
  ...
}
```

Figura 5.4: String de conectare backend cu baza de date

Repository

În acest nivel se află legătura dintre aplicație și baza de date. Modelele create aici vor deveni entități ale bazei de date. Spre exemplu, clasa User cu atributele Id, FirstName, LastName, Username, Email, PasswordSalt, PasswordHash, va determina structura și atributele tabelei Users din baza de date, cea din urmă având toate atributele prezentate în clasă.

După crearea unui model, Entity Framework Core permite configurarea suplimentară a modelului, după cum poate fi văzut în figura 5.5 și nu mai sunt necesare adnotări pentru atribut. În acest mod, a fost specificat că atributul de Id este cheie primară (implicit nu poate fi null), iar celelalte atribută FirstName, LastName, Username, Email, PasswordHash, PasswordSalt trebuie să fie diferite de null.



```
public class UserConfiguration : IEntityTypeConfiguration<User>
{
    public void Configure(EntityTypeBuilder<User> builder)
    {
        builder.HasKey(b => b.Id);
        builder.Property(b => b.FirstName).IsRequired();
        builder.Property(b => b.LastName).IsRequired();
        builder.Property(b => b.Username).IsRequired();
        builder.Property(b => b.Email).IsRequired();
        builder.Property(b => b.PasswordHash).IsRequired();
        builder.Property(b => b.PasswordSalt).IsRequired();
    }
}
```

Figura 5.5: Configurare pentru modelul unui utilizator

Aici se introduce conceptul de DbContext, care reprezintă o sesiune cu baza de date, când se pot face operații de tip stocare, interogare, actualizare și ștergere a elementelor din baza de date, creare de modele și maparea datelor.

DbContext este responsabil de deschiderea și gestionarea conexiunilor cu baza de date. Pentru crearea unui model, DbContext îl construiește bazându-se pe configurarea dată și apoi este mapat. Pentru maparea datelor, rezultatele interogărilor sunt mapate la

instante și entități definite de utilizator, cum sunt în figura 5.6.

Se adaugă conceptul de DbSet, prin care este posibilă aplicarea operațiilor de tip stocare, interogare, actualizare, ștergere pe acea entitate.

Clasele DbSet sunt proprietăți ale DbContextului și sunt mapate la baza de date care iau numele dat de utilizator. DbSet este o implementare a patternului Repository, care încearcă să adauge un layer între nivelul de acces la baza de date și nivelul de logică.



```

public class Context : DbContext
{
    public Context()
    {
    }

    public Context(DbContextOptions<Context> options) : base(options) { }

    public DbSet<User> Users { get; set; }
    public DbSet<Article> Articles { get; set; }
    public DbSet<Keyword> Keywords { get; set; }
    public DbSet<UserKeywordSearch> UserKeywordSearches { get; set; }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.ApplyConfiguration(new UserConfiguration());
        modelBuilder.ApplyConfiguration(new ArticleAnalysisConfiguration());
        modelBuilder.ApplyConfiguration(new UserKeywordSearchConfiguration());
        modelBuilder.ApplyConfiguration(new KeywordConfiguration());
    }
}

```

Figura 5.6: Configurare DbContext

Command/Query Handler

Aici se află partea de logică și procesare a datelor din aplicație. Prin procesarea datelor înțelegem toate interacțiunile ce pot exista cu baza de date și din această cauză a fost introdus un pattern al cărui scop este de separare a operațiilor de citire și scriere în baza de date - CQRS sau Command and Query Responsibility Segregation.

Așadar, se face o separare în 2 modele, comenzi și interogări. Comenzile sunt, de fapt, operațiile ce presupun stocarea, actualizarea sau ștergerea din baza de date, iar interogările reprezintă citirile din baza de date.

În partea de comenzi, operațiile care se fac sunt următoarele, în funcție de clasă:

- User
 - AddUserCommand - care adaugă un utilizator nou, folosită pentru crearea unui cont nou
 - EditUserCommand - care actualizează detaliile unui utilizator
 - DeleteUserCommand - care șterge un utilizator

- UserKeywordSearches
 - AddUserKeywordSearchCommand - care adaugă o nouă căutare ce cuprinde cuvântul cheie sau expresia identificată în textul analizat
- Articles
 - AddArticleCommand - care adaugă un nou articol ce cuprinde toate rezultatele analizei efectuate pe un text finanțier
 - DeleteArticleCommand - care șterge un articol ce cuprinde toate rezultatele analizei efectuate pe un text finanțier

În partea de interogări, operațiile care se fac sunt următoarele, în funcție de clasă:

- User
 - GetUserDetailsQuery - care cere informațiile despre un anumit utilizator, într-un anumit format solicitat
 - EditUserCommand - care actualizează detaliile unui utilizator
 - DeleteUserCommand - care șterge un utilizator
- UserKeywordSearches
 - GetKeywordStatisticsQuery - care interoghează baza de date și trimite înregistrările găsite pentru un anumit cuvânt cheie sau expresie într-o perioadă selectată
 - GetSearchNumberOfOccurrencesQuery - care interoghează baza de date și trimite înregistrările pe zile pentru itemii (cuvinte cheie sau expresii) găsite în perioada selectată de timp
 - FilterSearchesByPeriodCommand - care interoghează baza de date și trimite înregistrările găsite într-o perioadă selectată pentru cele mai populare 10 subiecte
- Articles
 - GetArticlesQuery - care interoghează baza de date și trimite înregistrările găsite pentru un anumit input, adică toate articolele salvate ale unui utilizator
 - GetUserArticleQuery - care interoghează baza de date și trimite informații despre articolul selectat pentru ca utilizatorul să poată revedea analiza detaliată
- TextAnalysis
 - GetSentimentScoreQuery - care realizează inferența modului de analiză de sentiment având drept input textul introdus de utilizator
 - GetTextSummaryQuery - care realizează inferența modului de sumarizare a textului având drept input textul introdus de utilizator
 - GetKeyphrasesQuery - care realizează inferența modului de extragere a cuvintelor cheie sau expresiilor având drept input textul introdus de utilizator



```

public class EditUserCommand : IRequest<bool>
{
    public Guid Id { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public string Username { get; set; }
    public string Email { get; set; }
    public string? NewPassword { get; set; }
    public string? ConfirmPassword { get; set; }
}

```

Figura 5.7: Comandă de editare utilizator

Model de comandă Interfața IRequest acceptă tipul obiectului pe care Handlerul ar trebui să îl returneze, în acest caz, o valoare true sau false. Un handler pentru această comandă este următorul, în figura 5.8.



```

public class EditUserCommandHandler: IRequestHandler<EditUserCommand, bool>
{
    private readonly Context _context;

    ...

    public EditUserCommandHandler(Context context)
    {
        _context = context;
    }

    ...
}

```

Figura 5.8: Handler pentru comanda de editare utilizator

IRequestHandlerul acceptă 2 tipuri de parametri, după cum e specificat și aici [28]: requestul la care trebuie să răspundă și tipul care trebuie returnat.

Model de query În figura 5.9 este un query care solicită ca răspuns un ArticleViewModel (un model pentru vizualizare a unui articol salvat de utilizator), atunci când primește ca identificatori un UserId și un ArticleId. Handlerul poate fi văzut în figura 5.10.

```

public class GetUserArticleQuery : IRequest<ArticleViewModel>
{
    public Guid UserId { get; set; }
    public Guid ArticleId { get; set; }
}

```

Figura 5.9: Query de citire a informațiilor despre un articol salvat de utilizator

```

public class GetUserArticleQueryHandler : IRequestHandler<GetUserArticleQuery, ArticleViewModel>
{
    private readonly Context _context;

    ...

    public GetUserArticleQueryHandler(Context context)
    {
        _context = context;
    }

    ...
}

```

Figura 5.10: Handler al unui query de citire a informațiilor despre un articol salvat de utilizator

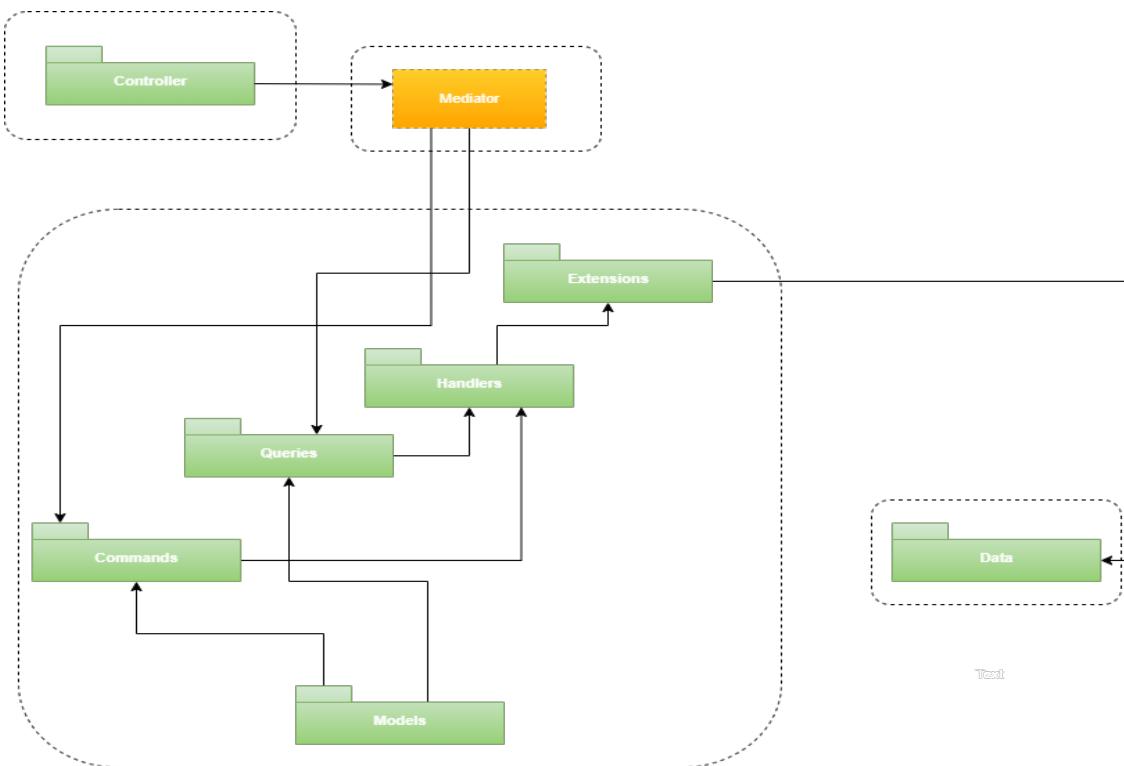


Figura 5.11: Diagrama pachetelor din modulul de backend

După cum se poate vedea, handlerul este asemănător cu handler-ul comenzi de mai sus, diferența o face rezultatul trimis și metodele de validare din clase. Tot în partea de logică a aplicației, sunt câteva handlere cu funcționalitate mai specială.

Dăugare user nou

La crearea unui cont nou, în comandă sunt scrise toate informațiile necesare. De aici, câmpul de email mai este folosit și în trimitera unui mail de confirmare a înregistrării. Pentru acest lucru am folosit Mailtrap, un server SMTP care ajută la testarea acestei funcționalități. Handlerul pentru această funcționalitate este în figura 5.12.



```

public async Task<bool> Handle(AddUserCommand request, CancellationToken cancellationToken)
{
    await ValidateIfUserExists(request.Email, request.Username);

    _hashingUtils.CreatePasswordHash(request.Password, out byte[] passwordHash, out byte[] passwordSalt);

    var user = request.ToUser();
    user.PasswordHash = passwordHash;
    user.PasswordSalt = passwordSalt;

    if (_context.Users.Add(user) != null)
    {
        _sendMail.SendEmail(request.Email, "Registration process report", "Successful registration.");
    }

    await _context.SaveChangesAsync(cancellationToken);
    return true;
}

```

Figura 5.12: Handler al unui query de citire a informațiilor despre un articol salvat de utilizator

Mai întâi se verifică dacă mai există alt utilizator cu același email sau nume de utilizator. În cazul în care există, va fi aruncată o excepție, altfel se continuă cu crearea unei parole criptate, se adaugă noul utilizator în baza de date și se trimită un email. Un exemplu de email este în figura 5.13.

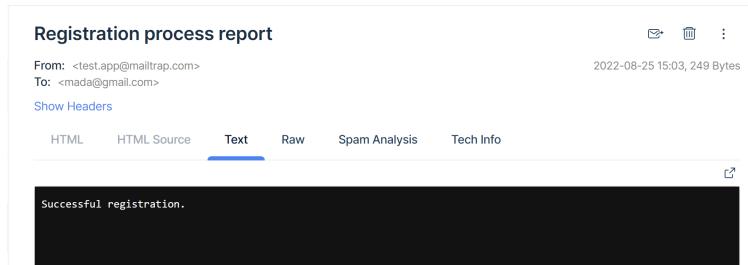


Figura 5.13: Email de confirmare a procesului de creare de cont cu succes

Handlerele care se ocupă de analiza sentimentelor dintr-un text financiar, sumarizarea textului și extragerea cuvintelor cheie se află tot aici. Acestea folosesc un serviciu pentru a apela metodele specifice, exemplu în figururile 5.14 și 5.15 .

Handlerul apelează metoda GetKeyphrases, având ca parametru textul introdus de utilizator.

HuggingFaceHelper este utilizat ca un serviciu și este posibilă apelarea metodelor din clasă.

Metoda menționată apelează altă metodă, CallModel, cu un URL pentru modelul care

```

public class GetKeyphrasesQueryHandler : IRequestHandler<GetKeyphrasesQuery, IEnumerable<string>>
{
    private readonly HuggingFaceHelper _huggingFace;

    public GetKeyphrasesQueryHandler(HuggingFaceHelper huggingFace)
    {
        _huggingFace = huggingFace;
    }

    public async Task<IEnumerable<string>> Handle(GetKeyphrasesQuery request, CancellationToken cancellationToken)
    {
        return await _huggingFace.GetKeyphrases(request.Args);
    }
}

```

Figura 5.14: Handler pentru extragerea cuvintelor cheie

extrage cuvintele cheie și inputul utilizatorului. În această metodă, se creează un body în format JSON pentru un request de tip POST, se trimit, apoi se așteaptă răspunsul.

Dacă răspunsul nu conține mesaj de eroare, acesta este transmis spre client, altfel se face o extra-validare pentru a detecta dacă eroarea primită este o eroare din cauza încărcării modelului, iar în cazul care este, metoda se va apela recursiv pentru a porni modelul și a primi rezultatul așteptat, după ce vor mai fi efectuate câteva parsări pentru formatul corect al acestuia.

Pentru cazurile în care metodele de validare aruncă excepții, pentru claritate cât mai mare a fluxului, excepțiile au fost customizate cu scopul de a fi transmise în format JSON în frontend și pentru a ajuta utilizatorul.

```

private async Task ValidateUserExists(AddArticleCommand command)
{
    var user = await _context.Users.Where(u =>
        u.Id.Equals(command.UserId)).FirstOrDefaultAsync();
    if (user == null)
    {
        throw new CustomException(ErrorCode.AddArticle_User, "User does not exist.");
    }
}

```

Figura 5.16: Exemplu validare care aruncă o excepție customizată

Metoda din figura 5.16 este folosită pentru validarea existenței unui utilizator al cărui Id corespunde cu cel din request.

Dacă nu a fost găsit un utilizator și variabila este nulă, atunci o să se arunce o excepție cu mesajul din figură și un status customizat.



```

private async Task<string> CallModel(string uri, string args)
{
    var bodyParams = new Dictionary<string, string>
    {
        { "inputs", args }
    };

    var response = await _client.PostAsJsonAsync(new Uri(uri), bodyParams);

    var result = await response.Content.ReadAsStringAsync();

    if (!result.Contains("\\"error\""))
        || !result.Contains("\\"estimated_time\"")
        || !result.Contains("is currently loading")) return result;

    try
    {
        var error = JsonConvert.DeserializeObject<LoadingModelError>(result);
        if (error is not null)
        {
            await Console.Out.WriteLineAsync(error.error + ". Estimated wait: " +
error.estimated_time + " s.");
            Thread.Sleep(TimeSpan.FromSeconds(error.estimated_time + SleepBuffer));
            return await CallModel(uri, args);
        }
    }
    catch (JsonException e)
    {
    }

    return result;
}

public async Task<IEnumerable<string>> GetKeyphrases(string args)
{
    var result = await CallModel(KeyphraseExtractorUri, args);
    var parsedResult = JsonConvert.DeserializeObject<ICollection<KeyphraseResultModel>>(result);

    if (parsedResult is null)
    {
        throw new CustomException(
            ErrorCode.GetKeyphrases_ParsingFailure,
            "Could not parse result from API: " + result);
    }

    return parsedResult.Select(r => r.word.Trim()).Distinct();
}

```

Figura 5.15: Metode pentru a accesa funcționalitățile modelului NLP

Controllers

Aici este nivelul în care modulul de frontend, clientul, interacționează cu modulul de backend, serverul, prin intermediul endpointurilor definite.

Un endpoint este, de fapt, un canal de comunicare, care include URL-ul spre o resursă.

Controlerle declarate sunt: UserController, ArticleController, UserKeywordsSearchController, TextAnalysisController.

UserController conține endpoint-uri pentru adăugarea unui utilizator nou, logarea în aplicație, afișarea informațiilor unui utilizator, cât și editarea acestora.

Diagrama de secvență pentru procesul de logare în aplicație poate fi văzută în figura 5.25. UserKeywordsSearchController conține endpointuri pentru citirea căutărilor în funcție de anumite condiții.

TextAnalysisController conține endpointuri pentru extragerea cuvintelor cheie, sumari-

zarea textului și analiza sentimentelor pe text finanțiar.

Tot aici sunt declarate și tipul requesturilor: POST pentru adăugare, DELETE pentru ștergere, PUT pentru update, etc., după cum se poate vedea în figura 5.17. Mediatorul este adăugat pentru a gestiona (în handle) requesturile, în funcție de tipul acestora: query pentru citiri din baza de date și commands pentru stocări sau actualizări ale bazei de date.

```

public class TextAnalysisController : Controller
{
    private readonly IMediator _mediator;

    public TextAnalysisController(IMediator mediator)
    {
        _mediator = mediator;
    }

    [HttpPost]
    [Route("summarize")]
    public async Task<IActionResult> Create([FromBody] GetTextSummaryQuery query)
    {
        return Ok(await _mediator.Send(query));
    }
    ...
}

```

Figura 5.17: Exemplu de controller cu Mediator

5.4. Structura modulului de frontend

În figura 5.18 este reprezentată diagrama modulului de backend.

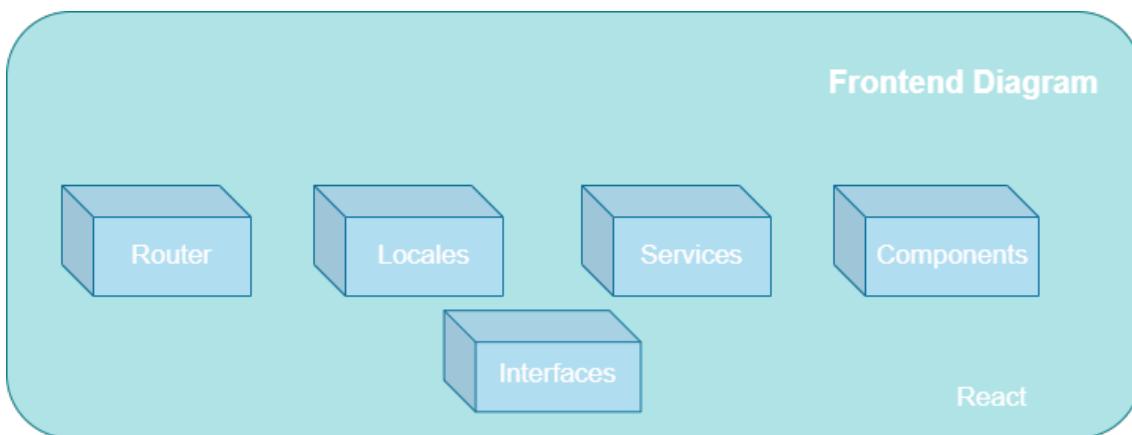


Figura 5.18: Diagrama modulului de frontend

Diagrama pachetelor din acest modul poate fi văzută în figura 5.19.

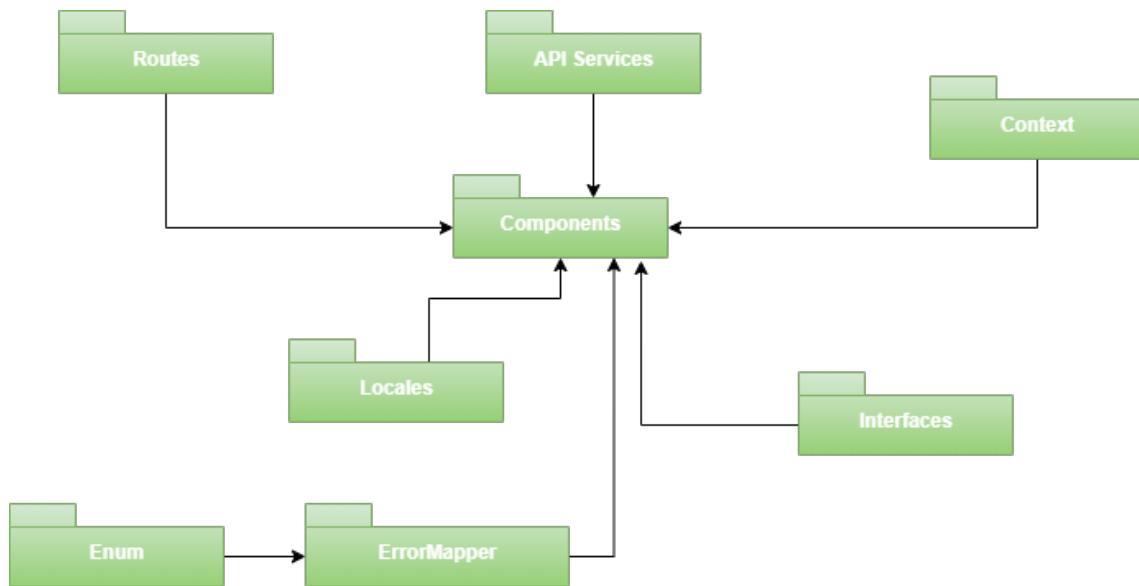


Figura 5.19: Diagrama pachetelor din modulul de frontend

Routes

În modulul de frontend, componenta de Router se ocupă de rutare. În general, o rută randează o pagină pentru un anumit URL. În aplicație, pentru securitatea utilizatorului, au fost create rute publice și private.

Rutele publice reprezintă rutele pe care utilizatorul le poate accesa atunci când nu este autentificat, adică poate accesa paginile de Creare cont nou și de Logare în aplicație.

Pentru un utilizator neautentificat, orice încercare de a modifica forțat URL-ul și a accesa o rută privată, se rezumă la o redirecționare la o pagină cu rută publică. Pentru rutele private, utilizatorul trebuie să aibă un cont creat și să fie logat. Pentru un utilizator autentificat, orice încercare de a modifica forțat URL-ul și accesa o rută publică, se rezumă la o redirecționare la pagina de Dashboard.

Pentru a putea gestiona global starea și pentru a obține statusul utilizatorului, dacă acesta este logat sau nu, se folosește un Context 5.20 [29].

Locales

Internationalizarea este implementarea unei aplicații astfel încât aceasta să fie disponibilă pentru utilizatori, indiferent de regiune.

Acest obiectiv a fost atins folosind i18n. Au fost create traduceri pentru toate textele din aplicație, fiind disponibile atunci când utilizatorul folosește switch-ul de limbă. Preferința utilizatorului va rămâne salvată, astfel încât atunci când va închide browserul și va reveni, va găsi aplicația în aceeași limbă.

În fișierul index.ts din figura 5.21, trebuie asigurat faptul că librăria i18n este instalată și importurile sunt făcute pentru a beneficia de funcționalitate.

Cu ajutorul funcției `use(...)` se pasează pluginul de i18n, apoi este inițializat cu resursele,

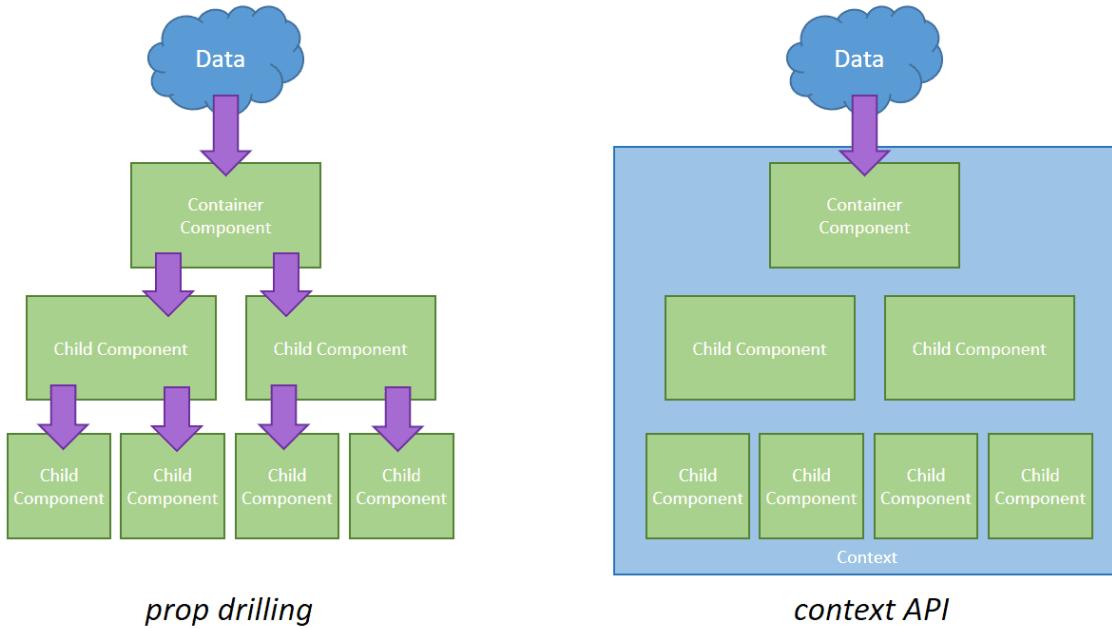


Figura 5.20: Contextul în React

care sunt fișierele de trăduceri. În această aplicație, trăducerile sunt pentru limba Engleză și limba Română.

lng este limba de început, dacă în localStorage există valoare pentru atributul 'language', atunci va fi folosit, altfel va fi folosită limba engleză.

Proprietatea de interpolare se referă la integrarea dinamică a valorilor în trăduceri, iar opțiunea de escapeValue se referă la escaparea valorilor pentru a evita atacurile de tip XSS (Cross Site Scripting). Acest tip de atac apare atunci când scripturi malicioase sunt injectate în paginile web, vizând alți utilizatori ai aplicației.

Atributul de keySeparator se folosește pentru a separa cheile cu un caracter definit în această configurație. Pentru că trăducerile sunt JSON cu format key-value, este recomandat să fie setat pe false.

Services

Pentru partea de servicii API și comunicare cu serverul, Axios este o librărie care ajută la creare request-urilor HTTP la endpoint-urile REST pentru a executa operații de stocare, citire, actualizare, ștergere (CRUD).

Folosind Axios cu un URL și, optional un body, se obține un Promise care returnează un răspuns, după cum se poate vedea în figura 5.22.

Components

Componentele folosite în aplicație sunt din librăriile Ant Design, iar graficele sunt din librăria recharts. Pentru customizarea ambelor componente s-a folosit styled-components, cu o metodă de a aplica stilul numită CSS-in-JS.



```

import i18n from 'i18next';
import { initReactI18next } from 'react-i18next';
import { TRANSLATIONS_EN } from './en';
import { TRANSLATIONS_RO } from './ro';

i18n
  .use(initReactI18next)
  .init({
    keySeparator: false,
    interpolation: {
      escapeValue: false
    },
    lng: localStorage.getItem('language') || 'en',
    resources: {
      en: {
        translation: TRANSLATIONS_EN
      },
      ro: {
        translation: TRANSLATIONS_RO
      }
    }
  });

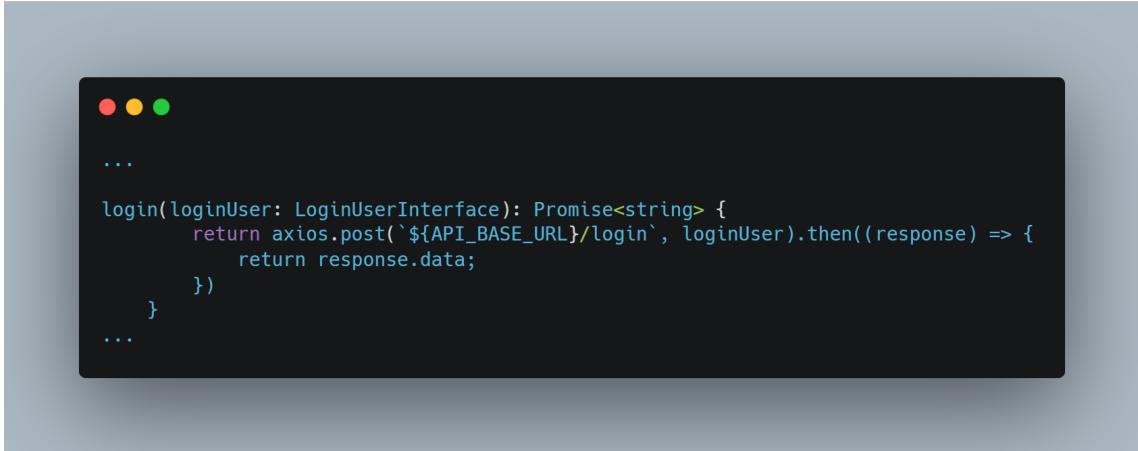
export default i18n;

```

Figura 5.21: Configurare i18n

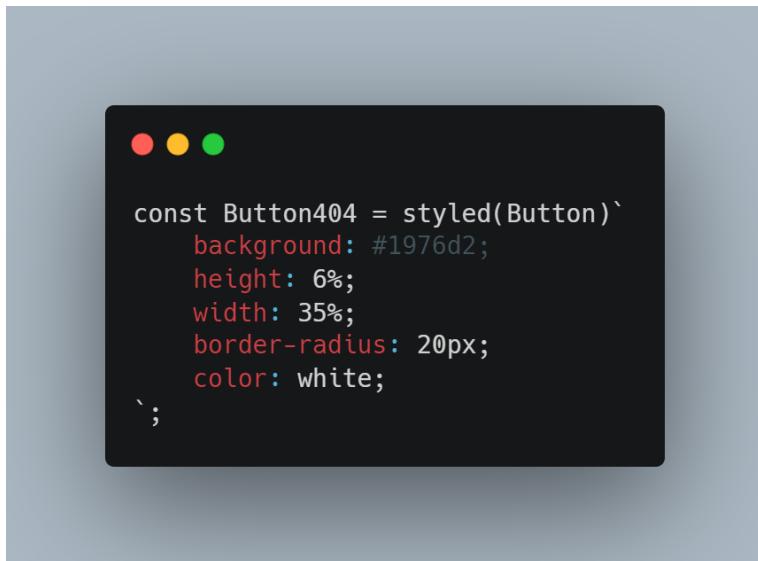
Un exemplu de aplicare a stilului pe o componentă se poate vedea în figura 5.23, iar rezultatul final în figura 5.24. Se crează o componentă reutilizabilă numită `Button404` (această componentă apare în pagina `NotFound`) și moștenește stilul inițial al componentei din AntD prin `styled(Button)`.

Customizare componente începe acum, când este setată o culoare de background pentru buton și scrisul trebuie să fie alb (`color: white`). Se setează dimensiunilor butonului și un `border-radius` de 12 pixeli pentru a rotunji marginile butonului.



```
...  
login(loginUser: LoginUserInterface): Promise<string> {  
    return axios.post(`${API_BASE_URL}/login`, loginUser).then((response) => {  
        return response.data;  
    })  
...  
}
```

Figura 5.22: Structură request Axios



```
const Button404 = styled(Button)`  
background: #1976d2;  
height: 6%;  
width: 35%;  
border-radius: 20px;  
color: white;  
`;
```

Figura 5.23: Mod de aplicare a stilului folosind styled-components pe o componentă AntD

5.5. Analizarea textului

Aplicația realizează inferență pentru analiza sentimentelor din textele analizate, pentru summarizarea textului și pentru extragerea cuvintelor cheie.

Hugging Face Inference API este accesat printr-un request HTTP, incluzând în body-ul requestului textul de la utilizator.

Cuvintele cheie sau expresiile sunt extrase din textul original și sunt evidențiate atât în partea de text original, cât și în partea de text summarizat din pagina de Dashboard din frontend.

După extragerea cuvintelor cheie, acestea sunt salvate în baza de date, de unde se fac următoarele procesări, în funcție de inputul utilizatorului:

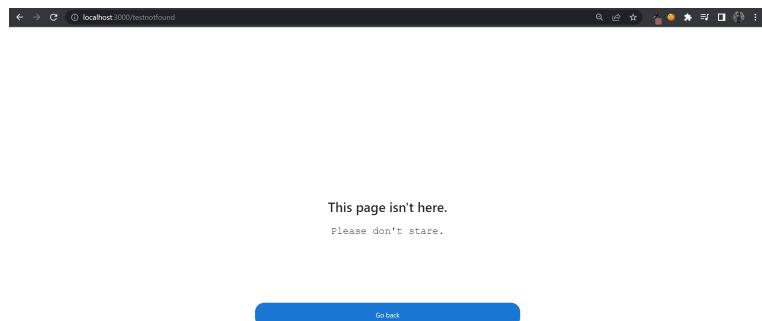


Figura 5.24: Stilul aplicat pe butonul din pagina de Not Found

Pentru generarea graficului cu cele mai populare subiecte dintr-o anumită perioadă, se transmite o cerere cu perioada la server.

Mai departe, serverul caută în baza de date înregistrările care corespund perioadei căutate și le contorizează în funcție de nume. La final, rezultatul transmis este o listă de 10 elemente care conține obiecte constituite din numele cuvintelor și numărul de apariții, în acest fel fiind generat graficul de trending.

Pentru generarea graficului de evoluție al unui cuvânt cheie sau expresii, inputul trebuie să conțină perioada și cuvântul/cuvintele cautate. Se face o căutare a înregistrărilor din baza de date care corespund perioadei căutate și al cărui nume corespunde cu inputul utilizatorului.

Înregistrările care corespund căutării sunt ordonate apoi după data în care au fost salvate. Rezultatul acestui filtru este o listă care conține obiecte constituite din ziua înregistrării și numărul de apariții din acea zi.

WordCloud-ul este o colecție de cuvinte sau expresii de diferite mărimi.

Pentru generarea acestuia, la fel ca pentru graficul de trending, este nevoie de numărul de apariții pentru fiecare termen existent în înregistrări. Diferența dintre cele două este că graficul de trending care este limitat la 10 elemente.

În funcție de numărul de înregistrări pentru un anumit termen, dimensiunea acestuia o sa varieze atunci când este afișat.

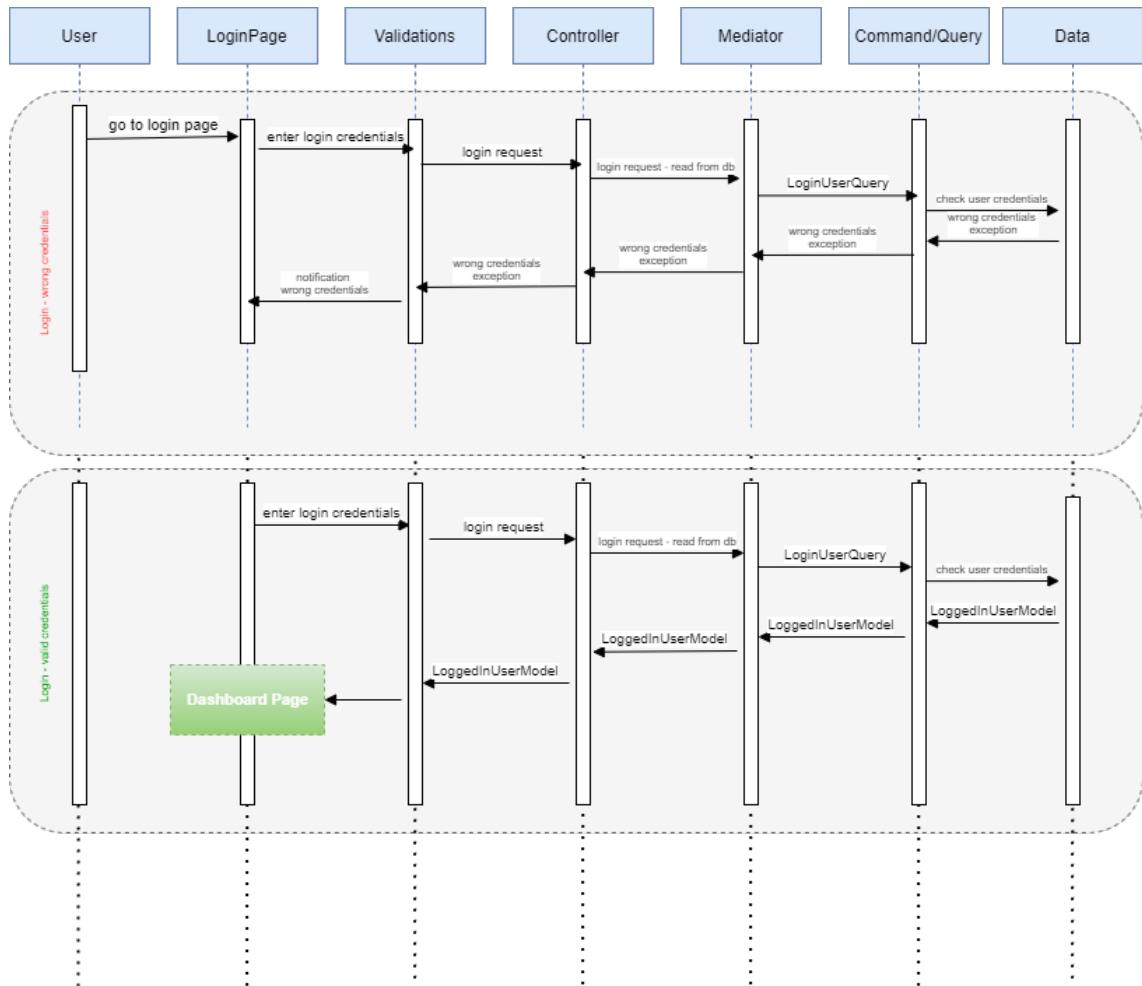


Figura 5.25: Diagramă de secvență pentru logarea în aplicație

Capitolul 6. Testare și Validare

6.1. Testare backend

Pentru partea de backend, s-a realizat testare automată a API-urilor, utilizând Swagger. Conform definiției de pe Wikipedia pentru API [30], este o interfață de programare a aplicației, care permite mai multor dispozitive să comunice între ele.

Așadar, trebuie să ne asigurăm că informațiile transmise bidirectional sunt corecte, atât din punct de vedere al URL-ului utilizat (acesta este modul în care se va face legătura între backend și frontend), cât și din punct de vedere al răspunsului și formatul acestuia.

6.1.1. Testare login cu credențiale greșite

În cazul încercării de logare în aplicație când utilizatorul introduce credențialele greșite, răspunsul va fi sugestiv și nu va fi logat în aplicație, după cum se poate vedea în figurile 6.1 și 6.2.



Figura 6.1: Request - Login cu credențiale greșite



Figura 6.2: Response - Login cu credențiale greșite

6.1.2. Testare login cu credențiale corecte

În cazul încercării de logare în aplicație când utilizatorul introduce credențialele corecte, răspunsul trimis la frontend va fi compus din Id-ul userului pentru a fi salvat în context și token-ul JWT generat, după cum se poate vedea în figurile 6.3 și 6.4.

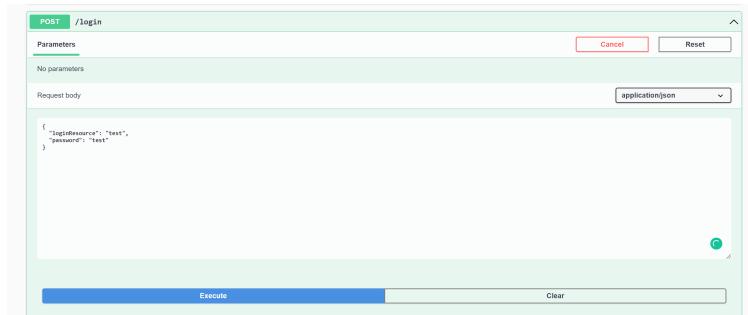


Figura 6.3: Request - Login cu credențiale corecte

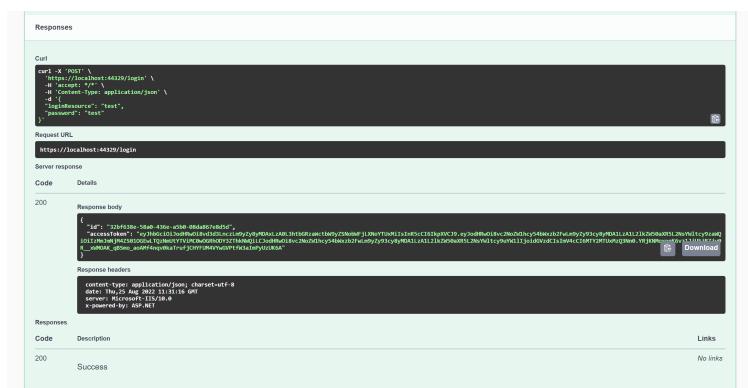


Figura 6.4: Response - Login cu credențiale corecte

6.1.3. Testare editare detalii utilizator când un câmp este gol

În cazul încercării de editare a detaliilor utilizatorului când un câmp este gol, deși ar trebui să conțină informație sub formă de string, răspunsul va fi generat de constrângerea validatorului creat cu ajutorul Fluent Validation, după cum se poate vedea în figurile 6.5 și 6.6.

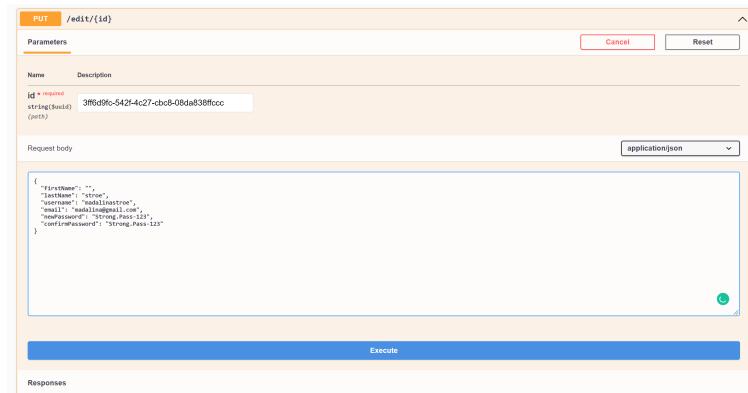


Figura 6.5: Request - Editare detalii personale când un câmp este gol

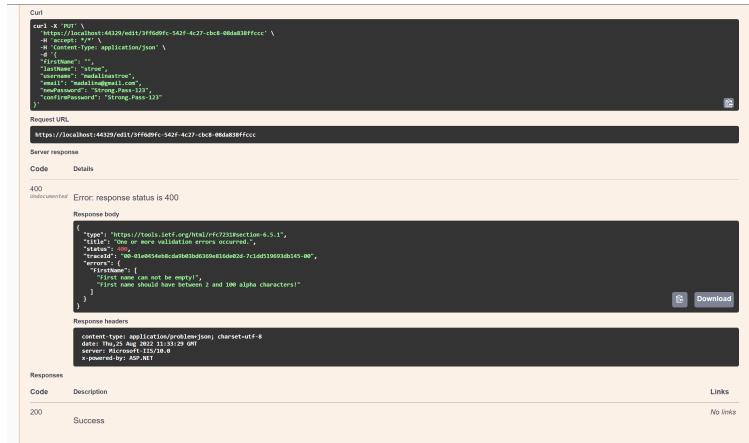


Figura 6.6: Response - Editare detalii personale când un câmp este gol

6.2. Testare frontend

Pentru partea de frontend, testarea s-a realizat manual, la nivelul fiecărei pagini, pentru a testa dacă funcționalitatea este cea așteptată în funcție de inputul utilizatorului. Rezultatele se pot observa în figurile 6.7, 6.8 și 6.9.

Descriere	Pagina	Input utilizator	Rezultat așteptat
Verificare validate în frontend pentru formatul corect al inputului din câmpuri	Creare cont nou	Un câmp nu are formatul corespunzător (ex: câmpul de email)	Validare la nivel de field, atenționează utilizatorul că inputul nu respectă formatul corect
Verificare validate în frontend pentru formatul corect al inputului din câmpuri	Creare cont nou	Un câmp nu are numărul de caractere corespunzător, cuprinzând minimul și maximul permis	Validare la nivel de field, atenționează utilizatorul că câte caractere sunt necesare pentru a trece validarea
Verificare validate în frontend pentru inputul din câmpuri	Creare cont nou	Unul sau mai multe câmpuri nu sunt completeate	Validare la nivel de field, atenționează utilizatorul că trebuie introdus input pentru a procesa cererea
Verificare validate pentru unicitatea detaliilor	Creare cont nou	Câmpul de email sau nume de utilizator conține o informație atribuită altui cont	Validare din server, utilizatorul este notificat că atributul trebuie să fie unic și că cel introdus deja aparține altui cont
Verificare validate pentru inputul introdus de utilizator	Creare cont nou	Câmpurile sunt completeate cu informații corecte	Redirecționare la pagina de login
Verificare validate în frontend pentru inputul din câmpuri	Logare	Unul sau mai multe câmpuri nu sunt completeate	Validare la nivel de field, atenționează utilizatorul că trebuie introdus input pentru a procesa cererea
Verificarea corectitudinii credențialelor introduse	Logare	Credențiale greșite introduse	Validare din server, utilizatorul este notificat că sunt greșite credențialele introduse
Verificarea corectitudinii credențialelor introduse	Logare	Credențiale corecte introduse	Redirecționare la pagina de Dashboard

Figura 6.7: Testare modul frontend - partea 1

6.3. Testare analiză a textului

Pe partea de analiză de text, testarea s-a realizat prin compararea empirică a mai multor modele din Hugging Face utilizând diferite texte.

Descriere	Pagina	Input utilizator	Rezultat așteptat
Verificare validări în frontend pentru inputul din caseta text	Dashboard	Nu este introdus niciun text	Validare la nivel de field, atenționează utilizatorul că trebuie introdus input pentru a procesa cererea
Verificare validări în frontend pentru inputul din caseta text	Dashboard	Este introdus text	Se procesează cererea și când răspunsul a ajuns în modulul de front-end, apare un buton sprijin pagină cu mai multe detalii despre analiză
Generare grafic cu numărul de apariții al cuvintelor cheie extrase din text	Dashboard - Analiză detaliată	Selectare perioadă și cuvintele cheie pentru care se dorește afișarea	Se vor afisa pe grafic data și numărul de apariții al cuvintelor, pe zile
Salvare articol analizat	Dashboard - Analiză detaliată	Se apasă pe butonul de salvare articol, dacă se dorește memorarea lui în istoricul personal	În tabelul din pagina de istoric personal apare articolul dacă a fost salvat, altfel, rezultatul analizei a fost decartat
Vezi mai multe detalii despre un item salvat	Istoric personal	Se apasă pe butonul de See more	Redirectionare la pagina de Dashboard cu analiza detaliată, de această dată nemaieșind un buton de salvare
Sterge detalii despre un item salvat	Istoric personal	Se apasă pe butonul de Delete	Modal de validare a decizie, dacă se dorește stergerea. Dacă utilizatorul acceptă, itemul va fi scos din tabel și baza de date
Top 10 cele mai populare cuvinte cheie apărute în textele analizate	Trending	Userul nu selectează nicio perioadă sau nu introduce niciun cuvânt și apasă pe butonul de Submit	Validare la nivel de field, atenționează utilizatorul că trebuie introdus input pentru a procesa cererea

Figura 6.8: Testare modul frontend - partea 2

Descriere	Pagina	Input utilizator	Rezultat așteptat
Top 10 cele mai populare cuvinte cheie apărute în textele analizate	Trending	Userul nu selectează nicio perioadă sau nu introduce niciun cuvânt și apasă pe butonul de Submit	Validare din server, utilizatorul este notificat că sunt nu există date înregistrate pentru acea perioadă
Top 10 cele mai populare cuvinte cheie apărute în textele analizate	Trending	Userul introduce o perioadă în care există date salvate și apasă pe buton	Se vor afisa pe grafic cuvântul cheie și numărul de apariții
Evoluția numărului de căutări pentru un cuvânt cheie introdus de utilizator într-o anumită perioadă	Evoluție		Validare la nivel de field, atenționează utilizatorul că trebuie introdus input pentru a procesa cererea
Evoluția numărului de căutări pentru un cuvânt cheie introdus de utilizator într-o anumită perioadă	Evoluție	Userul introduce o perioadă în care există date salvate și cuvântul, apoi apasă pe buton	Validare din server, utilizatorul este notificat că sunt nu există date înregistrate pentru acea perioadă
Evoluția numărului de căutări pentru un cuvânt cheie introdus de utilizator într-o anumită perioadă	Evoluție	Userul introduce o perioadă în care există date salvate și cuvântul, apoi apasă pe buton	Se va afisa pe grafic numărul de apariții pe zile

Figura 6.9: Testare modul frontend - partea 3

Capitolul 7. Manual de Instalare și Utilizare

7.1. Instalare

7.1.1. Condiții prealabile generale

Resursele hardware necesare pentru instalarea și rularea aplicației sunt:

- Sistem de operare: Windows
- Arhitectura sistemului de operare: x64
- Memorie RAM: minim 8GB

7.1.2. Condiții prealabile pentru baza de date

Este preferat să fie instalate versiunile cele mai noi, *Long Term Support*.

Pentru instalarea resurselor necesare bazei de date, este nevoie de:

- **SQL Server**
 - Se descarcă din secțiunea Free Specialized Edition -> Developer
 - Se deschide fișierul descărcat și se urmează instrucțiunile până la tipul instalației. Aici se va selecta *Basic*
 - După ce instalarea a fost finalizată cu succes, se apasă butonul de *Install SSMS*
 - După ce SSMS a fost instalat, deschideți Microsoft SQL Server Management Studio
 - Pentru Server Type se poate păstra Database Engine, iar pentru Authentication se va selecta Windows Authentication
- **SQL Server Management Studio (SSMS)**
 - Se deschide fișierul descărcat, se va selecta locația unde se dorește a fi instalat, apoi se apasă pe butonul de Install
 - După ce instalarea e completa, e nevoie să fie restartat calculatorul

După ce toate instalările au fost finalizate cu succes, se pot crea baze de date, tabele și se pot efectua diferite comenzi și interogări.

7.1.3. Condiții prealabile pentru modulul de backend

Aplicația a fost creată utilizând .NET 6. De preferat, a se folosi această versiune, pentru că este posibil să fie modificări în versiunile ulterioare sau anterioare ce implică o sintaxă diferită. Spre exemplu, la trecerea de la .NET 5 la .NET 6, fișierele Program.cs și Startup.cs au fost comasate.

- **.NET 6**
 - Se apasă butonul Download *.NET SDK x64*, versiunea .NET 6.0, LTS
 - Din pagina unde s-a făcut redirectionarea după apăsarea pe buton, se va selecta Installer-ul potrivit pentru sistemul de operare

- După descărcare, se va deschide fișierul executabil și se vor urma instrucțiunile
- Se poate testa într-un command line dacă instalarea s-a făcut cu succes, scriind `dotnet -info`
- IDE: [Rider](#)
 - Se poate descărca de pe site-ul oficial sau din [Jetbrains Toolbox](#)
 - Se acceptă toate setările default

7.1.4. Condiții prealabile pentru modulul de frontend

- Node v16.13.0
 - Se poate descărca [de aici](#)
 - Se acceptă toate setările default
- Node Package Manager (npm) 8.1.0
 - Se poate descărca [de aici](#)
 - Se acceptă toate setările default
- Text Editor: Visual Studio Code
 - Se poate descărca [de aici](#)
 - Se acceptă toate setările default

7.2. Rulare

7.2.1. Setup baza de date

Dacă apar erori legate de baza de date, trebuie să se creeze un manual care să coincidă cu numele din fișierul appsettings.json din modulul de backend.

7.2.2. Rulare modul backend

La prima utilizare, trebuie să se importe proiectul și să se ruleze configurarea IIS Express. Se va rula comanda `dotnet ef database update` pentru a fi actualizată baza de date și pentru a fi create tabelele care vor fi ulterior folosite.

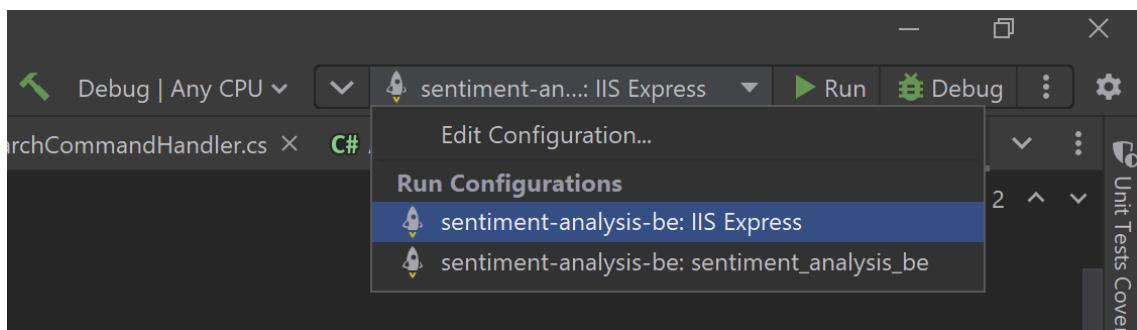


Figura 7.1: Configurări rulare backend

Aplicația va deschide în browser Swagger cu <https://localhost:44329/swagger/index.html>, un instrument care permite testarea și documentarea serviciilor REST.

Figura 7.2: Swagger, instrument pentru testarea endpointurilor aplicatiei

În acest moment, backend-ul rulează pe portul 44329 la care se va conecta modulul de frontend pentru a folosi API-ul creat.

7.2.3. Rulare modul frontend

După instalarea editorului de text Visual Studio Code, se va importa proiectul aici. Pentru a fi instalate toate dependințele necesare rulării proiectului, se va rula în terminal `npm init`. După ce toate dependințele au fost descărcate și instalate cu succes, se poate rula în terminal comanda `npm start`. După ce aplicația a pornit, se va deschide în browser un tab de React App cu `http://localhost:3000/login`.

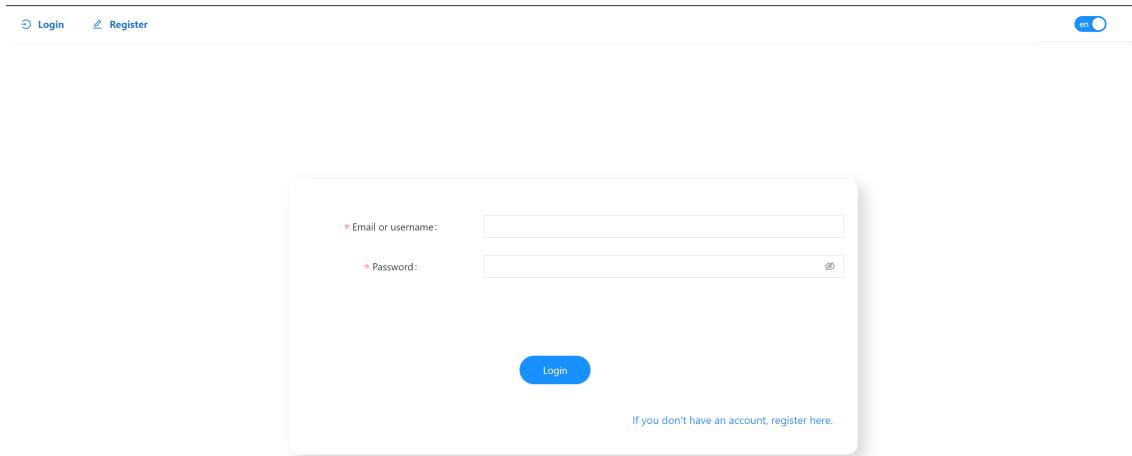


Figura 7.3: Rulare frontend

7.3. Manual de utilizare

La pornirea aplicației, dacă nu a folosit logat niciun alt utilizator înainte, prima pagină care se va deschide va fi pagina de login. În funcție de preferințele utilizatorului, acesta poate modifica limba aplicației cu ajutorul switch-ului de limbi. Această setare poate fi modificată și mai târziu.

Pentru crearea unui cont nou, se va apăsa pe butonul indicat în figura 7.4. După ce am ajuns pe pagina de Register sau Creare cont nou, utilizatorul va trebui să completeze datele. Aplicația oferă feedback vizual utilizatorului atunci când informațiile introduse de acesta în pagină nu respectă una dintre condițiile de număr minim, număr maxim de caractere format corect pentru email. Acest lucru se poate vedea în figura 7.5. După ce toate câmpurile au fost corect completate din punct de vedere al restricțiilor menționate mai sus, se mai fac 2 verificări - numele de utilizator și email-ul să fie unice. În cazul în care una din ele nu respectă condiția de unicitate, utilizatorul va fi notificat cu un mesaj corespunzător pentru a putea lua măsuri, după cum se poate vedea în figura.

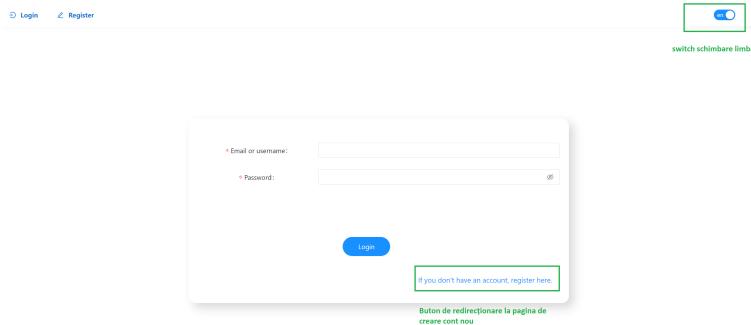


Figura 7.4: Pagina de login

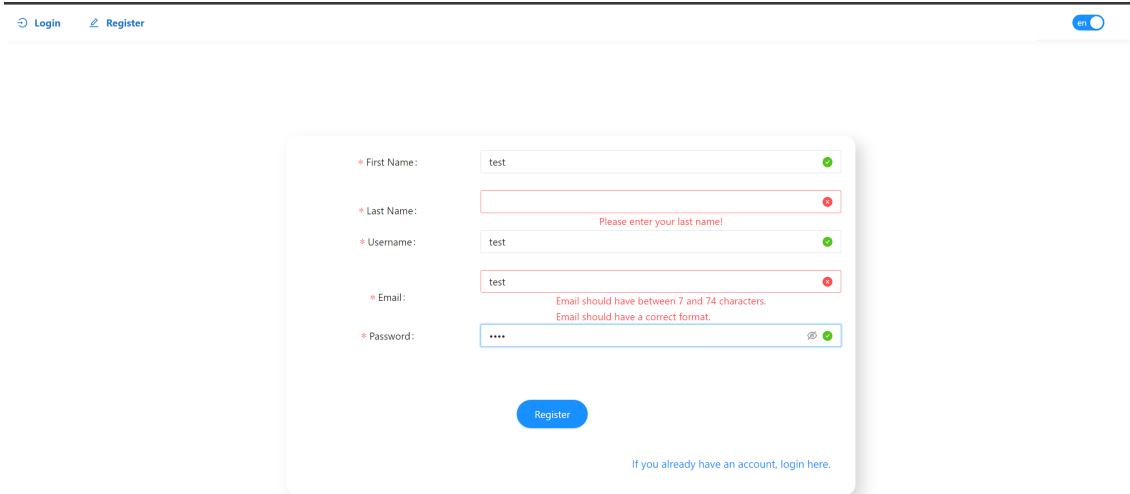


Figura 7.5: Pagina de creare cont nou

După ce contul a fost creat cu succes, se ajunge la pagina de login, unde trebuie completate credențialele pentru a putea intra în aplicație. În cazul în care credențialele nu sunt corecte, utilizatorul va fi notificat, după cum se poate vedea în figura 7.6.

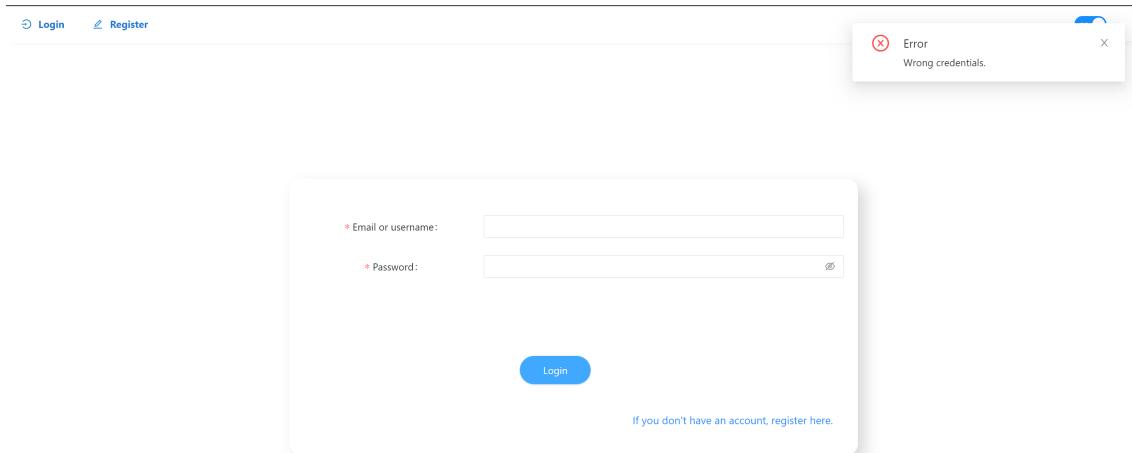


Figura 7.6: Credențiale greșite la logarea în aplicație

În acest moment, utilizatorul nu este logat și are acces doar la pagina de Login și pagina de Register, toate celelalte pagini din aplicație sunt inaccesibile, iar dacă va încerca să ajungă la una din ele din URL, acesta va fi redirectionat automat la pagina de login.

Când credențialele corecte au fost completate, utilizatorul va intra în aplicație. În acest moment, are acces la toate paginile și resursele din aplicație, dar nu are acces la paginile de Login și Register.

După logare, pe pagina de Dashboard, utilizatorul poate adăuga în caseta de text un text pe care dorește să îl analizeze. Nu poate trece mai departe, la componente de afișare a informațiilor detaliate până când nu există input în casetă; altfel, va primi un mesaj specific, cum e în figura 7.7.

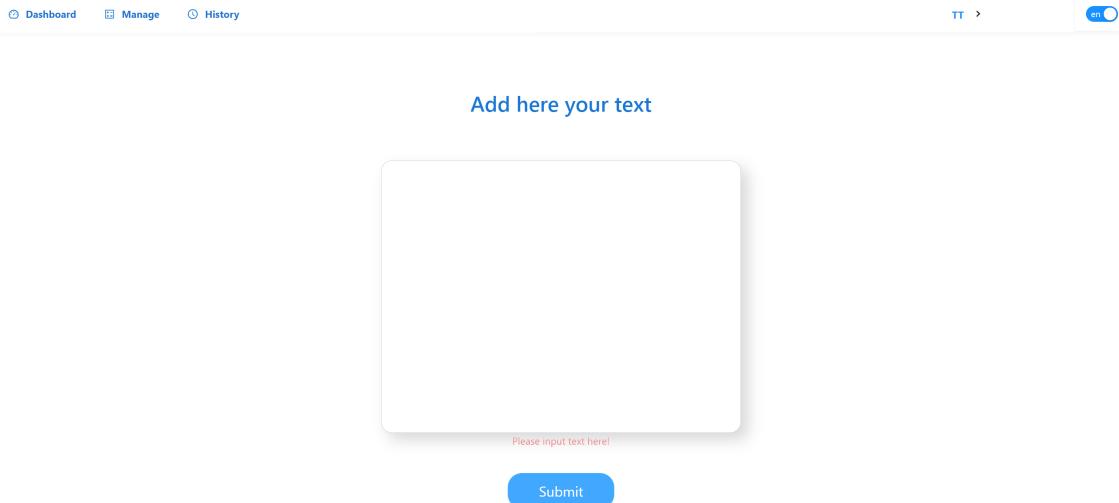


Figura 7.7: Încercare de a efectua o analiză fără input de la utilizator

După ce userul a adăugat textul pe care dorește să îl analizeze și apasă pe butonul SUBMIT, rezultatele vor apărea și un buton de mers la următoarea pagină se va încărca, cum apare în figura 7.8.

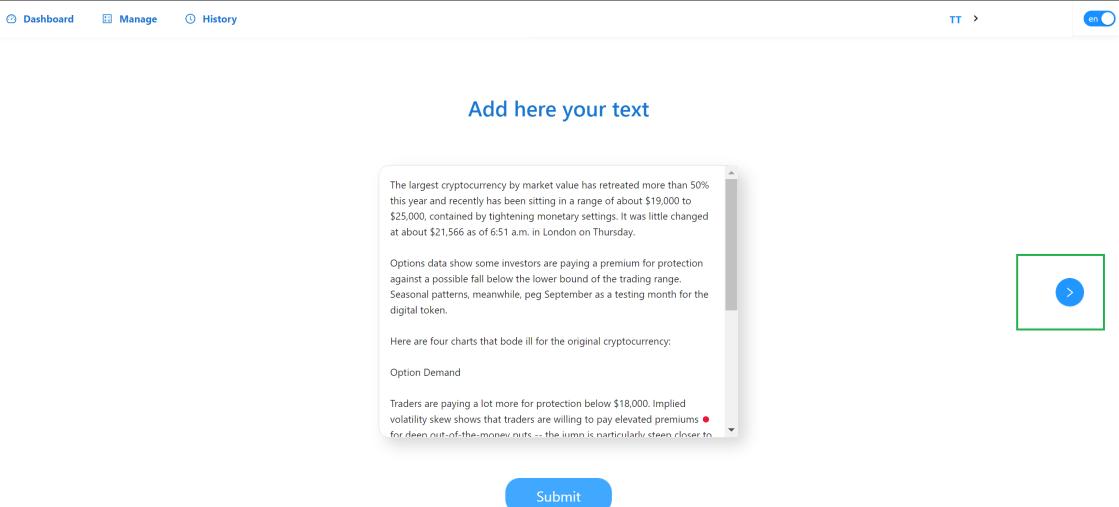


Figura 7.8: Buton pentru a merge în pagina de vizualizare a analizei detaliate a textului

După ce ajungem în pagina cu informații detaliate despre textul analizat, putem observa pe primul rând 2 carduri în care este afișat textul original și textul sumarizat. În următorul card, avem statisticile sentimentelor - pozitiv, neutru și negativ. Aceste informații se pot vedea în figura 7.9.

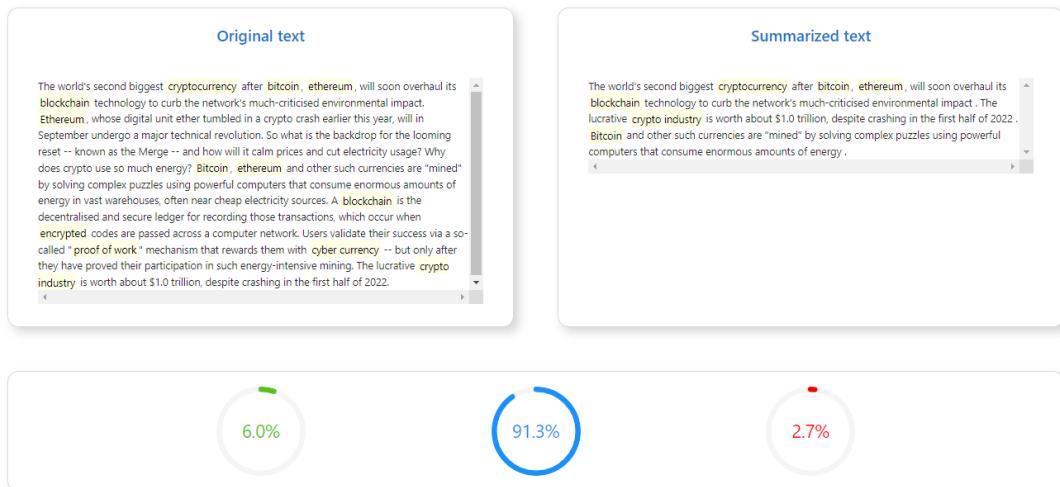


Figura 7.9: Rezultate analiza - partea 1

În ultimul card, care este unul interactiv, utilizatorul poate alege dintre cuvintele cheie extrase din text și perioada dorită pentru a genera graficul aparițiilor, după cum se poate vedea în figura 7.10.

În acest grafic, în primul dropdown sunt cuvintele cheie extrase din text. Acestea pot fi observate fiind evidențiate prin subliniere în textul original din figura 7.9. Rezultatul pe grafic poate fi observat în figura 7.11.



Figura 7.10: Rezultate analiza - partea 2



Figura 7.11: Rezultate analiza - evoluție cuvinte cheie

În acest punct, se poate observa butonul de salvare a articolului analizat, împreună cu rezultatele obținute în istoricul personal al utilizatorului, în figura 7.12. Informațiile pot fi redeschise ulterior, în orice moment, din pagina de Dashboard, cum apar în figura 7.13, apăsând butonul See More. De asemenea, articolele pot fi șterse din istoricul personal al utilizatorului apăsând butonul Delete.

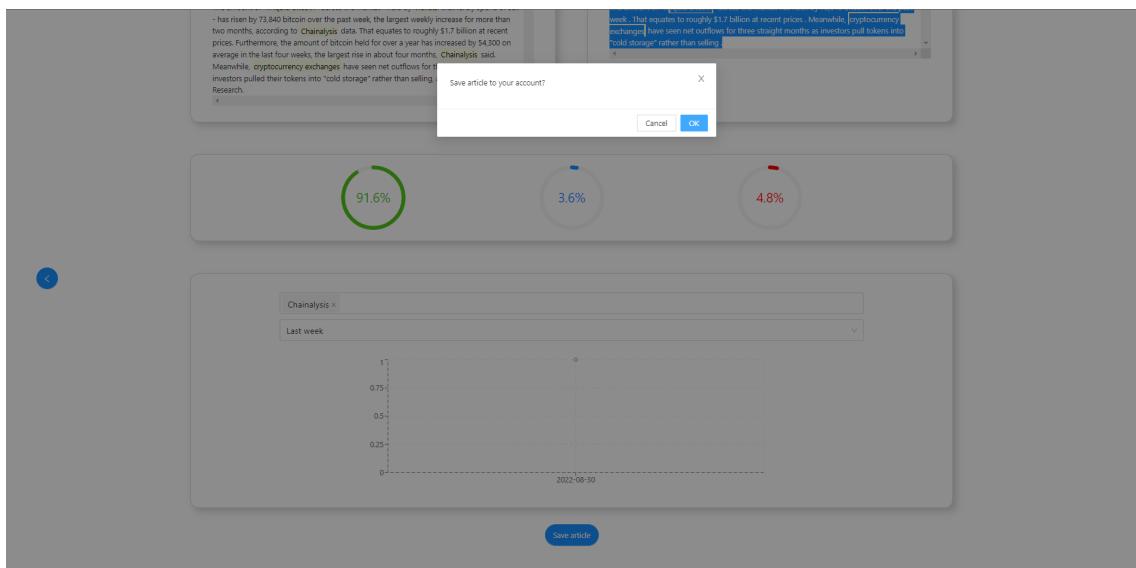


Figura 7.12: Buton de salvare articol

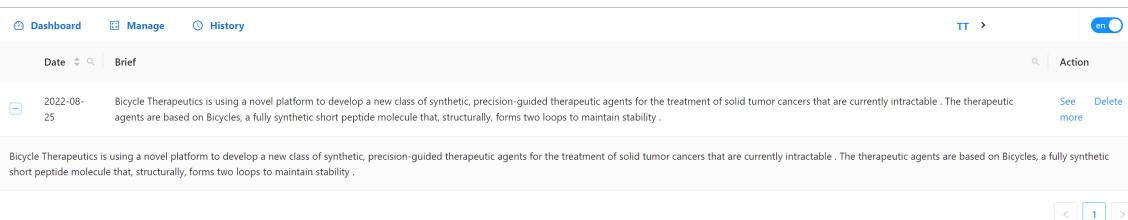


Figura 7.13: Istoricul analizelor de text salvate de utilizator

Pentru a edita informațiile contului utilizatorul, se accesează pagina de Profile. Câmpurile care pot fi modificate sunt Nume, Prenume, Nume de utilizator, Email, Parola. Singurele restricții în modificarea acestor câmpuri sunt ca Numele de utilizator și Email-ul să fie unice, neatribuite altui cont existent, altfel utilizatorul va fi notificat cu una dintre următoarele erori, ca în figura 7.14.

The screenshot shows a web application interface. At the top, there are three navigation links: 'Dashboard', 'Manage', and 'History'. Below them is a modal window titled 'Edit account'. Inside the modal, there are several input fields with validation icons: 'First Name' (green checkmark), 'Last Name' (green checkmark), 'Email' (green checkmark), and 'Confirm password' (green checkmark). There are two empty fields for 'New password' and 'Confirm password'. A red error message box at the top right of the modal says 'Error' with a red X icon and the text 'Username already exists!'. At the bottom of the modal is a blue 'Submit' button.

Figura 7.14: Utilizatorul încearcă să își schimbe username-ul, dar acesta este deja atribuit altui cont

După ce utilizator a ales un nou username și/sau email care respectă condiția de unicitate, acesta va fi notificat că actualizarea detaliilor contului s-a făcut cu succes, cum apare în figura 7.15.

This screenshot is similar to Figure 7.14, showing the 'Edit account' modal. The input fields are identical: 'First Name' (green checkmark), 'Last Name' (green checkmark), 'Email' (green checkmark), and 'Confirm password' (green checkmark). The 'New password' and 'Confirm password' fields are empty. A green success message box at the top right of the modal says 'Success' with a green checkmark icon and the text 'User details changed!'. At the bottom of the modal is a blue 'Submit' button.

Figura 7.15: Actualizare detalii cont cu succes

În pagina de Manage, utilizatorul poate vedea topicele cele mai populare, adică cele mai des apărute cuvinte cheie sau expresii în analizele efectuate și de restul utilizatorilor. În cazul în care utilizatorul a selectat o perioadă în care nu sunt înregistrate date, va fi notificat de acest lucru, ca în figura 7.16, altfel rezultatele vor fi afișate pe grafic, ca în figura 7.17.

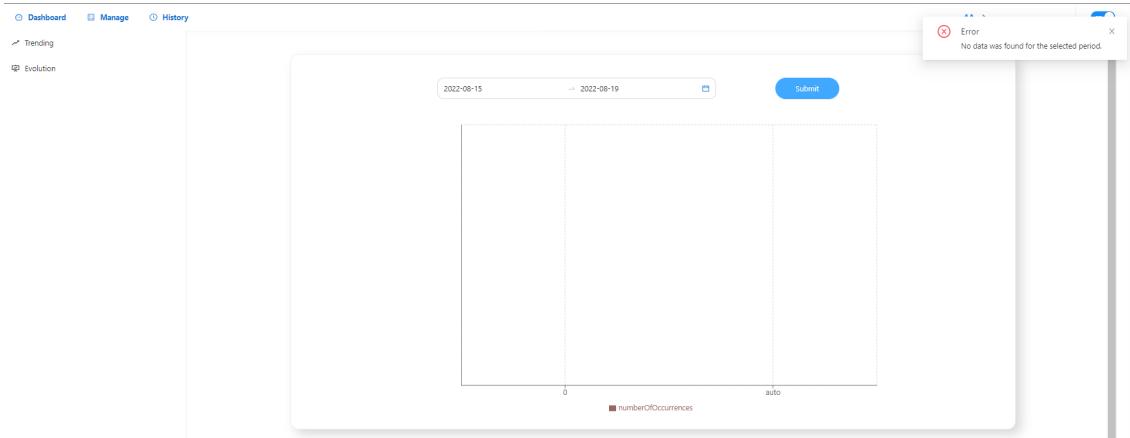


Figura 7.16: Grafic pentru top 10 cele mai populare subiecte - niciun rezultat

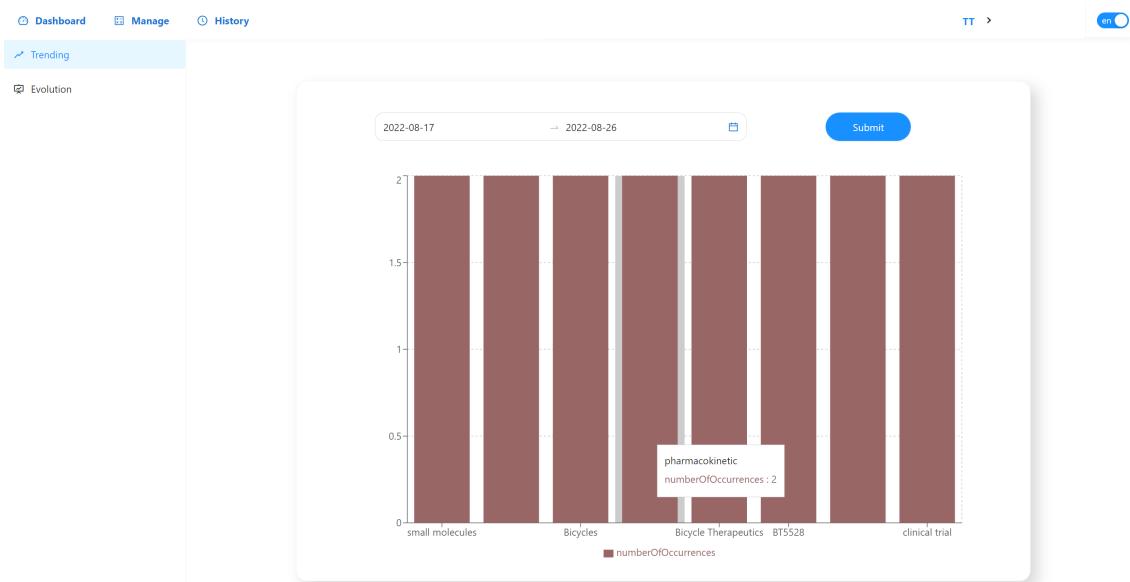


Figura 7.17: Grafic pentru top 10 cele mai populare subiecte

Alt element important pe care utilizatorul îl poate vedea aici este un WordCloud, care este, de fapt, o colecție de cuvinte sau expresii de diferite mărimi, ca în figura 7.18. Mărimea este determinată de numărul total de apariții.

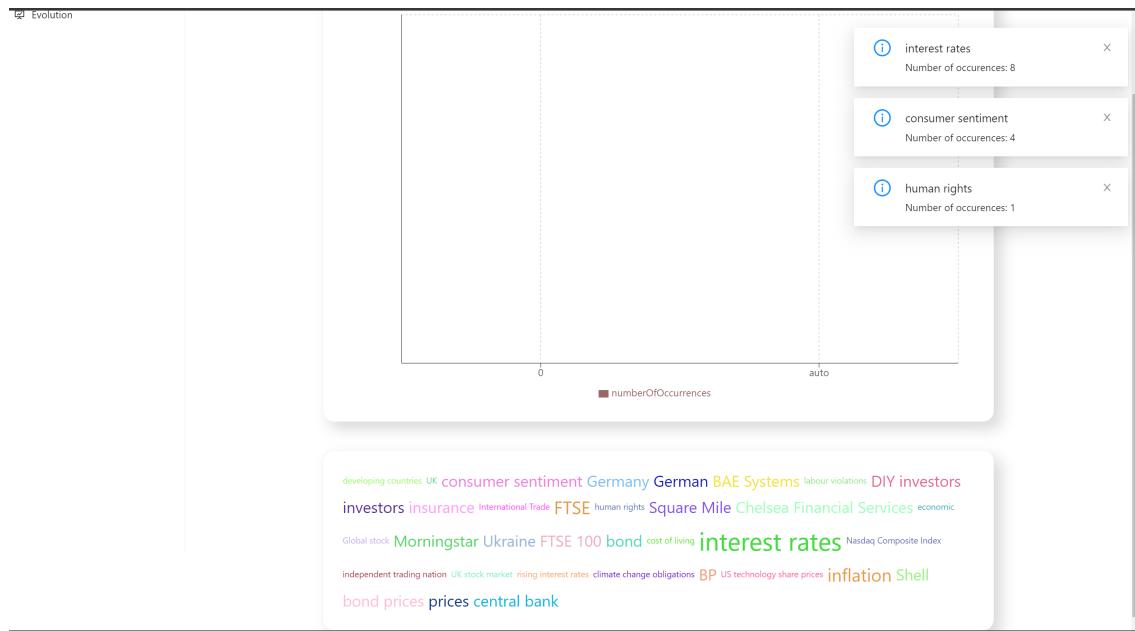


Figura 7.18: WordCloud

Evoluția unui cuvânt cheie sau expresie poate fi observată într-o perioadă de timp selectată.

La fel ca în cazul topicelor populare, dacă nu există date salvate pentru perioada sau inputul selectat, utilizatorul va fi notificat de acest lucru, ca în figura 7.19.

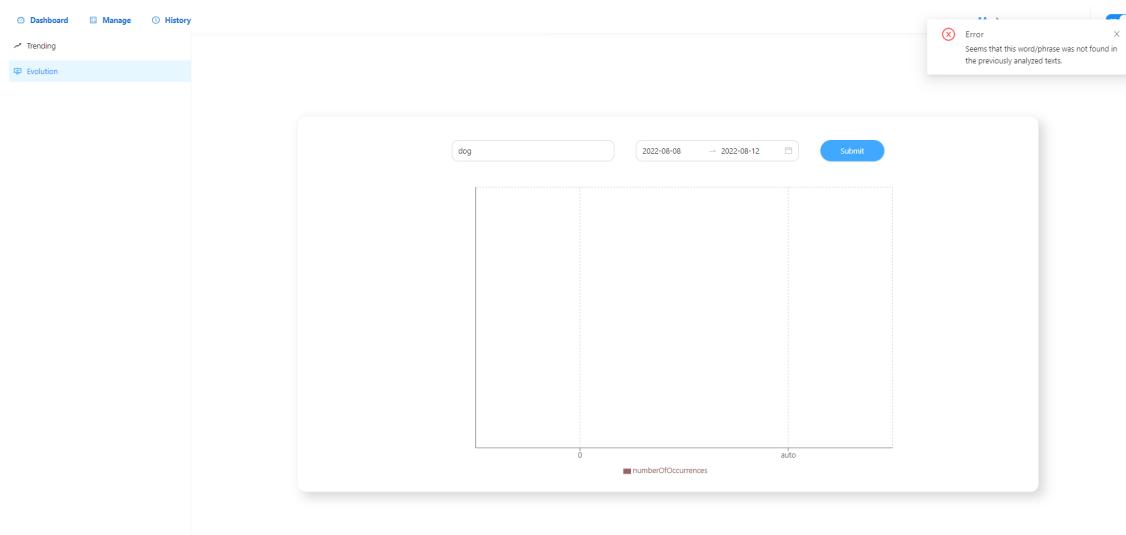


Figura 7.19: Evoluția unui cuvânt cheie într-o anumită perioadă - niciun rezultat

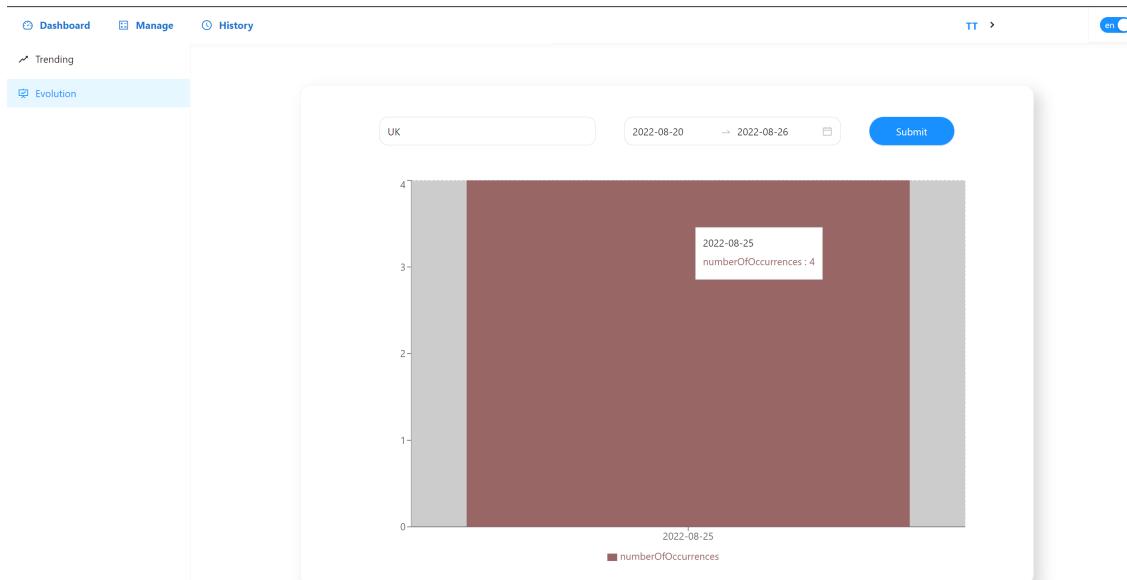


Figura 7.20: Evoluția unui cuvânt cheie într-o anumită perioadă

Altfel, în grafic se poate observa evoluția pe zile, ca în figura 7.20.

Capitolul 8. Concluzii

8.1. Analiza rezultatelor obținute

Aplicația realizată îndeplinește obiectivele setate la început, fiind un instrument de analiză detaliată a unui text cu specific financiar.

Reușind integrarea modelelor de procesare de limbaj natural în aplicația web, aceasta oferă, pe lângă funcționalități generale ale unei aplicații web, analiza sentimentelor financiare dintr-un text, sumarizarea textului și extragerea cuvintelor cheie.

Unul dintre obiectivele setate la început pentru aplicație era ca utilizatorul să poată lua o decizie financiară informată. Cu ajutorul analizei sentimentelor financiare, sunt obținute scoruri pentru fiecare dintre sentimentele negativ, neutru, pozitiv și acestea pot influența decizia utilizatorului.

De multe ori, un scor pozitiv ridicat poate fi percepțut ca un lucru bun, la fel cum un scor negativ ridicat poate fi percepțut ca un lucru rău. Pentru cazurile excepționale când un scor negativ ridicat poate însemna și un lucru benefic pentru investitor, acesta trebuie determinat din context.

Textele analizate pot avea diferite dimensiuni și posibil, doar anumite părți sunt interesante pentru investitor. Deci dimensiunea unui text e măsurată în timp și se dorește un răspuns într-un timp scurt.

Autorul scrie în [31] următoarea afirmație: *"If you are looking for a good rule of thumb, write content between 1,500 and 3,000 words."*. Continuând apoi cu o analiză a timpului petrecut citind un anumit număr de cuvinte [32], considerând o viteza medie, pentru aproximativ 2000 de cuvinte o persoană va petrece aproximativ 6.7 minute per articol.

A citi nu e nici pe de parte un lucru rău, doar că aici se doresc informații în timp cât mai scurt pentru că se urmărește momentul potrivit. Așadar, intervine sumarizarea textului, alt obiectiv al acestei aplicații. Utilizatorul va reduce timpul petrecut pentru citirea unui articol, extragând ideile principale, astfel încât să își poată da seama din context de motivul scorului sentimentelor rezultat.

Tot în acest scop, aplicația realizează și extragerea cuvintelor cheie, pentru a oferi o privire de perspectivă asupra a ce se întâmplă în acest moment.

Cu rezultatele obținute, utilizatorul poate vedea evoluția interesului general pentru un anumit subiect, cele mai populare 10 subiecte și anumite definiții pentru cuvintele cheie extrase.

Pe lângă aceste funcționalități, aplicația web se concentrează și pe partea de securitate; încrucișând utilizatorul poate să își salveze documente în istoricul personal, accesul altor persoane cu intenții rele trebuie să fie restricționat. Acest lucru e realizat cu ajutorul tokenilor, rutelor protejate și parolele unice.

8.2. Dezvoltări și îmbunătățiri ulterioare

Pentru partea de îmbunătățiri ulterioare, în primul rând, o funcționalitate necesară consider că ar fi o analiză a sentimentelor pe text, pe anumite părți din text selectate de utilizator.

Spre exemplu să existe 2 cursoare, unul pentru a fi poziționat la începutul textului care se dorește a fi analizat și alt cursor pentru a fi poziționat la finalul textului. Momen-tan, în aplicație, sentimentul finanțiar este obținut pentru tot textul analizat, așa că o îmbunătățire ar fi selectarea unui sub-text, spre exemplu, frază sau paragraf.

Poate fi introdus un mod de vizitator unde un utilizator neautentificat poate să analizeze un număr fix de articole, dar pentru stocarea rezultatelor sau vizualizarea graficelor să fie necesară autentificarea.

Tot pe partea de autentificarea, poate fi utilă și logarea în aplicație folosind Google.

O altă îmbunătățire care poate fi adusă poate fi reprezentată de sugestia de articole relevante de pe Internet pentru cuvintele cheie sau expreziile identificate.

De asemenea, se poate introduce și varianta de analizare a unui text introducând doar URL-ul acestuia. Pentru a fi posibil acest lucru, modelul de analiză ar trebui să accepte un text întreg și să nu fie limitat la un număr fix de tokeni. O posibilă soluție poate fi găsită aici [33].

Bibliografie

- [1] "Client-Server Application". [Online]. Available: https://madooei.github.io/cs421_sp20_homepage/client-server-app/
- [2] "Web Application Architecture in 2022: Moving in the Right Direction". [Online]. Available: <https://mobidev.biz/blog/web-application-architecture-types>
- [3] "Mediator Design Pattern". [Online]. Available: <https://www.scaler.com/topics/design-patterns/mediator-design-pattern/>
- [4] "CQRS and MediatR in ASP.NET Core". [Online]. Available: <https://code-maze.com/cqrs-mediatr-in-aspnet-core/>
- [5] "The Most Popular Front-end Frameworks in 2022". [Online]. Available: <https://stackdiary.com/front-end-frameworks/>
- [6] J. Alammar. The illustrated transformer. [Online]. Available: <https://jalammar.github.io/illustrated-transformer/>
- [7] N. P. Ashish Vaswani, Noam Shazeer, "Attention is all you need," 2017.
- [8] "Webinar: How NLP and BERT will change the language game". [Online]. Available: <https://peltarion.com/webinars/nlp-and-bert>
- [9] "Transformer's Self-Attention Mechanism Simplified". [Online]. Available: <https://vaclavkosar.com/ml/transformers-self-attention-mechanism-simplified>
- [10] D. T. Araci, "Finbert: Financial sentiment analysis with pre-trained language models," 2019.
- [11] A. H. Yi Yang, Mark Christopher Siy UY, "Finbert: A pretrained language model for financial communications," 2020.
- [12] N. G. Mike Lewis, Yinhan Liu, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," 2019.
- [13] "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension". [Online]. Available: <https://crossminds.ai/video/bart-denoising-sequence-to-sequence-pre-training-for-natural-language-generation-translation-and-comprehension>
- [14] Y. L. Mandar Joshi, Danqi Chen, "Spanbert: Improving pre-training by representing and predicting spans," 2020.
- [15] "Transformers BART Model Explained for Text Summarization". [Online]. Available: <https://www.projectpro.io/article/transformers-bart-model-explained/553>

- [16] "Metric: rouge". [Online]. Available: <https://huggingface.co/spaces/evaluate-metric/rouge>
- [17] R. A. Mayank Kulkarni, Debanjan Mahata, "Learning rich representation of key-phrases from text," 2022.
- [18] "Guideline: Detail Use Cases and Scenarios". [Online]. Available: https://www.utm.mx/~caff/doc/OpenUPWeb/openup/guidances/guidelines/detail_ucs_and_scenarios_6BC56BB7.html?proc=_0o9ygclgEdmt3adZL5Dmdw&path=_4BJ_YCxSEdqjsdw1QLH_6Q
- [19] "HTTP - Overview". [Online]. Available: https://www.tutorialspoint.com/http/http_overview.htm
- [20] "The Hypertext Transfer Protocol (HTTP)". [Online]. Available: <https://researchhubs.com/post/computing/web-application/the-hypertext-transfer-protocol-http.html>
- [21] "What Is TCP/IP?". [Online]. Available: <https://www.avg.com/en/signal/what-is-tcp-ip>
- [22] "The TCP/IP Protocol Framework". [Online]. Available: <https://sourcedaddy.com/windows-xp/the-tcp-ip-protocol-framework.html>
- [23] "What is SMTP". [Online]. Available: <https://serversmtp.com/what-is-smtp/>
- [24] "Client-Server Architecture". [Online]. Available: <https://kitrum.com/blog/client-server-architecture-advantages-and-disadvantages/>
- [25] "DbDiagram". [Online]. Available: <https://dbdiagram.io/home>
- [26] Describe benefits and pitfalls of the mediator pattern. [Online]. Available: <https://www.gofpatterns.com/design-patterns/module6/benefits-pitfalls-mediatorPattern.php>
- [27] "Carbon". [Online]. Available: <https://carbon.now.sh/>
- [28] How to cqrs with mediatr in asp.net core easy. [Online]. Available: <https://referbruv.com/blog/implementing-cqrs-using-mediator-in-aspnet-core-explained/>
- [29] "Playing with the Context API in React 16.3". [Online]. Available: <https://www.carlrippon.com/playing-with-the-context-api-in-react-16-3/>
- [30] "API". [Online]. Available: <https://en.wikipedia.org/wiki/API>
- [31] Matthew Royse. "What's the Ideal Length for an Article?" Mar 17, 2021. [Online]. Available: <https://medium.com/technical-excellence/whats-the-ideal-length-for-an-article-e1dd8f03daf4>
- [32] "How Long Does It Take to Read 200 Words?". [Online]. Available: <https://titlecapitalisation.com/reading-time/200-words/>
- [33] "How to Apply Transformers to Any Length of Text". [Online]. Available: <https://towardsdatascience.com/how-to-apply-transformers-to-any-length-of-text-a5601410af7f>